**Sentiment Analysis on Twitter**

**Ashwal H Vishwanath, Goutam A Bhavimani, and V Yashaswini Naidu**

[ashwalgowdaalike@gmail.com](mailto:ashwalgowdaalike@gmail.com)

[edithgoutam@gmail.com](mailto:edithgoutam@gmail.com)

[yashaswininaidu.08@gmail.com](mailto:yashaswininaidu.08@gmail.com)

***Abstract*—In this introductory paper, we explain the process of storing, preparing and analyzing twitter streaming data, then we examine the methods and tools available in python programming language to visualize the analyzed data Twitter’s popularity as an information source has led to the development of applications and research in various domains. In our paper, we focus on using Twitter, the most popular microblogging platform, for the task of sentiment analysis. We show how to automatically collect a compilation for sentiment analysis. Using the compilation, we put up sentimental classifier, which is able to determine as positive**

**, negative and neutral sentiments for a document. Experimental evaluations show that our proposed techniques are efficient and performs better than previously proposed methods.**

**Keywords—Twitter; Data Analysis; Python;**

1. INTRODUCTION

Microblogging today has become a very popular commu- nication tool among Internet users. With the booming of e-commerce, people are getting used to consuming online and writing comments about their purchase experiences on merchant/review Websites. These opinionated contents are valuable resources both to future customers for decision- making and to merchants for improving their products and/or service.

Twitter has emerged as one such major micro-blogging website, having over 100 million users generating over 500 million tweets every day. With such large audience, Twitter has consistently attracted users to convey their opinions and perspectives about any issue, brand, company or any other topic of interest. Due

to this reason, Twitter is used as an informative source by many organizations, institutions and companies.

In our paper, we study how microblogging can be used for

sentimental analysis purposes. We show how to use Twitter as a compilation for sentimental analysis. We use microblogging and more particularly Twitter for the following reasons:

* Social Media platforms are used by different people to express their opinion about different topics, thus it is a valuable source of people’s opinions.
* On twitter, people share the opinions in the form of tweets. This enormous data can be used for sentimental analysis.
* Twitter platform varies from common people to celebri- ties, company representatives, politicians4, and even country presidents. Therefore, it is possible to collect tweets of users from different social and interest groups.

Ashwal H Vishwanath (e-mail: ashwalgowdaalike@gmail.com) Goutam A Bhavimani (e-mail: goutamab007@gmail.com)

1. COLLECTING DATA

Using your Twitter account, you will need to apply for Developer Access and then create an application that will generate the API credentials that you will use to access Twitter from Python.

1. *Accessing data*

Data in the form of raw tweets is acquired by using the Python library “tweepy” which provides a package for simple twitter streaming API. Using the Python library “tweepy” which provides a package for simple twitter streaming API

.This API allows two modes of accessing tweets: Sam- pleStream and FilterStream.

SampleStream simply delivers a small, random sample of all the tweets streaming at a real time. FilterStream delivers tweets which match a certain criteria.

To install Tweepy we can use below command.

*pip install tweepy*

1. *Tweets Retrieval*

Since human labelling is an expensive process we further filter out the tweets to

be labelled so that we have the greatest amount of variation in tweets without the

loss of generality. The filtering criteria applied are stated below:

* Remove Retweets (any tweet which contains the string “RT”)

Remove very short tweets (tweet with length less than 20 characters)

•

Remove non-English tweets (by comparing the words of the tweets with a list

•

of 2,000 common English words, tweets with less than 15% of content matching

threshold are discarded)

Remove similar tweets (by comparing every tweet with every other tweets with more than 90% of content matching with some other tweets are discarded).

•

After this filtering roughly 30% of tweets remain for human labelling on average per sample, which made a total of 10,173 tweets to be labelled.

