CHAPTER-1

MOTIVATION AND LITERATURE REVIEW

1.1 MOTIVATION

Twitter is a popular micro blogging service where users create status messages (called "tweets"). These tweets sometimes express opinions about different topics. Using this analyzer,

- Consumers can use sentiment analysis to research products or services before making a purchase. E.g. Kindle
- Marketers can use this to research public opinion of their company and products, or to analyse customer satisfaction. E.g. Election Polls
- Organizations can also use this to gather critical feedback about problems in newly released products. E.g. Brand Management (Nike, Adidas)

1.2 LITERATURE SURVEY

1.2.1 Limitations of prior work

Sentiment analysis is a relatively new research topic so there is still a lot of scope of 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 becomes very expensive. Some work has been done on unsupervised and semi-supervised approaches, and there is a lot of scope for 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 achieved through different features and classification techniques in order to select the best features and most efficient classification techniques for particular applications.

1.2.2 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 and its good performance. The model represents the text to be classified as a collection of individual words with no link or dependency of one word with another, i.e. it completely disregards grammar and order of words within the text. This model is also very popular for 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 it is assumed 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. 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 “better”; while it would be negative if the word is generally associated with negative connotations, for example “sad”. 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. The Senti Word Net 3.0 is another such resource which gives probability of each word belonging to positive, negative and neutral classes.

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 the probability of that word belonging to positive class can be calculated 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.

The third approach is a middle ground between the above two approaches. In this approach the polarity lexicon is being constructed but not necessarily from the training data, so there is no need of having labelled training data. One way of doing this as proposed by Turney 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". They used the number of result hit counts from online search engines of a relevant query to compute the mutual information.

Another graphical way of calculating polarity of adjectives has been discussed by Hatzivassiloglou. 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. The basic problem with the approach of prior polarity approach has been identified by Wilson who distinguished between prior polarity and contextual polarity. 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 Trust, sums up well the general thrust of the reaction of environmental movements: "There is no reason at all to believe that the polluters are suddenly going to become reasonable."

In this example all of the four highlighted 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 negative sentences as well. Henceforth prior polarities of individual words (whether the words generally carry positive or negative sentiment) may alone not enough for the problem. A seminal paper on phrase level sentiment analysis was presented in 2005 by Wilson 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 simple classifier is being used 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.

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 the prior polarity of the bigram / trigram -or the prior probability of that bigram / trigram belonging to a certain class can be calculated instead of prior polarity of individual words. Many researchers have experimented with them with the general conclusion that if used one of them alone unigrams perform the best, while unigrams along with bigrams may give better results with certain classifiers .However trigrams usually result in poor performance. 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.

CHAPTER- 2

INTRODUCTION

2.1 OVERVIEW

This project of sentiment analysis of tweets comes under the domain of "Pattern Classification" and "Data Mining". Both of these terms are very closely related , and they can be formally defined as the process of identifying "useful" patterns from large set of data, either automatically (unsupervised) or semi-automatically (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 unlabeled data samples (tweets) according to whichever pattern model best describes them.

The features that can be used for modeling 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. Prior sentiment polarity means that some words and phrases have a natural tendency for expressing particular and specific sentiments in general. For example the word "better" has a positive sentiment while the word "sad" possesses a negative sentiment. So whenever a word with positive sentiment is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment.

Patterns can be extracted from analyzing the frequency distribution of these parts of speech (ether individually or collectively with some other part of speech) in a particular class of labeled 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.

2.1.1 What is real time analysis?

Real-time analytics is the use of all available enterprise data and resources, when they are needed. It consists of dynamic analysis and reporting, based on the data entered into a system, it takes less than one minute before the actual time of use. Real-time analytics is also known as real-time data analytics, real-time data integration, and real-time intelligence.

2.1.2 Significance of real time analysis

The need for real-time analytics has been growing with time. Its importance in various domains has proved that the application brings quicker solutions. Whether it is banking, retail or telecommunication, real-time analytics has its way around.

In banking, it’s been heard and experienced various types of frauds. Fraud transactions, are one of them occurring on a daily basis. For example, the credit card may have had transactions, twice in two different parts of the country. Real-time analytics enables to detect the location and longitude. If the locations of both the transactions do not match, then there is definitely a grave issue.

Another simple example is the social networking sites. Twitter users would be aware of the trending topics in the twitter page. Here, real time analytics comes in the picture, since it thrives on the user data. Based on a user’s tweets, they source the most trending and talked about topics, and post it on the page about what’s trending. This immediately drives revenue and traffic.

2.1.3 Usefulness

Sentiment analysis is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics.

- The applications of sentiment analysis are broad and powerful. The ability to extract insights from social data is a practice that is being widely adopted by organizations across the world.
- Shifts in sentiment on social media have been shown to correlate with shifts in the stock market.
- The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election.
- The ability to quickly understand consumer attitudes and react accordingly is something that Expedia Canada took advantage of when they noticed that there was a steady increase in negative feedback to the music used in one of their television

adverts.

- Sentiment analysis conducted by the brand revealed that the music played on the commercial had become incredibly irritating after multiple airings, and consumers were flocking to social media to vent their frustrations.

2.2 FRAMEWORK AND LIBRARIES USED

2.2.1 NLTK Library:

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

2.2.2 TwitterSearch Library:

This library allows the user to easily create a search through the Twitter API without having to know too much about the API details. Based on such a search the user can even iterate throughout all tweets reachable via the Twitter Search API. There is an automatic reload of the next pages while using the iteration. TwitterSearch was developed as part of an interdisciplinary project at the Technische Universitat Munchen. More than that, TwitterSearch is:

- Small (around 500 lines of code currently)

- Easy to use, even for beginners
- Good at giving all available information (including meta information)
- Without any need to manually reload more results from the API
- Wrong values of API arguments are to raise an exception. This is done before the API gets queried and therefore helps to avoid to reach Twitters' limitations by obviously wrong API calls
- Friendly to Python >= 2.7 **and** Python >= 3.2
- Pretty to look at :)

2.2.3 Regex Library:

A regular expression is a special sequence of characters that helps the user to match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are widely used in UNIX world.

The module **‘re’** provides full support for Perl-like regular expressions in Python. The **‘re’** module raises the exception regular expression error if an error occurs while compiling or using a regular expression.

There are various characters, which would have special meaning when they are used in regular expression. To avoid any confusion while dealing with regular expressions, the user should use Raw Strings as regular expression.

2.2.4 Django Framework:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so the user can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django was designed to help developers take applications from concept to completion as quickly as possible.

Django takes security seriously and helps developers avoid many common security mistakes.

Some of the busiest sites on the Web leverage Django’s ability to quickly and flexibly scale.

2.2.5 Twitter Bootstrap:

Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

Bootstrap is the second most-starred project on GitHub, with more than 107,000 stars and 48,000 forks.

CHAPTER- 3

PROJECT OBJECTIVE/FEATURES

3.1 OBJECTIVES

- To implement an algorithm for automatic classification of text into positive, negative or neutral.
- Sentiment Analysis to determine the attitude of the mass is positive, negative or neutral towards the subject of interest.
- Graphical representation of sentiment in the form of pie-chart.
- Sentiment analysis of Persons, Movies, Mobiles and Places.

3.2 SYSTEM FEATURES

3.2.1 Hardware Requirements

- Dual/quad core CPU, 2.5GHz.
- 4 GB RAM.
- 10 Gigabit Ethernet
- Turbo speed- 3.7 GHz.
- Cache Memory- 6 MB

3.2.2 Software requirements

- System Software
    - Operating System - Windows
- Application Software
    - Web Browsers - Chrome, Opera, Edge
    - Programming languages - Python.

- Framework
    - Django
    - Bootstrap

3.2.3 Optimal data for training the classifier

Huge amount of data available online for training the classifier. But data to train our classifier should be in English language and labelled as positive, negative or neutral.

3.2.4 Custom search

When a user wants to do sentiment analysis of any topic then user will simply put that keyword in search space of the website and result of sentiment of that keyword will be displayed on the frontend in the form of pie-chart.

3.2.5 View previous analysis

User can also see sentiment analysis on the topic that were previously analyzed. For example sentiment analysis of people like Narendra Modi, Arvind Kejriwal, Rahul Ghandhi is already done.Similarly for recent movies like Bahubali, Half girlfriend etc. is also done.

3.2.6 Responsive

The website is responsive and mobile first i.e. user can view the same on desktop, laptop, tablets, mobiles and other small computing devices without compromising on user experience.

CHAPTER- 4

IMPLEMENTATION

4.1 OVERVIEW

The process of designing a functional classifier for sentiment analysis can be broken down into five basic categories. They are as follows:

1. Training Data Set
2. Data Gathering
3. Pre processing
4. Feature Extraction
5. Classification

4.2 ANALYSIS MODELS

4.2.1 Use Case

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Use case diagrams are in fact two fold - they are both behaviour diagrams, because they describe behaviour of the system, and they are also structure diagrams - as a special case of class diagrams where classifiers are restricted to be either actors or use cases related to each other with associations.

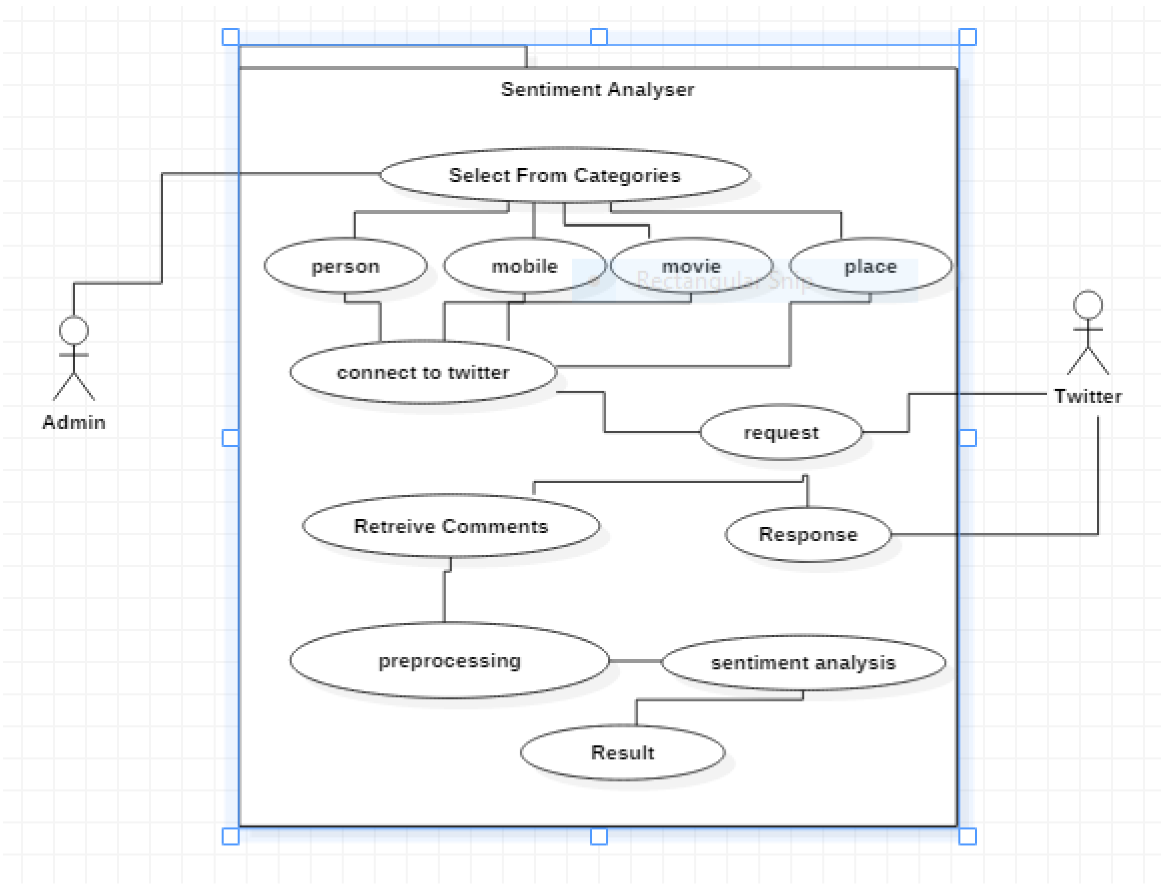


Figure 1. Use case

4.2.2 Class Diagram

Class diagram is UML structure diagram which shows structure of the designed system at the level of classes and interfaces, shows their features, constraints and relationships - associations, generalizations, dependencies, etc.

