Social Media Sentiment Analysis Using Twitter Dataset

By

ADITYA RAJ

Registration Number: 12020440

A Project report submitted in fulfillment

of the requirement of the class assignment.

Department of Computer Science and Engineering with a specialization in Data Science

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to both teachers Mr. Subhodeep and Mr. Ajay Sharma of the Machine Learning subject. Mr. Subhodeep has helped me to understand the subject well and grasp all concepts easily. With a deep understanding, I was able to complete this project under his guidance. I would also give a special gratitude to Mr. Ajay Sharma sir. The encouragement he gave in every class helped me to fuel up and complete this project.

ABSTRACT

This project addresses the problem of sentiment analysis in Twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative, or neutral. Twitter is an online micro-blogging and social networking platform which allows users to write short status updates of a maximum length of 140 characters. It is a rapidly expanding service with over 200 million registered users [24] - out of which 100 million are active users and half of them log on to Twitter on a daily basis - generating nearly 250 million tweets per day [20]. Due to this large amount of usage, we hope to achieve a reflection of public sentiment by analysing the sentiments expressed in the tweets. Analysing public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections, and predicting socioeconomic phenomena like the stock exchange. This project aims to develop a model for accurate and automatic sentiment classification of an unknown tweet stream.

INTRODUCTION

Motivation

I have chosen to work with Twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for Twitter, as compared to traditional blogging sites. Moreover, the response on Twitter is more prompt and more general (since the number of users who tweet is substantially more than those who write web blogs daily). Sentiment analysis of the public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analyzing overall public sentiment towards that firm concerning time and using economics tools for finding the correlation between public sentiment and the firm’s stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market are having a favorable response, and in which a negative response (since Twitter allows us to download a stream of geotagged tweets for locations. If firms can get this information they can analyse the reasons behind geographically differentiated responses, and so they can market their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis. One such study was conducted by Tumasjan et al. in Germany for predicting the outcome of federal elections which concluded that Twitter is a good reflection of offline sentiment [4].

Domain Introduction

This project of analysing sentiments of tweets comes under the domain of “Pattern Classification” and “Data Mining”. Both of these terms are very closely related and intertwined, and they can be formally defined as the process of discovering “useful” patterns in large set of data, either automatically (unsupervised) or semiautomatically (supervised). The project would heavily rely on techniques of “Natural Language Processing” in extracting significant patterns and features from the large data set of tweets and on “Machine Learning” techniques for accurately classifying individual unlabelled data samples (tweets) according to whichever pattern model best describes them.

The features that can be used for modelling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word “excellent” has a strong positive connotation while the word “evil” possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment.

Parts of Speech tagging, on the other hand, is a syntactical approach to the problem. It means to automatically identify which part of speech each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb, interjection, etc. Patterns can be extracted from analysing the frequency distribution of these parts of speech (ether individually or collectively with some other part of speech) in a particular class of labelled tweets. Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word lengthening [13], question marks, presence of url in tweets, exclamation marks, internet emoticons and internet shorthand/slangs.

Classification techniques can also be divided into two categories: Supervised vs. unsupervised and non-adaptive vs. adaptive/reinforcement techniques. Supervised approach is when we have pre-labelled data samples available and we use them to train our classifier. Training the classifier means to use the pre-labelled to extract features that best model the patterns and differences between each of the individual classes, and then classifying an unlabelled data sample according to whichever pattern best describes it. For example, if we come up with a highly simplified model that neutral tweets contain 0.3 exclamation marks per tweet on average while sentiment-bearing tweets contain 0.8, and if the tweet we have to classify does contain 1 exclamation mark then (ignoring all other possible features) the tweet would be classified as subjective, since 1 exclamation mark is closer to the model of 0.8 exclamation marks. Unsupervised classification is when we do not have any labelled data for training. In addition to this adaptive classification techniques deal with feedback from the environment. In our case feedback from the environment can be in form of a human telling the classifier whether it has done a good or poor job in classifying a particular tweet and the classifier needs to learn from this feedback. There are two further types of adaptive techniques: Passive and active. Passive techniques are the ones which use the feedback only to learn about the environment (in this case this could mean improving our models for tweets belonging to each of the three classes) but not using this improved learning in our current classification algorithm, while the active approach continuously keeps changing its classification algorithm according to what it learns at real-time.