*# removes pattern in the input text*

def remove\_pattern(input\_txt, pattern):

r = re.findall(pattern, input\_txt) for word in

r: input\_txt = re.sub(word, "", input\_txt) return input\_txt

*# remove twitter handles (@user)*

df[’clean\_tweet’] = np.vectorize(remove\_pattern)(df[’tweet’], "@[\w]\*")

*# remove special characters, numbers and punctuations*

df[’clean\_tweet’] = df[’clean\_tweet’].str.replace("[^a-zA- Z#]", " ")

df.head()

# remove short words

df[’clean\_tweet’] = df[’clean\_tweet’].apply(lambda x: " ".join([w for w in x.split() if len(w)>3]))

df.head()

1. DATA PREPROCESSING
2. *Tokenization*

It is the process of breaking a stream of text up into words, symbols and other meaningful elements called “tokens”. To- kens can be separated by whitespace characters and/or punc- tuation characters. It is done so that we can look at tokens as individual components that make up a tweet. Emoticons and abbreviations (e.g., OMG, WTF, BRB) are identified as part of the tokenization process and

treated as individual tokens.

Tokenization prepares the text for next step, which is to removing stop-words like ‘the’, ‘or’, ‘to’, ‘and’ etc.

# individual words considered as tokens

tokenized\_tweet = df[’clean\_tweet’].apply(lambda x: x.split())

tokenized\_tweet.head()

1. *Normalization*

For the normalization process, the presence of abbreviations within a tweet is noted and then abbreviations are replaced by their actual meaning (e.g., BRB > be right back). We also identify informal intensifiers such as all-caps (e.g., I LOVE this show!!! and character repetitions (e.g., I’ve got a mortgage!! happyyyyyy”), note their presence in the tweet. All-caps words are made into lower case, and instances of repeated characters are replaced by a single character.

−

Finally, the presence of any special Twitter tokens is noted (e.g., #hashtags, usertags, and URLs) and placeholders in- dicating the token type are substituted. Our hope is that this normalization improves the performance of the POS tagger, which is the last preprocessing step.

1. *Parts of speech*

POS-Tagging is the process of assigning a tag to each word in the sentence as to which grammatical part of speech that word belongs to, i.e. noun, verb, adjective, adverb, coordi- nating conjunction etc. For each tweet, we have features for counts of the number of verbs, adverbs, adjectives, nouns, and any other parts of speech.

# stem the wordsfrom nltk.stem.porter

import PorterStemmerstemmer = PorterStemmer() tokenized\_tweet = tokenized\_tweet.apply(lambda sentence:

[stemmer.stem(word) for word in sentence]) tokenized\_tweet.head()

# combine words into single sentence for i in range(len(tokenized\_tweet)):

tokenized\_tweet[i] = " ".join(tokenized\_tweet[i]) df[’clean\_tweet’] = tokenized\_tweet

df.head()

1. ALGORITHM BENCHMARKING

In order to extract sentiment from tweets, sentiment analysis is used. Sentiment analysis is also known as “opinion mining” or “emotion Artificial Intelligence” and alludes to the utilization of natural language processing (NLP), text mining that methodically recognize, extricate, evaluate, and examine emotional states

and subjective information. Sentiment analysis is generally concerned with the voice in client materials; for example, sur- veys and reviews on the Web and web based social networks.

The basic idea of sentiment analysis is to detect the polarity of text documents or short sentences and classify them on this premise. Sentiment polarity is categorized as “positive”, “negative” or “impartial” (neutral). It is important highlight the fact that sentiment mining can be performed on three levels as follows:

1. Document-level sentiment classification: At this level, a document can be classified entirely as “positive”, “negative”, or “neutral”.

2. Sentence-level sentiment classification: At this level, each sentence is classified as “positive”, “negative” or unbiased.

3. Aspect and feature level sentiment classification: At this level, sentences/documents can be categorized as “positive”, “negative” or “nonpartisan” in light of certain aspects of sentences/archives and commonly known as “perspective-level assessment grouping & quote.

1. *Classification of Algorithms*

**Positive**: If the entire tweet has a positive/happy/excited/joyful attitude or if something is mentioned with positive connotations. Also if more than one sentiment is expressed in the tweet but the positive sentiment is more dominant.