Some common types of class diagrams are:

- Domain model diagram,
- Diagram of implementation classes.

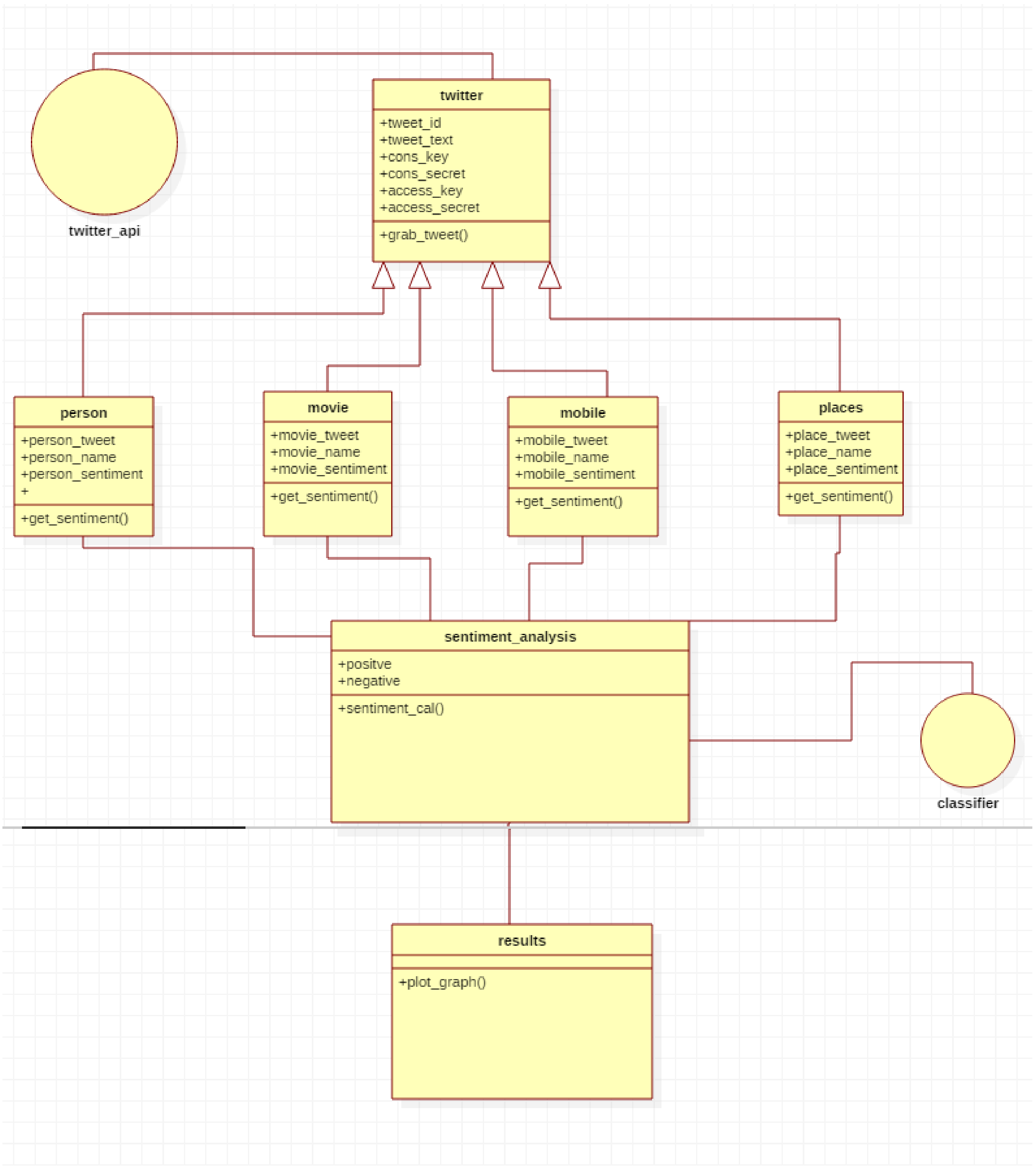


Figure 2. Class Diagram

4.2.3 Swimlane Diagram

Activity diagram is UML behaviour diagram which shows flow of control or object flow with emphasis on the sequence and conditions of the flow. The actions coordinated by activity models can be initiated because other actions finish executing, because objects and data become available, or because some events external to the flow occur. The following nodes and edges are typically drawn on UML activity diagrams: activity, partition, action, object, control, activity edge.

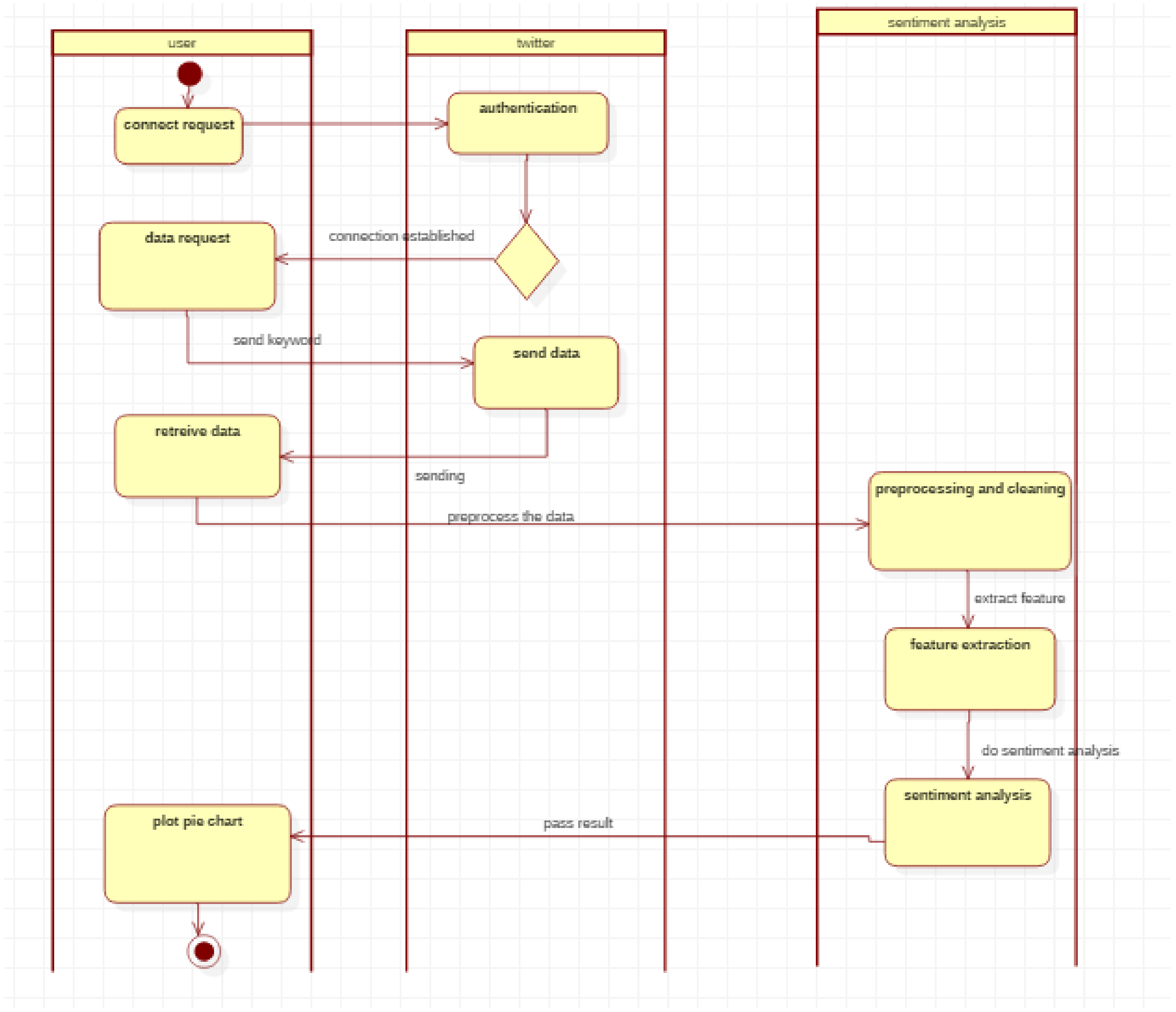
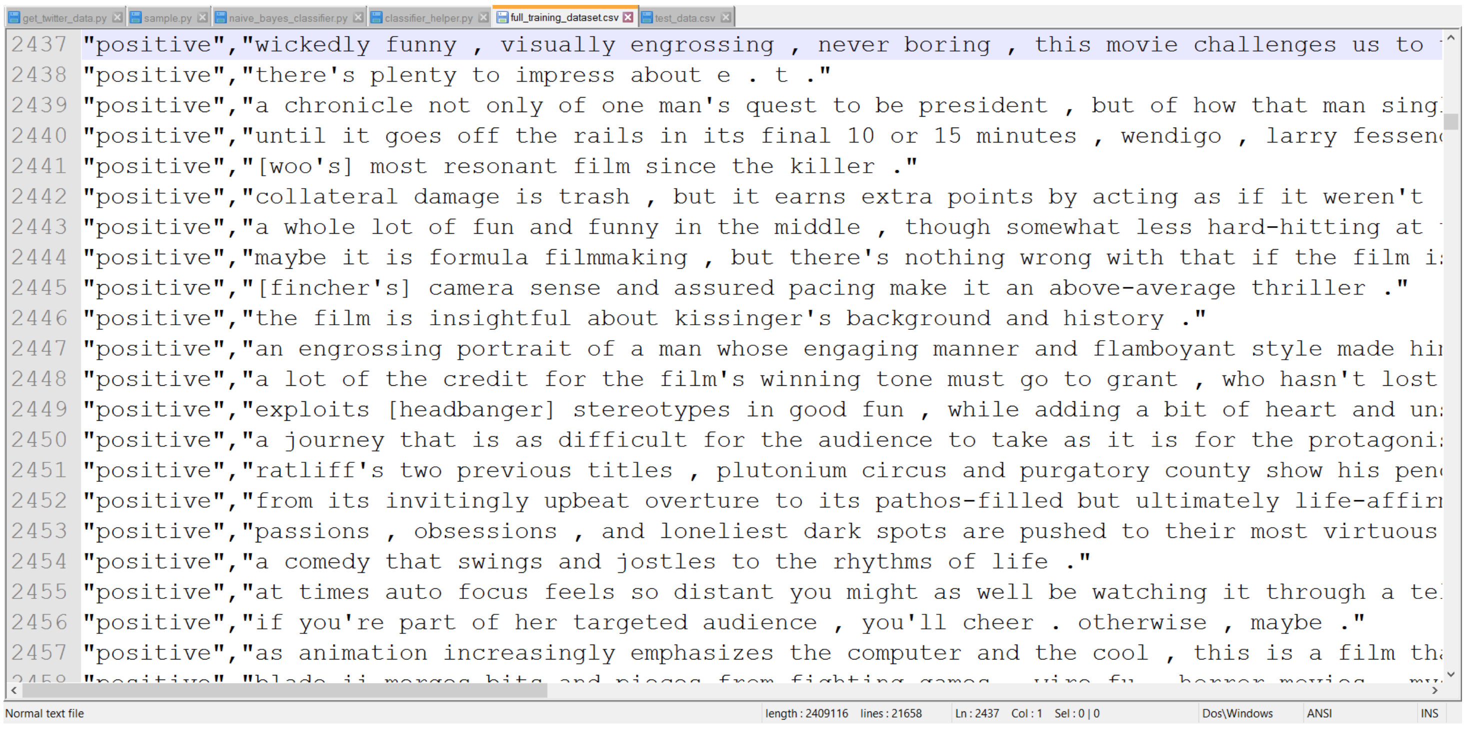