LITERATURE REVIEW

Limitations of Prior Art

Sentiment analysis of in the domain of micro-blogging is a relatively new research topic so there is still a lot of room for further research in this area. Decent amount of related prior work has been done on sentiment analysis of user reviews, documents, web blogs/articles and general phrase level sentiment analysis. These differ from twitter mainly because of the limit of 140 characters per tweet which forces the user to express opinion compressed in very short text. The best results reached in sentiment classification use supervised learning techniques such as Naive Bayes and Support Vector Machines, but the manual labelling required for the supervised approach is very expensive. Some work has been done on unsupervised (*e.g.,* [11] and [13]) and semi-supervised (*e.g.,* [3] and [10]) approaches, and there is a lot of room of improvement. Various researchers testing new features and classification techniques often just compare their results to base-line performance. There is a need of proper and formal comparisons between these results arrived through different features and classification techniques in order to select the best features and most efficient classification techniques for particular applications.

Related Work

The bag-of-words model is one of the most widely used feature model for almost all text classification tasks due to its simplicity coupled with good performance. The model represents the text to be classified as a bag or collection of individual words with no link or dependence of one word with the other, i.e., it completely disregards grammar and order of words within the text. This model is also very popular in sentiment analysis and has been used by various researchers. The simplest way to incorporate this model in our classifier is by using unigrams as features. Generally speaking n-grams is a contiguous sequence of “n” words in our text, which is completely independent of any other words or grams in the text. So unigrams is just a collection of individual words in the text to be classified, and we assume that the probability of occurrence of one word will not be affected by the presence or absence of any other word in the text. This is a very simplifying assumption but it has been shown to provide rather good performance (for example in [7] and [2]). One simple way to use unigrams as features is to assign them with a certain prior polarity, and take the average of the overall polarity of the text, where the overall polarity of the text could simply be calculated by summing the prior polarities of individual unigrams. Prior polarity of the word would be positive if the word is generally used as an indication of positivity, for example the word “sweet”; while it would be negative if the word is generally associated with negative connotations, for example “evil”. There can also be degrees of polarity in the model, which means how much indicative is that word for that particular class. A word like “awesome” would probably have strong subjective polarity along with positivity, while the word “decent” would although have positive prior polarity but probably with weak subjectivity.

There are three ways of using prior polarity of words as features. The simpler unsupervised approach is to use publicly available online lexicons/dictionaries which map a word to its prior polarity. The Multi-Perspective-Question-Answering (MPQA) is an online resource with such a subjectivity lexicon which maps a total of 4,850 words according to whether they are “positive” or “negative” and whether they have “strong” or “weak” subjectivity [25]. The SentiWordNet 3.0 is another such resource which gives probability of each word belonging to positive, negative and neutral classes [15]. The second approach is to construct a custom prior polarity dictionary from our training data according to the occurrence of each word in each particular class. For example if a certain word is occurring more often in the positive labelled phrases in our training dataset (as compared to other classes) then we can calculate the probability of that word belonging to positive class to be higher than the probability of occurring in any other class. This approach has been shown to give better performance, since the prior polarity of words is more suited and fitted to a particular type of text and is not very general like in the former approach. However, the latter is a supervised approach because the training data has to be labelled in the appropriate classes before it is possible to calculate the relative occurrence of a word in each of the class. Kouloumpis et al. noted a decrease in performance by using the lexicon word features along with custom n-gram word features constructed from the training data, as opposed to when the n-grams were used alone [7].