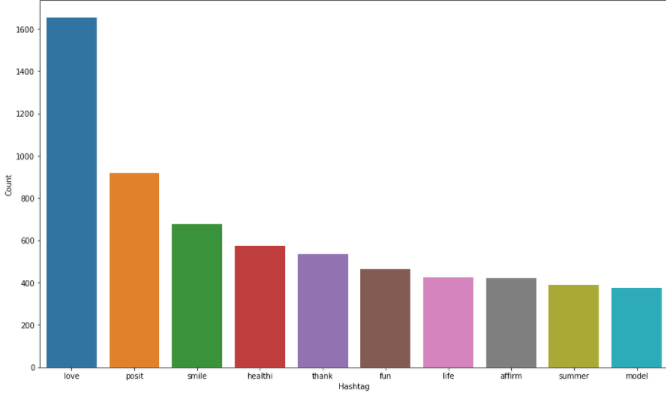
# select top 10 hashtags

d = d.nlargest(columns='Count', n=10)

plt.figure(figsize=(15,9))

sns.barplot(data=d, x='Hashtag', y='Count')

plt.show()



**Negative**: If the entire tweet has a negative/sad/displeased attitude or if something is mentioned with negative connotations. if more than one sentiment is expressed in the tweet but the negative sentiment is more dominant.

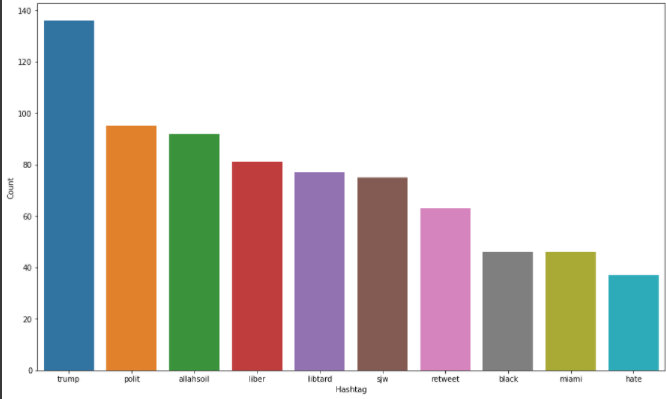
# select top 10 hashtags

d = d.nlargest(columns='Count', n=10)

plt.figure(figsize=(15,9))

sns.barplot(data=d, x='Hashtag', y='Count')

plt.show()



## *Conceptual Model*

It is used to estimate discrete values ( Binary values like 0/1, yes/no, true/false ) based on given set of independent variable(s). In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as logit regression. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).

y=1/(1+e−x)

from sklearn.model\_selection import train\_test\_splitx\_train,

x\_test, y\_train, y\_test = train\_test\_split(bow, df['label'], random\_state=42, test\_size=0.25)

from sklearn.linear\_model import LogisticRegression from sklearn.metrics import f1\_score, accuracy\_score

*# training*

model = LogisticRegression()model.fit(x\_train, y\_train)

# V. Accuracy of the model

Considering a typical set which would include more than 90% benign (0) class. So if the model predicts all the examples as 0 without making any calculation, the accuracy is more than 90%. Coming to our trained model accuracy score above 90% and also we use the to get the output.

# testing

pred = model.predict(x\_test)

f1\_score(y\_test, pred)

#accuracy

accuracy\_score(y\_test,pred)

# use probability to get output

pred\_prob = model.predict\_proba(x\_test)

pred = pred\_prob[:, 1] >= 0.3pred = pred.astype(np.int)

f1\_score(y\_test, pred)

accuracy\_score(y\_test,pred)

# VI . Conclusion

Nowadays became one of the major types of the communication. The large amount of information contained in web-sites makes them an attractive source of data for sentiment analysis and data mining.

In our Research we have used open-source libraries. We used the collected corpus to train a sentiment classifier. Our classifier is able to determine positive, negative and neutral sentiments of documents. The classifier is based on the logistic regression classifier, classifies TRUE if above 30% , else FALSE below 30%.

# VII. Reference

[1] and Johanna Moore International Journal of Computer Science Issues, Vol. 9, Issue 4, No 3, July 2012

[2] S. and D. , Based Sentiment Analysis on Twitter”, Stanford University,2010

[3] M. and Sentiment Analysis on of social media texts:,2014

[4] of Cambridge the Cambridge coding Academy Data Science.2016

[5] and Deepa Gupta Exploring Sentiment Analysis on Twitter Data, IEEE 2015

[6] Brett Duncan and Yanqing Zhang, Neural Networks for Sentiment Analysis on Twitter.2017

[7] Afroze Ibrahim , Twitter Sentiment Analysis: The Good the Bad and the OMG!, Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media.2011.