Figure 3. Swimlane Diagram

Activity is a parameterized behaviour represented as coordinated flow of actions. The flow of execution is modelled as activity nodes connected by activity edges. A node can be the execution of a subordinate behaviour, such as an arithmetic computation, a call to an operation, or manipulation of object contents. Activity nodes also include flow of control obstructs, such as synchronization, decision, and concurrency control. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions. In an object-oriented model, activities are usually invoked indirectly as methods bound to operations that are directly invoked.

4.3 TRAINING DATA SET

For training the classifier 8521 tweets were downloaded from www.sentiment140.com which were manually labelled as positive, negative or neutral.


get_twitter_data.py
sample.py
naive_bayes_classifier.py
classifier_helper.py
full_training_dataset.csv
test_data.csv
Normal text file
length : 2409116 lines : 21658
Ln : 2437 Col : 1 Sel : 0 | 0
Dos\Windows
ANSI
INS

```
"positive","wickedly funny , visually engrossing , never boring , this movie challenges us to
"positive","there's plenty to impress about e . t ."
"positive","a chronicle not only of one man's quest to be president , but of how that man sing
"positive","until it goes off the rails in its final 10 or 15 minutes , wendigo , larry fessen
"positive","[woo's] most resonant film since the killer ."
"positive","collateral damage is trash , but it earns extra points by acting as if it weren't
"positive","a whole lot of fun and funny in the middle , though somewhat less hard-hitting at
"positive","maybe it is formula filmmaking , but there's nothing wrong with that if the film i
"positive","[fincher's] camera sense and assured pacing make it an above-average thriller ."
"positive","the film is insightful about kissinger's background and history ."
"positive","an engrossing portrait of a man whose engaging manner and flamboyant style made hi
"positive","a lot of the credit for the film's winning tone must go to grant , who hasn't lost
"positive","exploits [headbanger] stereotypes in good fun , while adding a bit of heart and un
"positive","a journey that is as difficult for the audience to take as it is for the protagoni
"positive","ratliff's two previous titles , plutonium circus and purgatory county show his pen
"positive","from its invitingly upbeat overture to its pathos-filled but ultimately life-affir
"positive","passions , obsessions , and loneliest dark spots are pushed to their most virtuous
"positive","a comedy that swings and jostles to the rhythms of life ."
"positive","at times auto focus feels so distant you might as well be watching it through a te
"positive","if you're part of her targeted audience , you'll cheer . otherwise , maybe ."
"positive","as animation increasingly emphasizes the computer and the cool , this is a film th
"positive","blade ii merges bits and pieces from fighting games , wire fu , horror movies , my
```

Figure 4. Training Set

4.4 DATA GATHERING

Data in the form of raw tweets is acquired from twitter using “twittersearch” python library which provides data for last week tweets.

Whenever a user wants to do sentiment analysis on a topic then user enters a keyword about which user wants to do sentiment analysis and at our backend with the help of twittersearch library we gathered tweets of last week for that topic. A tweet acquired by this method has a lot of raw information in it which we may or may not find useful for our particular application. It comes in the form of the python “dictionary” data type with various key-value pairs. A list of some key-value pairs are given below:

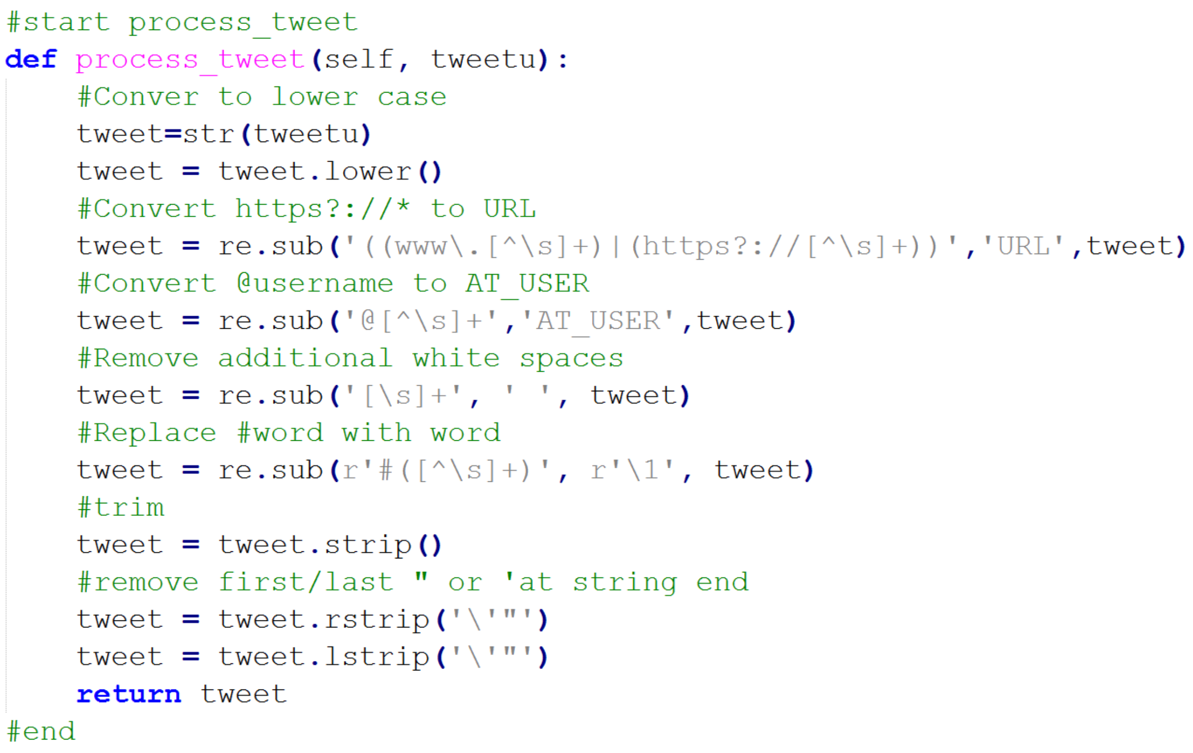
- User ID
- Screen name of the user
- Original Text of the tweet
- Presence of hashtags
- Whether it is a re-tweet
- Language under which the twitter user has registered their account
- Geo-tag location of the tweet
- Date and time when the tweet was created

4.5 PRE PROCESSING

Since this is a lot of information available, only necessary information was collected and discard the rest. Only the text part of the tweet is stored in a separate file given that language is specified to be English. The original text content of the tweet is given under the dictionary key “text” and the language of user’s account is given under “lang”. Since human labelling is an expensive process therefore further filter out the tweets to be labelled and then greatest amount of variation in tweets without the loss of generality is available.

The filtering criteria applied are stated below:

1. Lower Case - Convert the tweets to lower case.
2. URLs - Don't intend to follow the short urls and determine the content of the site, so all these URLs were replaces by word URL via regular expression.
3. @username - Eliminate "@username" via regex matching or replace it with generic word AT_USER.
4. #hashtag - hash tags can give us some useful information, so it is useful to replace them with the exact same word without the hash. E.g. #nike replaced with 'nike'.
5. Punctuations and additional white spaces - remove punctuation at the start and ending of the tweets. E.g: ' the day is beautiful! replaced with 'the day is beautiful'. It is also helpful to replace multiple whitespaces with a single whitespace



```
#start process_tweet
def process_tweet(self, tweetu):
    #Conver to lower case
    tweet=str(tweetu)
    tweet = tweet.lower()
    #Convert https?://* to URL
    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
    #Convert @username to AT_USER
    tweet = re.sub('@[^\s]+','AT_USER',tweet)
    #Remove additional white spaces
    tweet = re.sub('[\s]+', ' ', tweet)
    #Replace #word with word
    tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
    #trim
    tweet = tweet.strip()
    #remove first/last " or 'at string end
    tweet = tweet.rstrip('\'"')
    tweet = tweet.lstrip('\'"')
    return tweet
#end
```

Figure 5. Pre-Processing

4.6 FEATURE EXTRACTION

After pre-processing the training set, features need to be extracted from it which can be used in the process of classification. There are some formatting techniques which will aid 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 to look tokens as individual components that make up a tweet.

•URLs and user references (identified by tokens “http” and “@”) are removed because they don’t contribute in analysing the text of the tweet.

•Punctuation marks and digits/numerals should be removed because they have to be compared with list of English words in the tweets.

•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. 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 there is no need to deal with complex grammatical transformations of the word. In this project the algorithm of “porter stemming” is employed on both the tweets and the dictionary, whenever there was a need of comparison.