The third approach is a middle ground between the above two approaches. In this approach we construct our own polarity lexicon but not necessarily from our training data, so we don’t need to have labelled training data. One way of doing this as proposed by Turney et al. is to calculate the prior semantic orientation (polarity) of a word or phrase by calculating it’s mutual information with the word “excellent” and subtracting the result with the mutual information of that word or phrase with the word “poor” [11]. They used the number of result hit counts from online search engines of a relevant query to compute the mutual information. The final formula they used is as follows:

$$\boldsymbol{Polarity}(\boldsymbol{phrase}) = \boldsymbol{log_2} \frac{\boldsymbol{hits}(\boldsymbol{phrase\ NEAR}\ \text{"excellent"}).\ \boldsymbol{hits}(\text{"poor"})}{\boldsymbol{hits}(\boldsymbol{phrase\ NEAR}\ \text{"poor"}).\ \boldsymbol{hits}(\text{"}\boldsymbol{excellent}\text{"})}$$

Where *hits(phrase NEAR “excellent”)* means the number documents returned by the search engine in which the phrase (whose polarity is to be calculated) and word “excellent” are co-occurring. While *hits(“excellent”)* means the number of documents retuned which contain the word “excellent”. Prabowo et al. have gone ahead with this idea and used a seed of 120 positive words and 120 negative to perform the internet searches [12]. So the overall semantic orientation of the word under consideration can be found by calculating the closeness of that word with each one of the seed words and taking and average of it. Another graphical way of calculating polarity of adjectives has been discussed by Hatzivassiloglou et al. [8]. The process involves first identifying all conjunctions of adjectives from the corpus and using a supervised algorithm to mark every pair of adjectives as belonging to the same semantic orientation or different. A graph is constructed in which the nodes are the adjectives and links indicate same or different semantic orientation. Finally, a clustering algorithm is applied which divides the graph into two subsets such that nodes within a subset mainly contain links of same orientation and links between the two subsets mainly contain links of different orientation. One of the subsets would contain positive adjectives and the other would contain negative.

Many of the researchers in this field have used already constructed publicly available lexicons of sentiment bearing words (*e.g.,* [7], [12] and [16]) while many others have also explored building their own prior polarity lexicons (*e.g.,* [3], [10] and [11]).

The basic problem with the approach of prior polarity approach has been identified by Wilson et al. who distinguish between prior polarity and contextual polarity [16]. They say that the prior polarity of a word may in fact be different from the way the word has been used in the particular context. The paper presented the following phrase as an example:

Philip Clapp, president of the National Environment <u>Trust</u>, sums up <u>well</u> the general thrust of the reaction of environmental movements: "There is no <u>reason</u> at all to believe that the polluters are suddenly going to become <u>reasonable</u>."

In this example all of the four underlined words "trust", "well", "reason" and "reasonable" have positive polarities when observed without context to the phrase, but here they are not being used to express a positive sentiment. This concludes that even though generally speaking a word like "trust" may be used in positive sentences, but this doesn't rule out the chances of it appearing in non-positive sentences as well.

Henceforth prior polarities of individual words (whether the words generally carry positive or negative connotations) may alone not enough for the problem. The paper explores some other features which include grammar and syntactical relationships between words to make their classifier better at judging the contextual polarity of the phrase.

The task of twitter sentiment analysis can be most closely related to phrase level sentiment analysis. A seminal paper on phrase level sentiment analysis was presented in 2005 by Wilson et al. [16] which identified a new approach to the problem by first classifying phrases according to subjectivity (polar) and objectivity (neutral) and then further classifying the subjective-classified phrases as either positive or negative. The paper noticed that many of the objective phrases used prior sentiment bearing words in them, which led to poor classification of especially objective phrases. It claims that if we use a simple classifier which assumes that the contextual polarity of the word is merely equal to its prior polarity gives a result of about 48%. The novel classification process proposed by this paper along with the list of ingenious features which include information about contextual polarity resulted in significant improvement in performance (in terms of accuracy) of the classification process. The results from this paper are presented in the table below:

Features	Accuracy	Subjective F.	Objective F.
Word tokens	73.6	55.7	81.2
Words + prior polarity	74.2	60.6	80.7
28 features	75.9	63.6	82.1

Table 2: Step 1 results for Objective / Subjective Classification in [16]

Features	Accuracy	Positive F.	Negative F.	Both F.	Objective F.
Word tokens	61.7	61.2	73.1	14.6	37.7
Word + prior	63.0	61.6	75.5	14.6	40.7
10 features	65.7	65.1	77.2	16.1	46.2

Table 3: Step 2 results for Polarity Classification in [16]

One way of alleviating the condition of independence and including partial context in our word models is to use bigrams and trigrams as well besides unigrams. Bigrams are collection of two contiguous words in a text, and similarly trigrams are collection of three contiguous words. So we could calculate the prior polarity of the bigram / trigram - or the prior probability of that bigram / trigram belonging to a certain class – instead of prior polarity of individual words. Many researchers have experimented with them with the general conclusion that if we have to use one of them alone unigrams perform the best, while unigrams along with bigrams may give better results with certain classifiers [2], [3]. However trigrams usually result in poor performance as reported by Pak et al. [3]. The reduction in performance by using trigrams is because there is a compromise between capturing more intricate patterns and word coverage as one goes to higher-numbered grams. Besides from this some researchers have tried to incorporate negation into the unigram word models. Pang et al. and Pakl et al. used a model in which the prior polarity of the word was reversed if there was a negation (like "not", "no", "don't", etc.) next to that word [5], [3]. In this way some contextual information is included in the word models.

Grammatical features (like "Parts of Speech Tagging" or POS tagging) are also commonly used in this domain. The concept is to tag each word of the tweet in terms of what part of speech it belongs to: noun, pronoun, verb, adjective, adverb, interjections, intensifiers etc. The concept is to detect patterns based on these POS and use them in the classification process. For example it has been reported that objective tweets contain more common nouns and third-person verbs than subjective tweets [3], so if a tweet to be classified has a proportionally large usage of common nouns and verbs in third person, that tweet would have a greater probability of being objective (according to this particular feature). Similarly subjective tweets contain more adverbs, adjectives and interjections [3]. These relationships are demonstrated in the figures below:

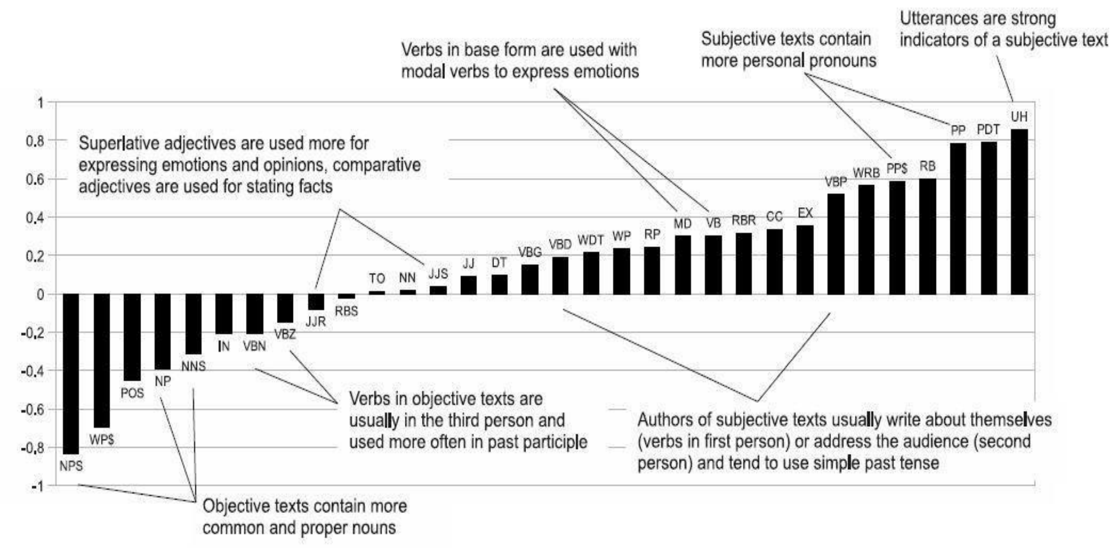

Figure 1: Using POS Tagging as features for objectivity/subjectivity classification