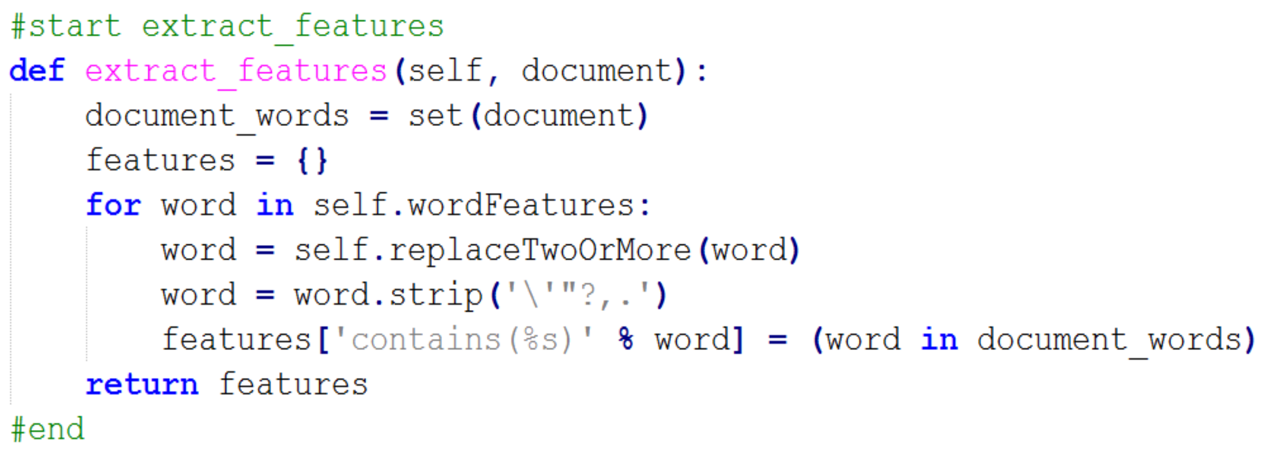
•**Stop-words removal**: Stop words are class of some extremely common words which hold no additional information when used in a text and are thus claimed to be useless. Examples include “a”, “an”, “the”, “he”, “she”, “by”, “on”, etc. It is sometimes convenient to remove these words because they hold 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.

After discussing some of the text formatting techniques employed in this project, features extracted need to be considered. Feature is any variable which can help the classifier in differentiating between different classes.

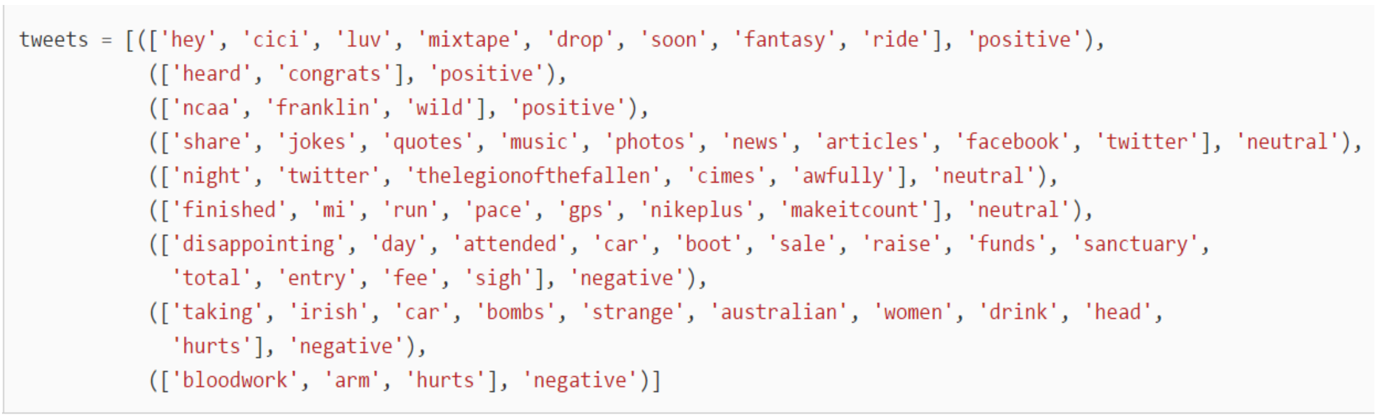
After pre-processing the training set data which consists of 3125 positive tweets, 3125 negative tweets and 2271 neutral tweets, feature vector is computed as below:

Unigrams At the end of pre-processing there are 24312 features extracted which are unigrams and each of the features have equal weights.



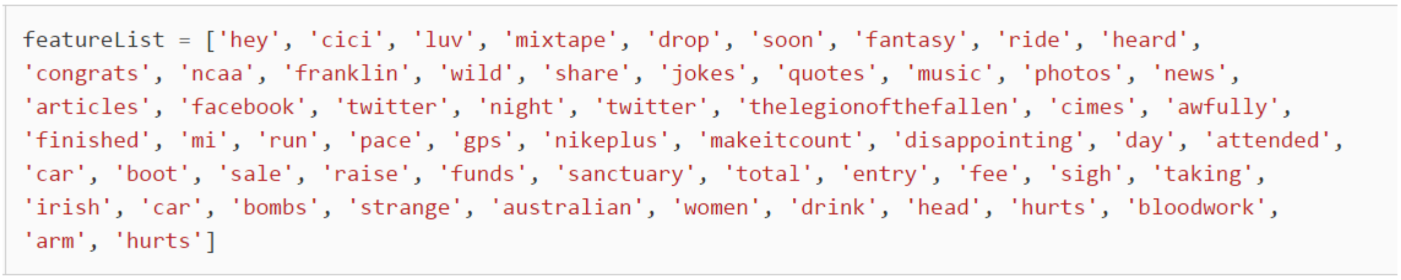
```
#start extract_features
def extract_features(self, document):
    document_words = set(document)
    features = {}
    for word in self.wordFeatures:
        word = self.replaceTwoOrMore(word)
        word = word.strip('\'"?,.')
        features['contains(%s)' % word] = (word in document_words)
    return features
#end
```

Figure 6.1 Feature Extraction



```
tweets = [(['hey', 'cici', 'luv', 'mixtape', 'drop', 'soon', 'fantasy', 'ride'], 'positive'),
          (['heard', 'congrats'], 'positive'),
          (['ncaa', 'franklin', 'wild'], 'positive'),
          (['share', 'jokes', 'quotes', 'music', 'photos', 'news', 'articles', 'facebook', 'twitter'], 'neutral'),
          (['night', 'twitter', 'thelegionofthefallen', 'cimes', 'awfully'], 'neutral'),
          (['finished', 'mi', 'run', 'pace', 'gps', 'nikeplus', 'makeitcount'], 'neutral'),
          (['disappointing', 'day', 'attended', 'car', 'boot', 'sale', 'raise', 'funds', 'sanctuary',
            'total', 'entry', 'fee', 'sigh'], 'negative'),
          (['taking', 'irish', 'car', 'bombs', 'strange', 'australian', 'women', 'drink', 'head',
            'hurts'], 'negative'),
          (['bloodwork', 'arm', 'hurts'], 'negative')]
```

Figure 6.2. Tokenized Tweets



```
featureList = ['hey', 'cici', 'luv', 'mixtape', 'drop', 'soon', 'fantasy', 'ride', 'heard',
'congrats', 'ncaa', 'franklin', 'wild', 'share', 'jokes', 'quotes', 'music', 'photos', 'news',
'articles', 'facebook', 'twitter', 'night', 'twitter', 'thelegionofthefallen', 'cimes', 'awfully',
'finished', 'mi', 'run', 'pace', 'gps', 'nikeplus', 'makeitcount', 'disappointing', 'day', 'attended',
'car', 'boot', 'sale', 'raise', 'funds', 'sanctuary', 'total', 'entry', 'fee', 'sigh', 'taking',
'irish', 'car', 'bombs', 'strange', 'australian', 'women', 'drink', 'head', 'hurts', 'bloodwork',
'arm', 'hurts']
```

Figure 6.3 Feature list

4.7 CLASSIFICATION

Pattern classification is the process through which data is divided into different classes according to some common patterns which are found in one class which differ to some degree with the patterns found in the other classes. The ultimate aim of the project is to design a classifier which accurately classifies tweets in the following four sentiment classes: positive, negative, neutral and ambiguous.

There can be two kinds of sentiment classifications in this area: contextual sentiment analysis and general sentiment analysis. Contextual sentiment analysis deals with classifying specific parts of a tweet according to the context provided, for example for the tweet "4 more years of being in shithole Australia then I move to the USA " a contextual sentiment classifier would identify Australia with negative sentiment and USA with a positive sentiment. On the other hand general sentiment analysis deals with the general sentiment of the entire text (tweet in this case) as a whole. Thus for the tweet mentioned earlier since there is an overall positive attitude, an accurate general sentiment classifier would identify it as positive. In this particular project latter case is only considered i.e. of general (overall) sentiment analysis of the tweet as a whole.

The classification approach generally perform Polarity Classification to determine whether the tweet is positive, negative or neutral.

4.7.1 Naive Bayes Classification

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

4.7.2 Maximum Entropy Classification

The motivating idea behind maximum entropy is that one should prefer the most uniform models that also satisfy any given constraints. For example, consider a four-way text classification task where on average 40% of documents with the word "professor" in them are in the faculty class. Intuitively, when given a document with "professor" in it can be said that it has a 40% chance of being a faculty document, and a 20% chance for each of the other three classes. If a document does not have "professor" it could be guessed that there is uniform distribution of class, 25% each. This model is exactly the maximum entropy model that conforms to the known constraint. Calculating the model is easy in this example, but when there are many constraints to satisfy, rigorous techniques are needed to find the optimal solution. In its most general formulation, maximum entropy can be used to estimate any probability distribution. Only classification is being used in this project, thus the further discussion for learning conditional distribution from labelled training data is being limited here.

4.8 ACCURACY OF THE ALGORITHMS

In this project two classification algorithms i.e. Naive Bayes and Maximum entropy are compared. Both the algorithms are trained, tested and then their accuracy is calculated. The training is done in three phases i.e. while training the training data is divided into 80%, 70% and 60% and the respective remaining used for testing and then the accuracy is found.

The results are as follows:

Data		**Accuracy**	
Testing data	Training Data	Naïve Bayes	Maximum Entropy
20%	80%	77.04%	53.7%
30%	70%	77.08%	54.01%
40%	60%	78.15%	54.25%