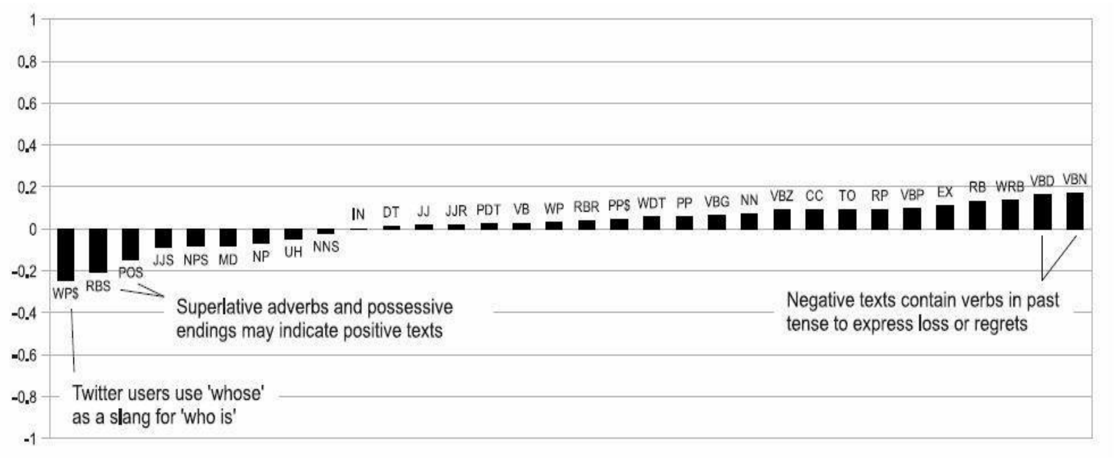

Figure 2: Using POS Tagging as features in positive/negative classification

However there is still conflict whether Parts-of-Speech are a useful feature for sentiment classification or not. Some researchers argue in favour of good POS features (*e.g.*, [10]) while others not recommending them (*e.g.*, [7]).

Besides from these much work has been done in exploring a class of features pertinent only to micro blogging domain. Presence of URL and number of capitalized words/alphabets in a tweet have been explored by Koulompis et al. [7] and Barbosa et al. [10]. Koulmpis also reports positive results for using emoticons and internet slang words as features. Brody et al. does study on word lengthening as a sign of subjectivity in a tweet [13]. The paper reports positive results for their study that the more number of cases a word has of lengthening, the more chance there of that word being a strong indication of subjectivity.

The most commonly used classification techniques are the Naive Bayes Classifier and State Vector Machines. Some researchers like Barbosa et al. publish better results for SVMs [10] while others like Pak et al. support Naive Bayes [3]. (1-9) and (26) also report good results for Maximum Entropy classifier.

It has been observed that having a larger training sample pays off to a certain degree, after which the accuracy of the classifier stays almost constant even if we keep adding more labelled tweets in the training data [10]. Barbosa et al. used tweets labelled by internet resources (*e.g.*, [28]), instead of labelling them by hand, for training the classifier. Although there is loss of accuracy of the labelled samples in doing so (which is modelled as increase in noise) but it has been observed that if the accuracy of training labels is greater than 50%, the more the labels, the higher the accuracy of the resulting classifier. So in this way if there are an extremely large number of tweets, the fact that our labels are noisy and inaccurate can be compensated for [10]. On the other hand Pak et al. and Go et al. [2] use presence of positive or negative emoticons to assign labels to the tweets [3]. Like in the above case they used large number of tweets to reduce effect of noise in their training data.

Some of the earliest work in this field classified text only as positive or negative, assuming that all the data provided is subjective (for example in [2] and [5]). While this is a good assumption for something like movie reviews but when analyzing tweets and blogs there is a lot of objective text we have to consider, so incorporating neutral class into the classification process is now becoming a norm. Some of the work which has included neutral class into their classification process includes [7], [10], [3] and [16].