Table 1. Accuracy

4.9 DEPLOYMENT

In the deployment phase of project, a website is developed using

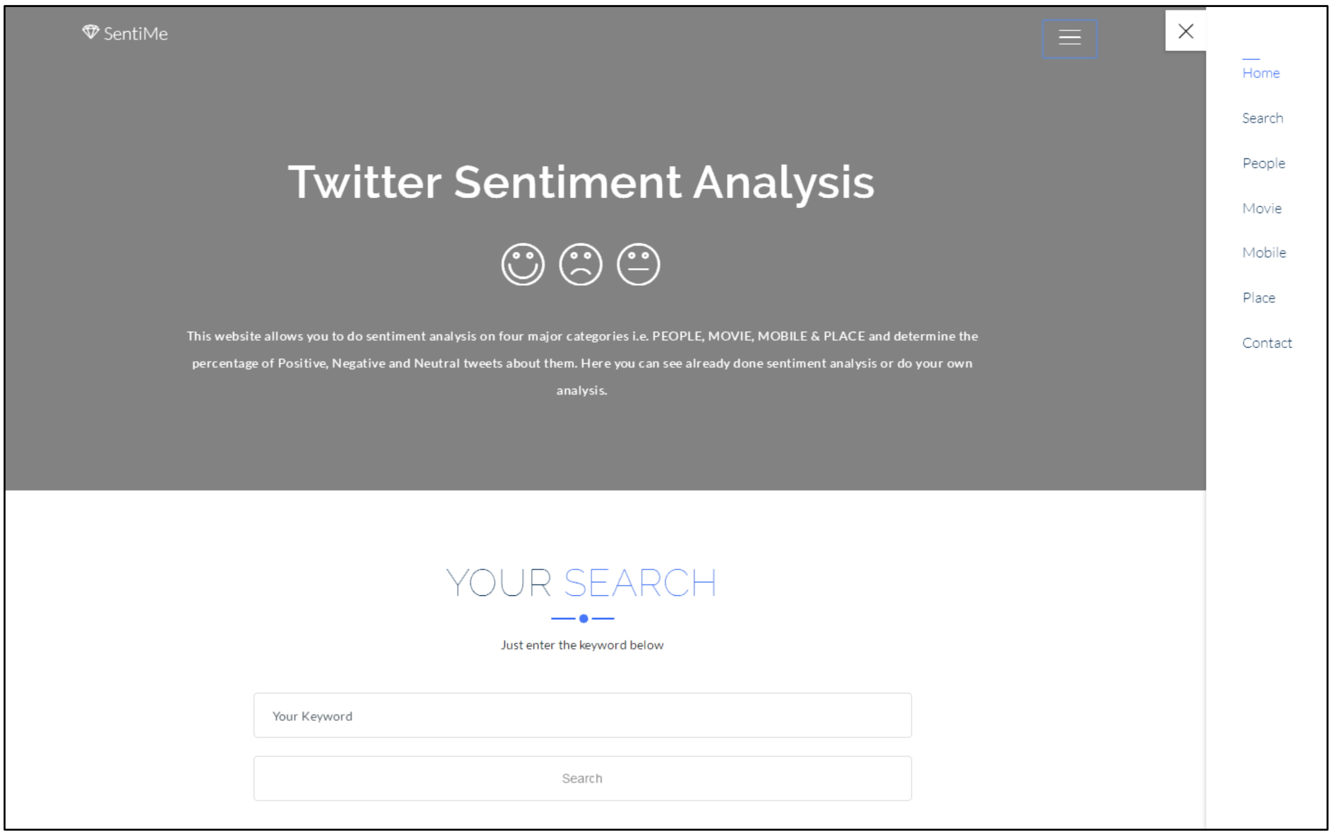
- Front End: HTML, CSS, JavaScript, Bootstrap
- Back End: Python, SQLite, Django

The web site runs on any latest web browsers like Chrome, Opera, Firefox, Edge etc. and the system should have Python 3 or any latest version installed.

In the web site the user can see the already done sentiment analysis of people, movie, mobile and places or the user can do his/her new search.

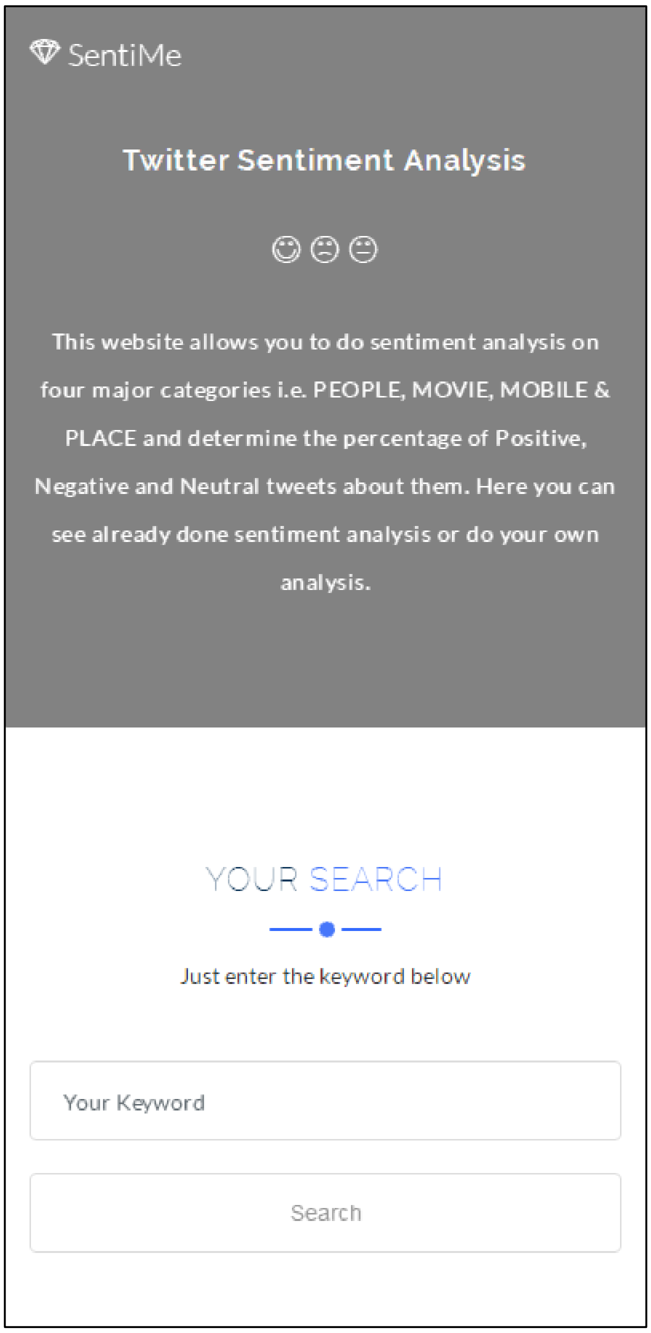
4.9.1 The Web Site

The website is responsive i.e. it is displayed on very large screens as well very small screens without losing any of its attractiveness and the information in it.



This website allows you to do sentiment analysis on four major categories i.e. PEOPLE, MOVIE, MOBILE & PLACE and determine the percentage of Positive, Negative and Neutral tweets about them. Here you can see already done sentiment analysis or do your own analysis.

Figure 7. Front End (Desktop)


SentiMe

Twitter Sentiment Analysis

This website allows you to do sentiment analysis on four major categories i.e. PEOPLE, MOVIE, MOBILE & PLACE and determine the percentage of Positive, Negative and Neutral tweets about them. Here you can see already done sentiment analysis or do your own analysis.

YOUR SEARCH

Just enter the keyword below

Figure 8. Front End (Mobile View)

4.9.2 Search Section

In the search section the user can enter a keyword and see the sentiment analysis for it in the form of a pie chart.

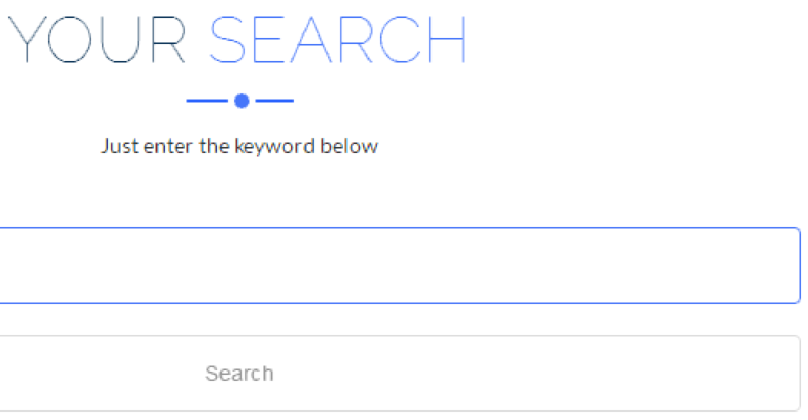


Figure 9. Search Box

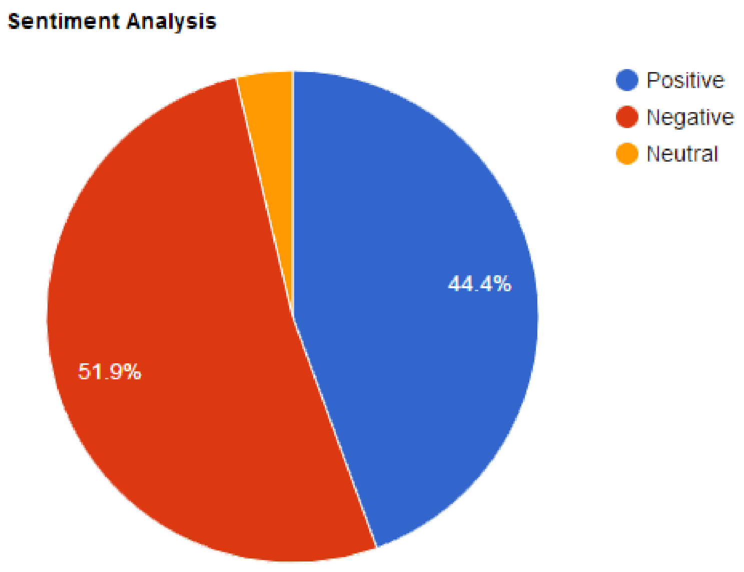


Figure 10. Pie Chart

4.9.3 People Section

In the people section the user can see the already done sentiment analysis of some people.

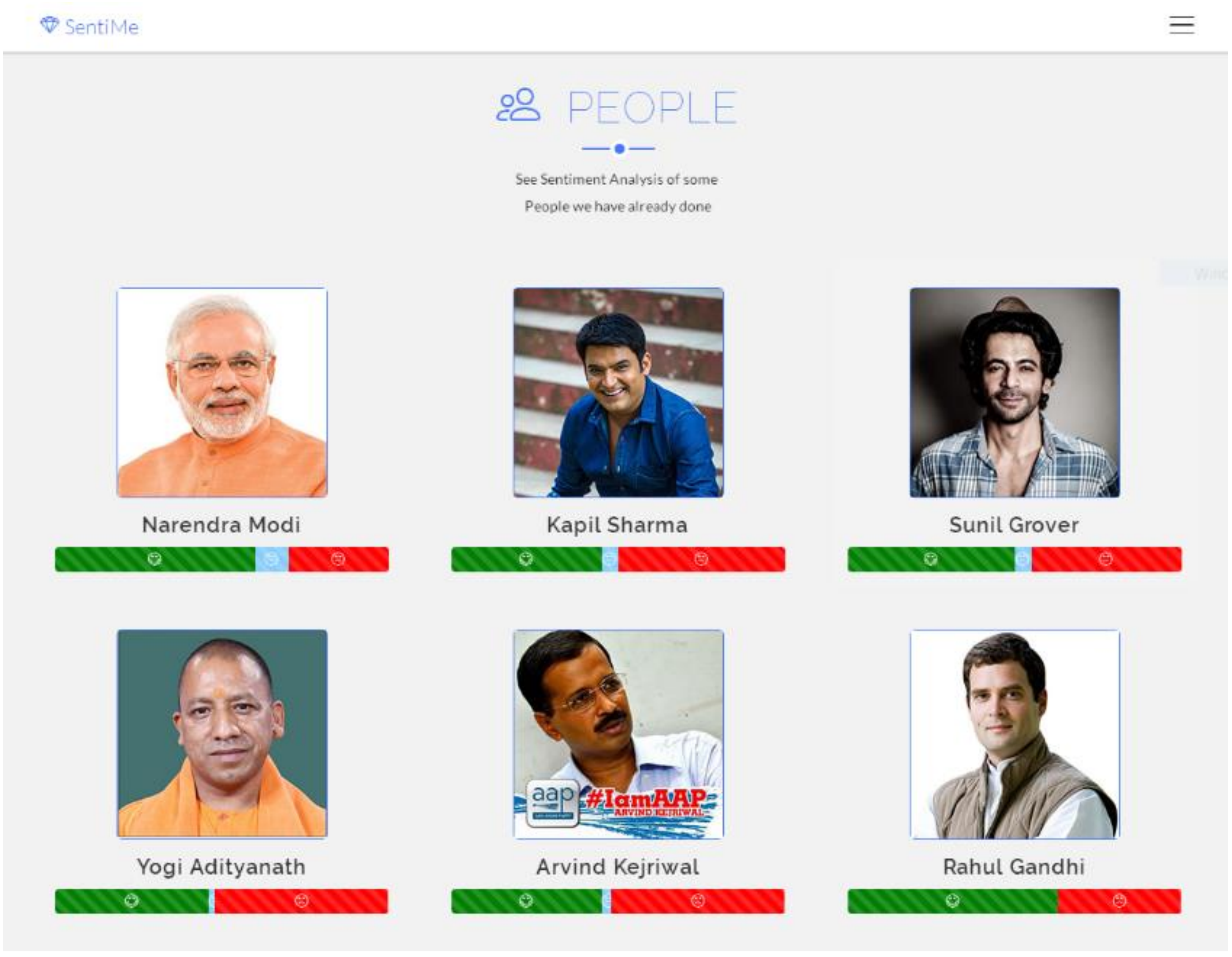


Figure 11. People Section

4.9.4 Movie Section

In the people section the user can see the already done sentiment analysis of some people.