There has also been very recent research of classifying tweets according to the mood expressed in them, which goes one step further. Bollen et al. explores this area and develops a technique to classify tweets into six distinct moods: tension, depression, anger, vigour, fatigue and confusion [9]. They use an extended version of Profile of Mood States (POMS): a widely accepted psychometric instrument. They generate a word dictionary and assign them weights corresponding to each of the six mood states, and then they represented each tweet as a vector corresponding to these six dimensions. However not much detail has been provided into how they built their customized lexicon and what technique did they use for classification.

FUNCTIONALITY AND DESIGN

We gave the following guidelines to our labellers to help them in the labelling process:

- **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. Example: “*4 more years of being in shithole Australia then I move to the USA! :D*”.

- **Negative**: If the entire tweet has a negative/sad/displeased attitude or if something is mentioned with negative connotations. Also if more than one sentiment is expressed in the tweet but the negative sentiment is more dominant. Example: “*I want an android now this iPhone is boring :S*”.

- **Neutral/Objective**: If the creator of tweet expresses no personal sentiment/opinion in the tweet and merely transmits information. Advertisements of different products would be labelled under this category. Example: “*US House Speaker vows to stop Obama contraceptive rule...*
http://t.co/cyEWqKlE”.

Feature Extraction:

Now that we have arrived at our training set we need to extract useful features from it which can be used in the process of classification. But first we will discuss some text formatting techniques which will aid us in feature extraction:

- Tokenization: It is the process of breaking a stream of text up into words, symbols and other meaningful elements called “tokens”. Tokens can be separated by whitespace characters and/or punctuation characters. It is done so that we can look at tokens as individual components that make up a tweet [19].
- Url’s and user references (identified by tokens “http” and “@”) are removed if we are interested in only analysing the text of the tweet.
- Punctuation marks and digits/numerals may be removed if for example we wish to compare the tweet to a list of English words.
- Lowercase Conversion: Tweet may be normalized by converting it to lowercase which makes it’s comparison with an English dictionary easier.
- Stemming: It is the text normalizing process of reducing a derived word to its root or stem [28]. For example a stemmer would reduce the phrases “stemmer”,

"stemmed", "stemming" to the root word "stem". Advantage of stemming is that it makes comparison between words simpler, as we do not need to deal with complex grammatical transformations of the word. In our case we employed the algorithm of "porter stemming" on both the tweets and the dictionary, whenever there was a need of comparison.

- Stop-words removal: Stop words are class of some extremely common words which hold no additional information when used in a text and are thus claimed to be useless [19]. Examples include "a", "an", "the", "he", "she", "by", "on", etc. It is sometimes convenient to remove these words because they hold no additional information since they are used almost equally in all classes of text, for example when computing prior-sentiment-polarity of words in a tweet according to their frequency of occurrence in different classes and using this polarity to calculate the average sentiment of the tweet over the set of words used in that tweet.
- Parts-of-Speech Tagging: 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, coordinating conjunction etc.

Now that we have discussed some of the text formatting techniques employed by us, we will move to the list of features that we have explored. As we will see below a feature is any variable which can help our classifier in differentiating between the different classes. There are two kinds of classification in our system (as will be discussed in detail in the next section), the objectivity / subjectivity classification and the positivity / negativity classification. As the name suggests the former is for differentiating between objective and subjective classes while the latter is for differentiating between positive and negative classes.

The list of features explored for objective / subjective classification is as below:

- Number of exclamation marks in a tweet
- Number of question marks in a tweet
- Presence of exclamation marks in a tweet

- Presence of question marks in a tweet
- Presence of url in a tweet
- Presence of emoticons in a tweet
- Unigram word models calculated using Naive Bayes
- Prior polarity of words through online lexicon MPQA
- Number of digits in a tweet
- Number of capitalized words in a tweet
- Number of capitalized characters in a tweet
- Number of punctuation marks / symbols in a tweet
- Ratio of non-dictionary words to the total number of words in the tweet
- Length of the tweet
- Number of adjectives in a tweet
- Number of comparative adjectives in a tweet
- Number of superlative adjectives in a tweet
- Number of base-form verbs in a tweet
- Number of past tense verbs in a tweet
- Number of present participle verbs in a tweet
- Number of past participle verbs in a tweet
- Number of 3rd person singular present verbs in a tweet
- Number of non-3rd person singular present verbs in a tweet
- Number of adverbs in a tweet
- Number of personal pronouns in a tweet
- Number of possessive pronouns in a tweet
- Number of singular proper noun in a tweet
- Number of plural proper noun in a tweet
- Number of cardinal numbers in a tweet
- Number of possessive endings in a tweet
- Number of wh-pronouns in a tweet
- Number of adjectives of all forms in a tweet
- Number of verbs of all forms in a tweet
- Number of nouns of all forms in a tweet
- Number of pronouns of all forms in a tweet

The list of features explored for positive / negative classification are given below:

- Overall emoticon score (where 1 is added to the score in case of positive emoticon, and 1 is subtracted in case of negative emoticon)
- Overall score from online polarity lexicon MPQA (where presence of strong positive word in the tweet increases the score by 1.0 and the presence of weak negative word would decrease the score by 0.5)
- Unigram word models calculated using Naive Bayes
- Number of total emoticons in the tweet
- Number of positive emoticons in a tweet
- Number of negative emoticons in a tweet
- Number of positive words from MPQA lexicon in tweet
- Number of negative words from MPQA lexicon in tweet
- Number of base-form verbs in a tweet
- Number of past tense verbs in a tweet
- Number of present participle verbs in a tweet
- Number of past participle verbs in a tweet
- Number of 3rd person singular present verbs in a tweet
- Number of non-3rd person singular present verbs in a tweet
- Number of plural nouns in a tweet
- Number of singular proper nouns in a tweet
- Number of cardinal numbers in a tweet
- Number of prepositions or coordinating conjunctions in a tweet
- Number of adverbs in a tweet
- Number of wh-adverbs in a tweet
- Number of verbs of all forms in a tweet

IMPLEMENTATION AND CONCLUSION

Data Collection: The first step involved in building model is to collect data. There is various method to do so. But for your analysis I have downloaded a dataset from Kaggle, which has almost 1,65000 tweets labelled with -1, 0, 1 for negative, neutral, and positive tweets respectively.

Data Processing: Once we have collected the data, the next step is to pre-process it. Data pre-processing involves cleaning the data, removing noise, and transforming the data into a format that can be easily used for analysis. This step includes removing UTLs, special characters, stop words and stemming or lemmatization.

Feature Extraction: After pre-processing the data, we need to extract features from it. Features extraction involves identifying relevant features from the pre-processed data. There are different methods for feature extraction, including bag-of-words, TF-IDF, and word embeddings. We can use these methods to represent the data in a format that can be used for analysis.

Model Selection: The next step is to select a suitable model for sentiment analysis. There are different types of models, including rule-based models, machine learning models, and deep learning models. We can choose a model based on the complexity of the problem and the size of the dataset. Some popular machine learning models for sentiment analysis include Naïve bays, Logistic Regression, and support vector machines.

Model Evaluation: After building the model, we need to evaluate our model performance. Model evaluation involves testing the models on a set of data that it has not seen before. We can use metrices such as accuracy, precision, recall, and F1 score to evaluate the model's performance.

Conclusion: Social media sentiment analysis is a powerful tool that can be used to extract valuable insights from social media date. In this report, we discussed how we can make model for social media sentiment analysis using a Twitter dataset. The process involves data collection, data pre-processing, feature extraction, model selection and model evaluation. By following there steps we can build an accurate model that can be used to analyse social media data and extract insights from it.