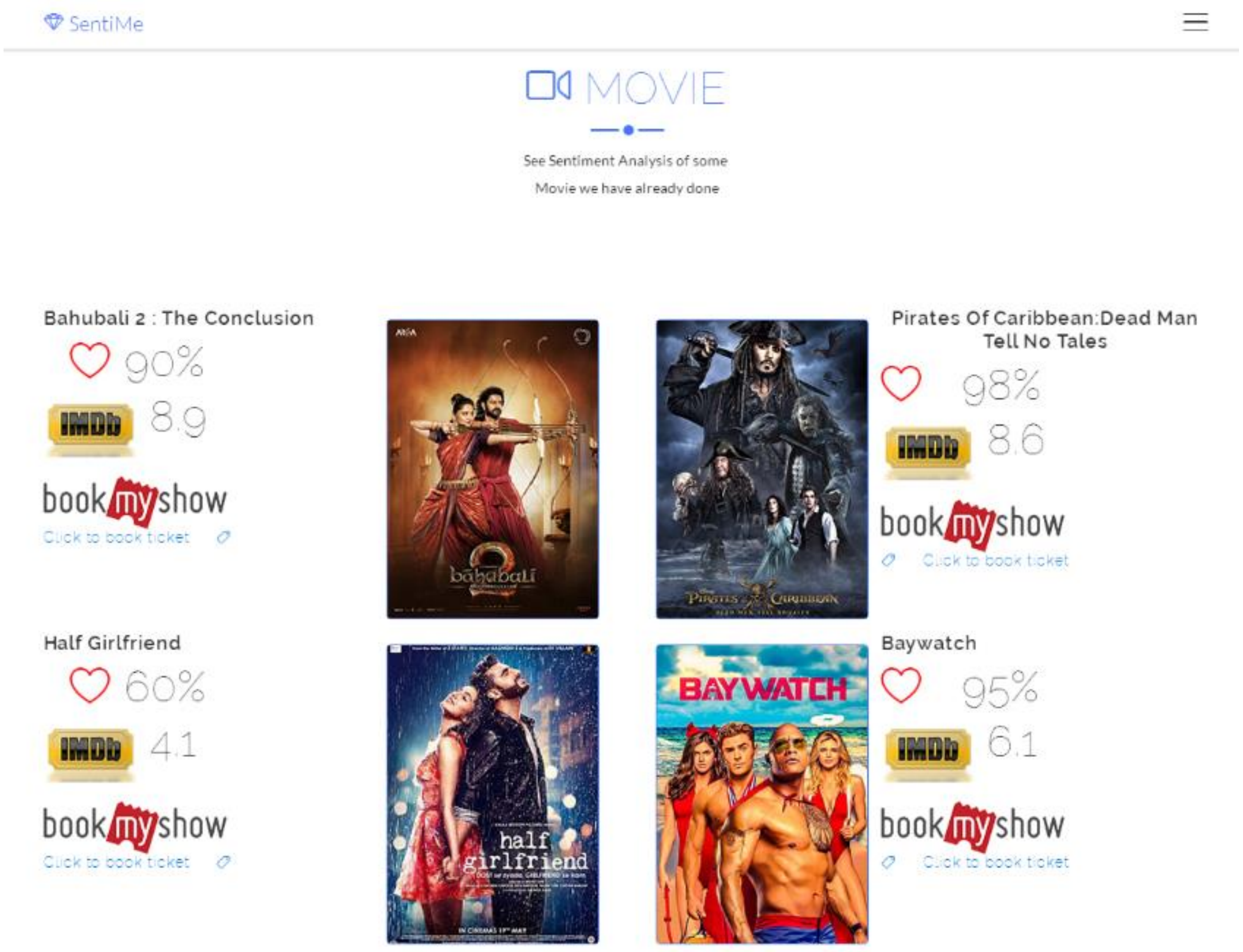
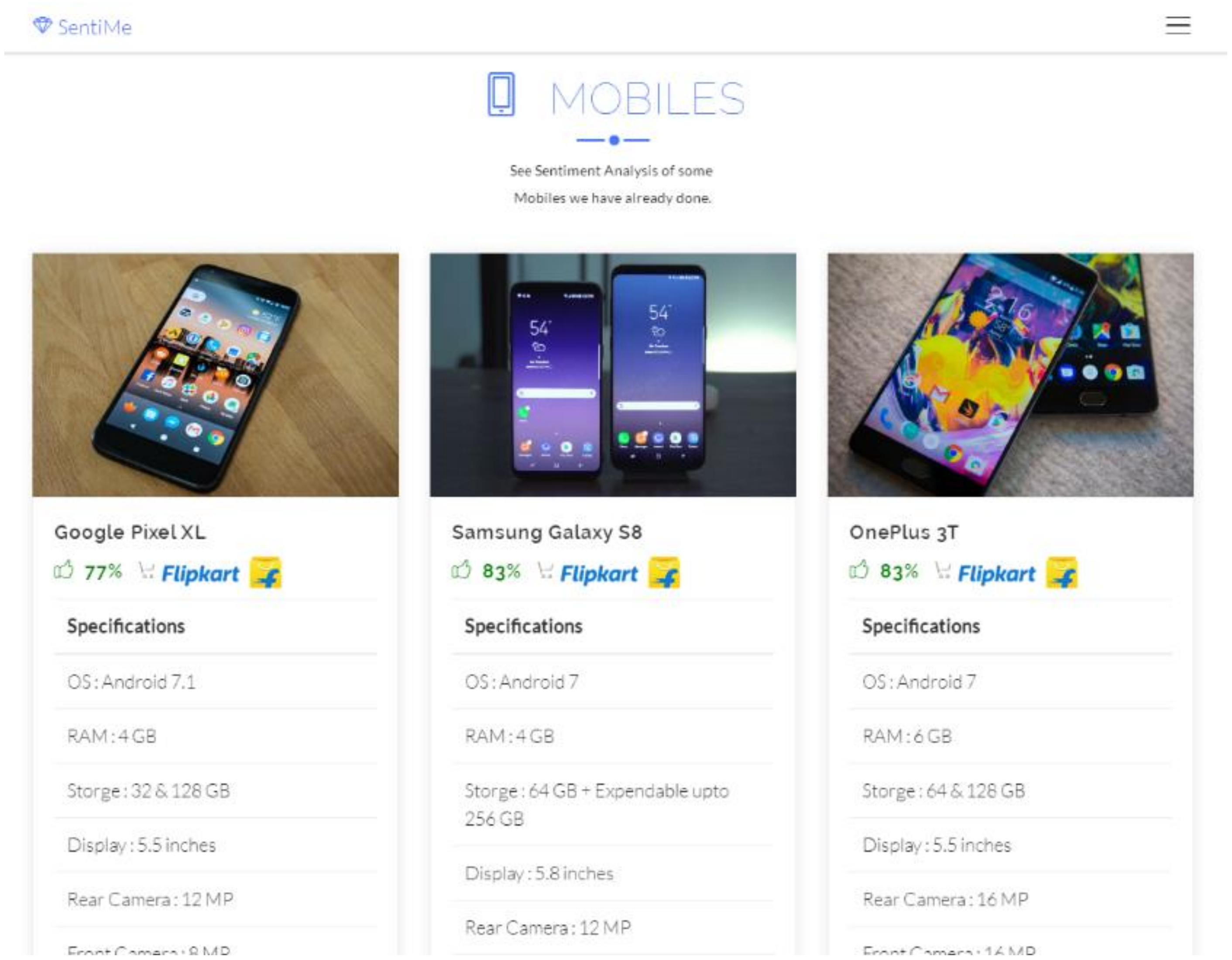


Figure 12. Movie Section

4.9.5 Mobile Section

In the people section the user can see the already done sentiment analysis of some people.



See Sentiment Analysis of some Mobiles we have already done.

Google Pixel XL
77% Flipkart

Specifications
OS : Android 7.1
RAM : 4 GB
Storge : 32 & 128 GB
Display : 5.5 inches
Rear Camera : 12 MP
Front Camera : 8 MP

Samsung Galaxy S8
83% Flipkart

Specifications
OS : Android 7
RAM : 4 GB
Storge : 64 GB + Expendable upto 256 GB
Display : 5.8 inches
Rear Camera : 12 MP

OnePlus 3T
83% Flipkart

Specifications
OS : Android 7
RAM : 6 GB
Storge : 64 & 128 GB
Display : 5.5 inches
Rear Camera : 16 MP
Front Camera : 16 MP

Figure 13. Mobile Section

4.9.6 Place Section

In the people section the user can see the already done sentiment analysis of some people.

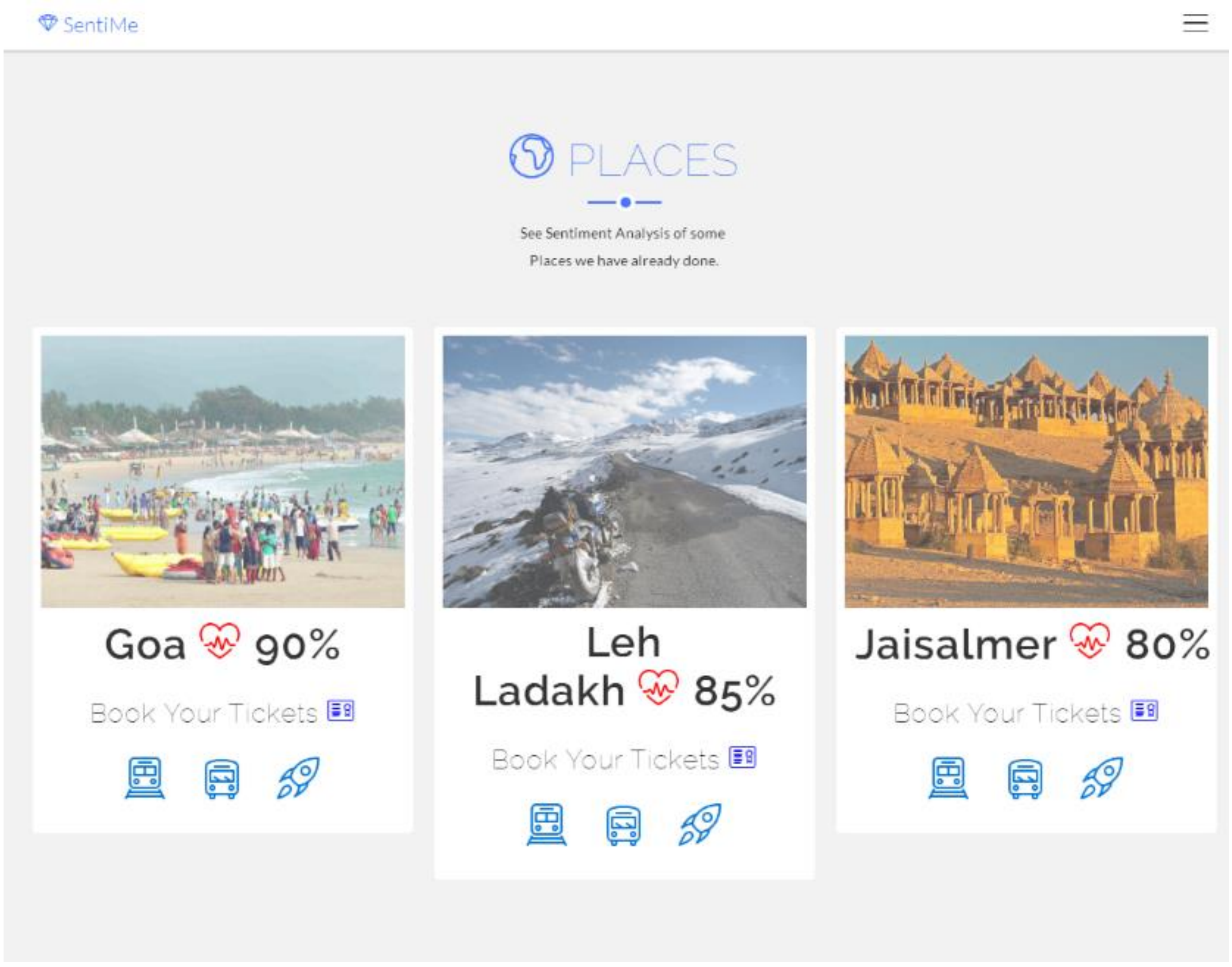


Figure 14. Place Section

CHAPTER-5

CONCLUSION & FUTURE WORK

5.1 CONCLUSION

5.1.1 Result Obtained

Maximum accuracy achieved is 78.93 % using Naive Bayes Algorithm. The accuracy obtained is higher than the accuracy achieved by using Max Entropy Algorithm. Using a novel feature vector of unigrams, machine learning algorithms such as Naive Bayes, and Maximum Entropy achieve competitive accuracy in classifying tweet sentiment

5.1.2 Risk Analysis

Our project is limited to unigram approach of text classification. Bigram approach for text classification is not considered.

- Our project does not handle sarcasm in the tweets.
- Not using bigram would sometime result in opposite sentiment like not good.

5.2 FUTURE WORK

As per our research project, the following future work could be incorporated in project to extend its capabilities.

- Incorporating bigram approach for text classification.
- Bigger Dataset The training dataset in the order of millions will cover a better range of twitter words and hence better unigram feature vector resulting in an overall improved model. This would vastly improve upon the existing classifier results.
- In this project the focus is only on English tweets but Twitter has a huge international audience. It should be possible to use the approach followed in the project to classify sentiment in other languages with a language specific positive/negative keyword list.

REFERENCES

[1] Alec Go, Richa Bhayani, Lei Huang, “Twitter Sentiment Classification using Distant Supervision” in *Technical report*, Stanford Digital Library Technologies Project, 2009,

[2] B. Pang and L. Lee, “Opinion mining and sentiment analysis Foundations and Trends in Information Retrieval”, 2008

[3] G. Mishne, Workshop on “Stylistic Analysis Of Text For Information Access”, 2005

[4] For training data set –www.sentiment140.com

[5] Python Natural Language Toolkit - http://www.nltk.org/

[6] Python 3 Documentation - https://docs.python.org/3/

[7] Django Documentation 1.11 - https://docs.djangoproject.com/en/1.11/

[8] Practical Machine Learning Tutorial - https://pythonprogramming.net/machine-learning-tutorial-python-introduction/

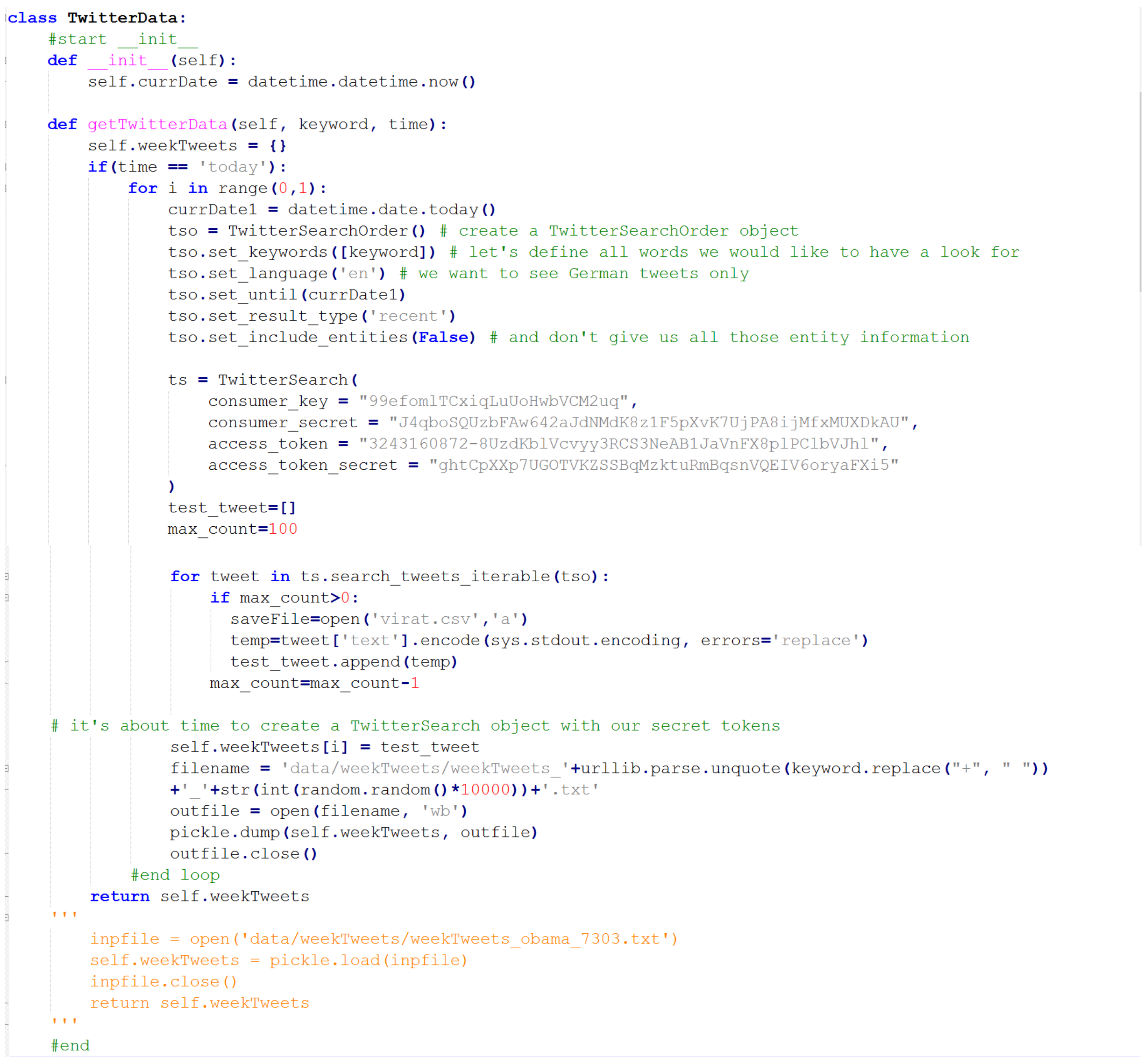
[9] Bootstrap framework - http://getbootstrap.com/

[10] Introduction to Matplotlib Tutorial - https://pythonprogramming.net/matplotlib-intro-tutorial/

APPENDIX

A CODE SNIPPETS

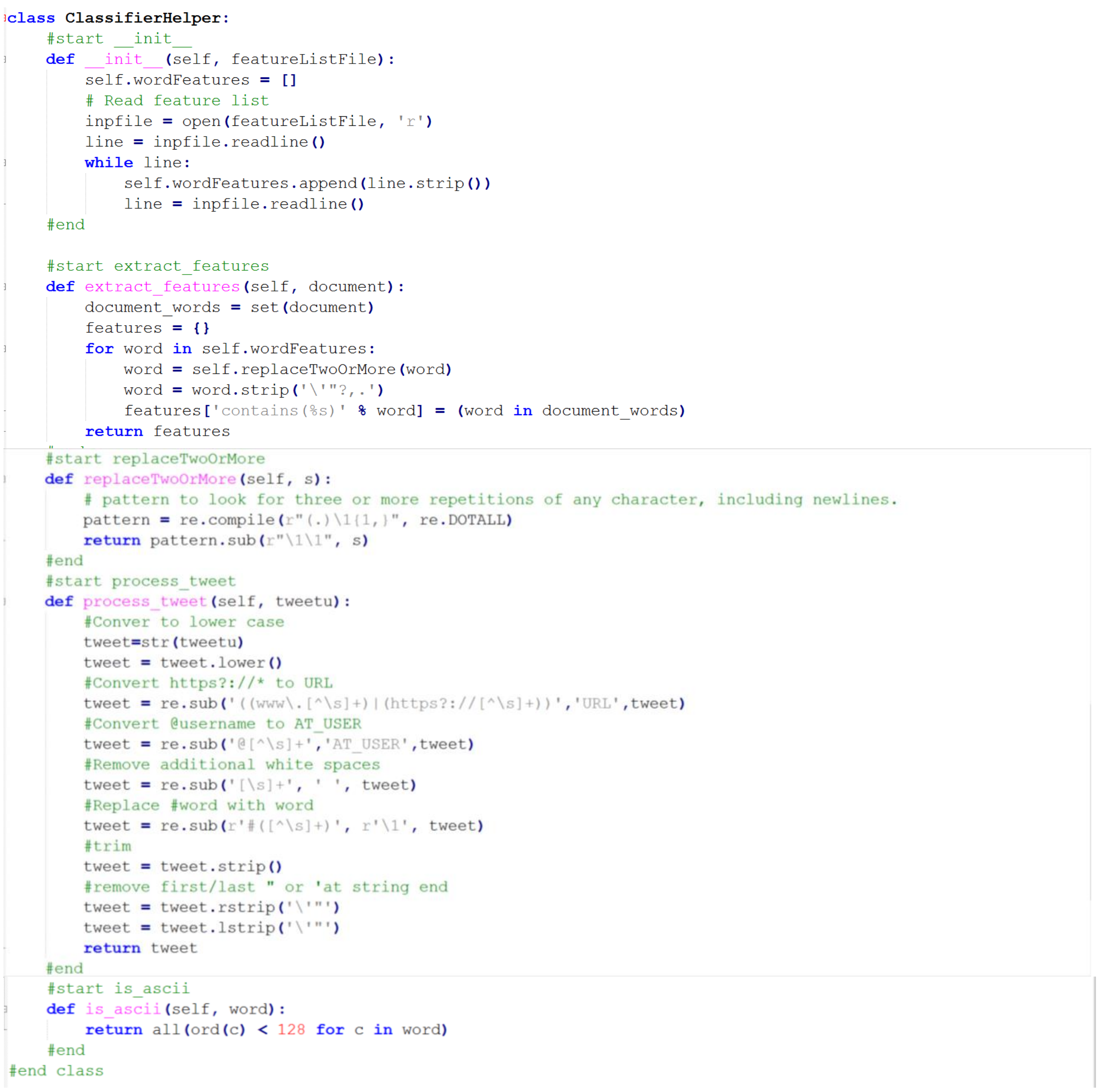
A.1 Get_Twitter_Data.py



```
class TwitterData:
    #start __init__
    def __init__(self):
        self.currDate = datetime.datetime.now()

    def getTwitterData(self, keyword, time):
        self.weekTweets = {}
        if(time == 'today'):
            for i in range(0,1):
                currDate1 = datetime.date.today()
                tso = TwitterSearchOrder() # create a TwitterSearchOrder object
                tso.set_keywords([keyword]) # let's define all words we would like to have a look for
                tso.set_language('en') # we want to see German tweets only
                tso.set_until(currDate1)
                tso.set_result_type('recent')
                tso.set_include_entities(False) # and don't give us all those entity information

                ts = TwitterSearch(
                    consumer_key = "99efomlTCxiqLuUoHwbVCM2uq",
                    consumer_secret = "J4qboSQUzbFAw642aJdNMdK8z1F5pXvK7UjPA8ijMfxMUXDkAU",
                    access_token = "3243160872-8UzdKblVcvyy3RCS3NeAB1JaVnFX8plPClbVJhl",
                    access_token_secret = "ghtCpXXp7UGOTVKZSSBqMzktuRmBqsnVQEIV6oryaFXi5"
                )
                test_tweet=[]
                max_count=100

                for tweet in ts.search_tweets_iterable(tso):
                    if max_count>0:
                      saveFile=open('virat.csv','a')
                      temp=tweet['text'].encode(sys.stdout.encoding, errors='replace')
                      test_tweet.append(temp)
                    max_count=max_count-1

    # it's about time to create a TwitterSearch object with our secret tokens
                self.weekTweets[i] = test_tweet
                filename = 'data/weekTweets/weekTweets_'+urllib.parse.unquote(keyword.replace("+", " "))
                +'_'+str(int(random.random()*10000))+'.txt'
                outfile = open(filename, 'wb')
                pickle.dump(self.weekTweets, outfile)
                outfile.close()
            #end loop
        return self.weekTweets
    '''
        inpfile = open('data/weekTweets/weekTweets_obama_7303.txt')
        self.weekTweets = pickle.load(inpfile)
        inpfile.close()
        return self.weekTweets
    '''
    #end
```

Figure 15. Fetching Tweets

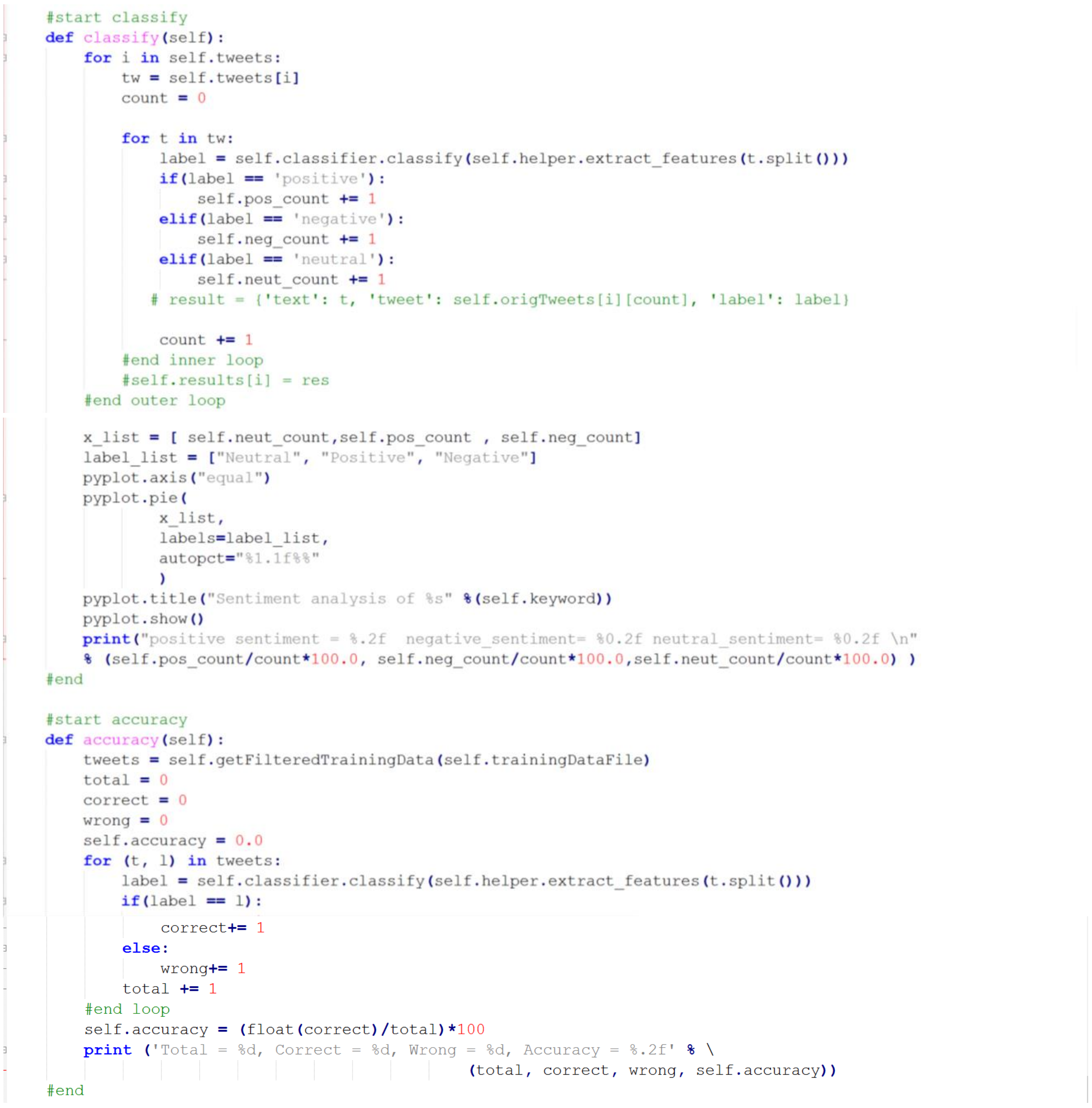
A.2 Classifier_helper.py



```
class ClassifierHelper:
    #start __init__
    def __init__(self, featureListFile):
        self.wordFeatures = []
        # Read feature list
        inpfile = open(featureListFile, 'r')
        line = inpfile.readline()
        while line:
            self.wordFeatures.append(line.strip())
            line = inpfile.readline()
    #end

    #start extract_features
    def extract_features(self, document):
        document_words = set(document)
        features = {}
        for word in self.wordFeatures:
            word = self.replaceTwoOrMore(word)
            word = word.strip('\'"?,.')
            features['contains(%s)' % word] = (word in document_words)
        return features
    #start replaceTwoOrMore
    def replaceTwoOrMore(self, s):
        # pattern to look for three or more repetitions of any character, including newlines.
        pattern = re.compile(r"(.)\1{1,}", re.DOTALL)
        return pattern.sub(r"\1\1", s)
    #end
    #start process_tweet
    def process_tweet(self, tweetu):
        #Conver to lower case
        tweet=str(tweetu)
        tweet = tweet.lower()
        #Convert https?://* to URL
        tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)
        #Convert @username to AT_USER
        tweet = re.sub('@[^\s]+','AT_USER',tweet)
        #Remove additional white spaces
        tweet = re.sub('[\s]+', ' ', tweet)
        #Replace #word with word
        tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
        #trim
        tweet = tweet.strip()
        #remove first/last " or 'at string end
        tweet = tweet.rstrip('\'"')
        tweet = tweet.lstrip('\'"')
        return tweet
    #end
    #start is_ascii
    def is_ascii(self, word):
        return all(ord(c) < 128 for c in word)
    #end
#end class
```

Figure 16. Classifier Helper

A.3 Naive_bayes_classifier.py



```
#start classify
def classify(self):
    for i in self.tweets:
        tw = self.tweets[i]
        count = 0

        for t in tw:
            label = self.classifier.classify(self.helper.extract_features(t.split()))
            if(label == 'positive'):
                self.pos_count += 1
            elif(label == 'negative'):
                self.neg_count += 1
            elif(label == 'neutral'):
                self.neut_count += 1
           # result = {'text': t, 'tweet': self.origTweets[i][count], 'label': label}

            count += 1
        #end inner loop
        #self.results[i] = res
    #end outer loop

    x_list = [ self.neut_count,self.pos_count , self.neg_count]
    label_list = ["Neutral", "Positive", "Negative"]
    pyplot.axis("equal")
    pyplot.pie(
            x_list,
            labels=label_list,
            autopct="%1.1f%%"
            )
    pyplot.title("Sentiment analysis of %s" %(self.keyword))
    pyplot.show()
    print("positive sentiment = %.2f  negative_sentiment= %0.2f neutral_sentiment= %0.2f \n"
    % (self.pos_count/count*100.0, self.neg_count/count*100.0,self.neut_count/count*100.0) )
#end

#start accuracy
def accuracy(self):
    tweets = self.getFilteredTrainingData(self.trainingDataFile)
    total = 0
    correct = 0
    wrong = 0
    self.accuracy = 0.0
    for (t, l) in tweets:
        label = self.classifier.classify(self.helper.extract_features(t.split()))
        if(label == l):
            correct+= 1
        else:
            wrong+= 1
        total += 1
    #end loop
    self.accuracy = (float(correct)/total)*100
    print ('Total = %d, Correct = %d, Wrong = %d, Accuracy = %.2f' % \
                                            (total, correct, wrong, self.accuracy))
#end
```

Figure 17. Naive Bayes Classifier