In [1]:
```
# importing necessary lobraries
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
nltk.download('stopwords')
nltk.download('punkt')

from tkinter import *
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\aditya\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to [nltk_data]
C:\Users\aditya\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [2]:
```
# Load the dataset into a pandas dataframe
df = pd.read_csv('Twitter_Data.csv')
```

Performing EDA to understand dataframe

In [3]:
```
df.head()
```

Out[3]:

tweets	category

```
0       when modi promised “minimum government      -1.0
maximum...
1     talk all the nonsense and continue all the dra...    0.0
2   what did just say vote for modi welcome bjp t...     1.0
3   asking his supporters prefix chowkidar their n...     1.0
4        answer who among these the most powerful
world...                                               1.0
```

In [4]:
```
# Information of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162980 entries, 0 to 162979 Data
columns (total 2 columns):
    #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
0       tweets    162976 non-null  object
1       category     162973   non-null     float64   dtypes:
float64(1), object(1) memory usage: 2.5+ MB
```

In [5]:
```
# Count of each Category
df['category'].value_counts()
```

Out[5]:
```
1.0     72250
 0.0    55213 -1.0
35510
Name: category, dtype: int64
```

Preprocess the data

[15]:
```
stop_words = set(stopwords.words('english')) # set of stop words in English def
preprocess_tweet(tweet):
    if isinstance(tweet, str): # check if the tweet is a string
```

In

```
        tweet = re.sub(r"http\S+|www\S+|https\S+", '', tweet, flags=re.MULTILINE) # remove URLs         tweet =
re.sub(r'\@\w+|\#', '', tweet) # remove mentions and hashtags         tweet = re.sub(r'\d+', '', tweet) # remove digits
tweet = tweet.lower() # convert to lowercase
        tokens = word_tokenize(tweet) # tokenize the tweet into words
        tokens = [word for word in tokens if not word in stop_words] # remove stop words
preprocessed_tweet = ' '.join(tokens) # join the remaining words back into a sentence         return
preprocessed_tweet     else:         return ''
```

In [16]:

```
# Applying the preprocessing function to each tweet
df['tweets'] = df['tweets'].apply(preprocess_tweet)
```

In [17]:

```
df['tweets'] = df['tweets'].replace('', np.nan) # replace empty strings with NaN
df = df.dropna() # drop rows with NaN values
```

In [18]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['tweets'], df['category'], test_size=0.2, random_state=42)
```

In [19]:

```
# Vectorize the tweets using TF-IDF technique
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

In [11]:

```
# Train a machine learning model on the training data
# Here, we will use Logistic Regression as our classifier
clf = LogisticRegression(random_state=42, max_iter=1000)
clf.fit(X_train, y_train)
```

Out[11]:

```
LogisticRegression(max_iter=1000, random_state=42)
```

In [12]:

```
# Evaluate the model's performance on the testing data
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
report = classification_report(y_test, y_pred)
print(f"Classification Report:\n{report}")
```

```
Accuracy: 0.8942295887047268
Classification Report:
           precision    recall  f1-score   support

      -1.0       0.89      0.78      0.83      7026
          0.0       0.87      0.97      0.91     10988       1.0       0.92
                                                   0.90      0.91     14566

    accuracy                           0.89     32580   macro avg       0.89
  0.88       0.88     32580 weighted avg       0.90      0.89      0.89
                                                                      32580
```

In [21]:
```
# Create a GUI
def predict_sentiment():
    tweet = tweet_input.get()
    preprocessed_tweet = preprocess_tweet(tweet)
    tweet_vector = vectorizer.transform([preprocessed_tweet])
    sentiment = clf.predict(tweet_vector)[0]
    if sentiment == -1:
        sentiment_text.set("Negative")
    elif sentiment == 0:
        sentiment_text.set("Neutral")
    else:
        sentiment_text.set("Positive")

root = Tk()
root.title("Twitter Sentiment Analysis")
root.geometry("400x200")

tweet_label = Label(root, text="Enter a tweet:")
tweet_label.pack()

tweet_input = Entry(root)
tweet_input.pack()

predict_button = Button(root, text="Predict", command=predict_sentiment)
predict_button.pack()

sentiment_label = Label(root, text="Sentiment:")
sentiment_label.pack()

sentiment_text = StringVar()
sentiment_output = Label(root, textvariable=sentiment_text)
sentiment_output.pack()

root.mainloop()
```

In [ ]: