

Abstract

In this project, we are attempting to create a system capable of classifying a given text as having positive or negative sentiment. Sentiment analysis also known as opinion mining involves using text processing, natural language processing and computational linguistics to extract relevant information about the subject under consideration.(Wikipedia defn.)

Twitter is a popular micro blogging service where users create status messages called tweets. Tweets are short messages with a maximum length of 140 characters. Authors of those messages write about their lives , share opinions on a variety of topics and discuss current issues. As more and more users post about the products and services they use, or express their political and religious views, micro blogging websites become valuable sources of peoples opinions and sentiments.

Contents

Contents	i
List of Figures	iii
List of Tables	iv
1 Introduction	1
1.1 Subtasks	1
1.2 Methods and features	2
1.3 Evaluation	4
1.4 Sentiment analysis and Web 2.0	4
1.5 Bag Of Words	6
1.6 Classifiers	7
1.6.1 Naive Bayes Classifier	8
1.6.2 Maximum Entropy Classifier	9
1.6.3 Boosted Trees classifier	10
1.6.4 Random Forest Classifier	11
1.6.5 Support Vector Machines	13
1.7 Findings	13
2 Related Work/Literature Survey	14
3 System Design	23
3.1 Naive Bayes Classifier	24
3.2 Implementation Details	25
4 Testing and result analysis	30
4.1 Training Corpus	30
4.2 Retrieving tweets for a particular topic	30
4.3 Most Informative Features	31
5 Conclusion/Future Work	37

6	References	39
A	Appendix	43

List of Figures

1.1	Maximum a posteriori decision rule.	8
3.1	The Overall view of the sytem	23
4.1	The Overall view of the sytem	32
4.2	Distribution of sentiment for search keyword 'modi'	35
4.3	Distribution of sentiment for keyword ipl	36

List of Tables

1.1	Comparison of classifiers	12
4.1	Most informative features found after using 500 tweets for training	31
4.2	Most informative features found after using 2000 tweets for training	33
4.3	Most informative features found after using 10000 tweets for training	34

Listings

3.1	Feature extractor	26
3.2	TesterFunc and classifier.main()	26
3.3	Cleaner.py	28
3.4	Main oauth request	29

Chapter 1

Introduction

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author when writing), or the intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader).

1.1 Subtasks

A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry," "sad," and "happy." Early work in that area includes application of different methods for detecting the polarity of product reviews and movie reviews respectively. This work is at the document level. One can also classify a document's polarity on a multi-way scale, which was attempted by Pang and Snyder expanded the basic task of classifying a movie review as either positive or negative to predicting star ratings on either a 3 or a 4 star scale, while Snyder performed an in-depth analysis of restaurant reviews, predicting ratings for various aspects of the given restaurant, such as the food and atmosphere (on a five-star scale). Even though in most statistical classification methods, the neutral class is ignored under the assumption that neutral texts lie near the boundary of

the binary classifier, several researchers suggest that, as in every polarity problem, three categories must be identified. Moreover it can be proven that specific classifiers such as the Max Entropy and the SVMs can benefit from the introduction of neutral class and improve the overall accuracy of the classification. A different method for determining sentiment is the use of a scaling system whereby words commonly associated with having a negative, neutral or positive sentiment with them are given an associated number on a -10 to +10 scale (most negative up to most positive) and when a piece of unstructured text is analyzed using natural language processing, the subsequent concepts are analyzed for an understanding of these words and how they relate to the concept. Each concept is then given a score based on the way sentiment words relate to the concept, and their associated score. This allows movement to a more sophisticated understanding of sentiment based on an 11 point scale. Alternatively, texts can be given a positive and negative sentiment strength score if the goal is to determine the sentiment in a text rather than the overall polarity and strength of the text.

A more fine-grained analysis model is called the feature/aspect-based sentiment analysis. It refers to determining the opinions or sentiments expressed on different features or aspects of entities, e.g., of a cell phone, a digital camera, or a bank. A feature or aspect is an attribute or component of an entity, e.g., the screen of a cell phone, or the picture quality of a camera. This problem involves several sub-problems, e.g., identifying relevant entities, extracting their features/aspects, and determining whether an opinion expressed on each feature/aspect is positive, negative or neutral.[13] More detailed discussions about this level of sentiment analysis can be found in Liu's NLP Handbook chapter, "Sentiment Analysis and Subjectivity".

1.2 Methods and features

Existing approaches to sentiment analysis can be grouped into four main categories:

1. **Keyword Spotting** : Keyword spotting classifies text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored.
2. **Lexical affinity** : Lexical affinity not only detects obvious affect words, it also assigns arbitrary words a probable affinity to particular emotions.
3. **Statistical methods** : Statistical methods leverage on elements from

machine learning such as latent semantic analysis, support vector machines, "bag of words" and Semantic Orientation Pointwise Mutual Information. More sophisticated methods try to detect the holder of a sentiment (i.e. the person who maintains that affective state) and the target (i.e. the entity about which the affect is felt). To mine the opinion in context and get the feature which has been opinionated, the grammatical relationships of words are used. Grammatical dependency relations are obtained by deep parsing of the text.

4. **Concept-level techniques** : Unlike purely syntactical techniques, concept-level approaches leverage on elements from knowledge representation such as ontologies and semantic networks and, hence, are also able to detect semantics that are expressed in a subtle manner, e.g., through the analysis of concepts that do not explicitly convey relevant information, but which are implicitly linked to other concepts that do so.

Open source software tools deploy machine learning, statistics, and natural language processing techniques to automate sentiment analysis on large collections of texts, including web pages, online news, internet discussion groups, online reviews, web blogs, and social media. Knowledge-based systems, instead, make use of publicly available resources, e.g., WordNet-Affect, SentiWordNet, and SenticNet, to extract the semantic and affective information associated with natural language concepts. Sentiment Analysis can also be performed on visual content i.e. images and videos.

A human analysis component is required in sentiment analysis, as automated systems are not able to analyze historical tendencies of the individual commenter, or the platform and are often classified incorrectly in their expressed sentiment. Automation impacts approximately 23% of comments that are correctly classified by humans.

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

1.3 Evaluation

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

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

1.4 Sentiment analysis and Web 2.0

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

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

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

it becomes.

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

1.5 Bag Of Words

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. Recently, the bag-of-words model has also been used for computer vision.

The bag-of-words model is commonly used in methods of document classification, where the (frequency of) occurrence of each word is used as a feature for training a classifier.

Example implementation The following models a text document using bag-of-words.

Here are two simple text documents:

- John likes to watch movies. Mary likes movies too.
- John also likes to watch football games.

Based on these two text documents, a dictionary is constructed as:

```
{  
  "John": 1,  
  "likes": 2,  
  "to": 3,  
  "watch": 4,  
  "movies": 5,  
  "also": 6,  
  "football": 7,  
  "games": 8,  
  "Mary": 9,  
  "too": 10  
}
```

which has 10 distinct words. And using the indexes of the dictionary, each document is represented by a 10-entry vector where each entry of the vectors refers to count of the corresponding entry in the dictionary (this is also the histogram representation). For example, in the first vector (which represents document 1), the first two entries are "1,2". The first entry corresponds to the word "John" which is the first word in the dictionary, and its value is "1" because "John" appears in the first document 1 time. Similarly, the second entry corresponds to the word "likes" which is the second word in the dictionary, and its value is "2" because "likes" appears in the first document 2 times. This vector representation does not preserve the order of the words in the original sentences. This kind of representation has several successful

applications, for example email filtering. Term weighting[edit] In the example above, the document vectors contain term frequencies. In both IR and text classification, it is common to weigh terms by various schemes, the most popular of which is tfidf. For the specific purpose of classification, supervised alternatives have been developed that take into account the class label of a document.[3] Additionally, binary (presence/absence or 1/0) weighting is used in place of frequencies for some problems. (For instance, this option is implemented in the WEKA machine learning software system.) ...

Hashing trick A common alternative to the use of dictionaries is the hashing trick, where words are directly mapped to indices with a hashing function. By mapping words to indices directly with a hash function, no memory is required to store a dictionary. Hash collisions are typically dealt with by using freed-up memory to increase the number of hash buckets. In practice, hashing greatly simplifies the implementation of bag-of-words models and improves their scalability.

Example usage: spam filtering In Bayesian spam filtering, an e-mail message is modeled as an unordered collection of words selected from one of two probability distributions: one representing spam and one representing legitimate e-mail ("ham"). Imagine that there are two literal bags full of words. One bag is filled with words found in spam messages, and the other bag is filled with words found in legitimate e-mail. While any given word is likely to be found somewhere in both bags, the "spam" bag will contain spam-related words such as "stock", "Viagra", and "buy" much more frequently, while the "ham" bag will contain more words related to the user's friends or workplace.

To classify an e-mail message, the Bayesian spam filter assumes that the message is a pile of words that has been poured out randomly from one of the two bags, and uses Bayesian probability to determine which bag it is more likely to be.

1.6 Classifiers

The field of information extraction and retrieval has grown exponentially in the last decade. Sentiment analysis is a task in which you identify the polarity of given text using text processing and classification. There are various approaches in the task of classification of text into various classes. Use of particular algorithms depends on the kind of input provided. Analyzing and understanding when to use which algorithm is an important aspect and can help in improving accuracy of results.

Classification is a stage in sentiment analysis that can be described as a process in which we predict qualitative response, or in this case we classify the

document into its polarity. Predicting a qualitative response of a document can be referred to as classifying the document since it involves assigning an observation to a category or class. There are many possible classification techniques, or classifiers that one might use for to predict the qualitative response or class of a document. In sentiment analysis some widely used classification techniques are as follows:

- Naive Bayes Classifier
- Max Entropy Classifier
- Boosted Trees Classifier
- Random Forest Classifier

In this section we have showcased a comparative study of the above stated classification algorithms that are widely used in sentiment classification.

1.6.1 Naive Bayes Classifier

Naive Bayes classifier is based on Bayes theorem. Its a baseline classification algorithm. Naive Bayes classifier assumes that the classes for classification are independent. Though this is rarely true Bayesian classification has shown that there are some theoretical reasons for this apparent unreasonable efficiency. There are various proofs that show that even though the probability estimates of Naive Bayes classification are low it delivers quite good results in real life examples. Naive Bayes just over estimates the class that certain object belongs too. Assuming that we are using it only for making decisions (which is true in case of sentiment analysis problem) the decision making is correct and the model is useful [4]. In Text classification we tokenize the document in order to classify it in its appropriate class. By using the Posterior Probability Decision rule we get the following classifier:

$$\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} p(C_k) \prod_{i=1}^n p(x_i | C_k).$$

Figure 1.1: Maximum a posteriori decision rule.

In the above equation t_k are the tokens / words in the document, C is the set of classes used in classification, $P(c|d)$ is the conditional probability of class c given the document d , $P(c)$ is the prior probability of class C and $P(t_k|C)$ is the conditional probability of token t_k given class C . This means that in order to find in which class we should classify a new document, we must estimate the product of the probability of each word of the document

given a particular class (likelihood), multiplied by the probability of the particular class (prior). After calculating the above for all the classes of set C , we select the one with the highest probability. Naive Bayes is used as a classifier in various real world problems like Sentiment analysis, email Spam Detection, email Auto Grouping, email sorting by priority, Document Categorization and Sexually explicit content detection. The major advantage of Naive Bayes is it requires low processing memory and less time for execution. Its advised that this classifier should be used when Training time is a crucial factor in the system. Naive Bayes is the baseline algorithm for researches in decision level classification problem. In presence of limited resources in terms of CPU and Memory Naive Bayes is recommended classifier.

1.6.2 Maximum Entropy Classifier

Another well-known classifier is the Max Entropy Classifier or MaxEnt as some people prefer to call it. The idea behind MaxEnt classifiers is that we should prefer the most uniform models that satisfy any given constraint. MaxEnt models are feature based models. We use these features to find a distribution over the different classes using logistic regression. The probability of a particular data point belonging to a particular class is calculated as follows: Where, c is the class, d is the data point we are looking at, and w is a weight vector. MaxEnt makes no independence assumptions for its features, unlike Naive Bayes. This means we can add features like bigrams and phrases to MaxEnt without worrying about feature overlapping. The principle of maximum entropy is useful explicitly only when applied to testable information. A piece of information is testable if it can be determined whether a given distribution is consistent with it. For example, the statements - The expectation of the variable x is 2.87 and $p_2 + p_3 \geq 0.6$ are statements of testable information. Given testable information, the maximum entropy procedure consists of seeking the probability distribution which maximizes information entropy (This is the average amount of data that one data set will contain.), subject to the constraints of the information. Entropy maximization with no testable information takes place under a single constraint: the sum of the probabilities must be one. Under this constraint, the maximum entropy discrete probability distribution is the uniform distribution. Various results that we came across in our study exclusively mentioned that Naive Bayes theorem has very less efficiency than a simple Max Entropy Algorithm. Our research revealed a variation in Max Entropy known as Max Entropy using Priors which enhances the efficiency of MaxEnt classifier by using Prior Results as a part of training Data set. MaxEnt using Priors is more effective in Natural language processing applications. The Major Advantages of using

MaxEnt or its variations can be listed out as follows: Accuracy Consistency This algorithm shows consistency in results and if priors are used results also improve over a period of time. Performance / Efficiency - Can handle huge amounts of data Flexibility - The algorithm is flexible of having many different typed of data in a unified platform and classifyit accordingly [8]

1.6.3 Boosted Trees classifier

Boosted trees is a classifier that is basically a combination of Boosting and Decision Trees. Boosting is a machine Meta learning algorithm for reducing preconception in supervised learning. In Boosting predictive classifiers are used to develop weighted trees which are further combined into single prediction models. Boosted trees combine the strengths of two algorithms:

- **regression trees** models that relate a response to their predictors by recursive binary splits.
- **boosting** An adaptive method for combining many simple models to give improved predictive performance.

and . Most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. When they are added, they are typically weighted in some way that is usually related to the weak learners accuracy. After a weak learner is added, the data is rerated and new weights are produced and examples that are incorrectly classified gain weight and examples that are classified correctly lose weight (some boosting algorithms actually decrease the weight of repeatedly misclassified examples, e.g., boost by majority and Brown Boost). There are many variants of Boosting algorithms some of them are Ada Boost, LP Boost, Total Boost, Logit Boost, Gradient Boosted Regression Trees etc. Boosting algorithms such as AdaBoost are known to perform well for classification and are very resistant to over fitting with respect to misclassification error, even though conditional class probability estimates eventually diverge to zero and one, implying complete over fit in terms of CCPF (Conditional Class Probabilities) estimation but not classification. Gradient boosting is a machine learning technique for regression problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function . The gradient boosting method can also be used for classification problems by reducing them to regression with a suitable loss function. Certain advantages of Boosted trees classifier are - Fast Training

without sacrificing accuracy, Can handle different types of predictor variables and accommodate missing data. On the contrary a major disadvantage is inability to compute conditional class probabilities.

1.6.4 Random Forest Classifier

Random forests are an ensemble learning method for classification that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. It produces multi-altitude decision trees at inputting phase and output is generated in the form of multiple decision trees. The correlation between trees is reduced by randomly selecting trees and thus the prediction power increases and leads to increase in efficiency. The predictions are made by aggregating the predictions of various ensemble data sets. Studies show that the performance is seen always rising. There is no downtrend for the performance of this algorithm in any available data sets. Applications and real life of examples Random Forests are widespread. There is no single type for RF data sets. They can vary from any kind of applications like medical as well as general data sets. Decrease is less relevant data set to 50 classifications in lowering the result accuracy. RF is a parallelized and multi-core friendly algorithm. So simultaneous running of different trees is also a support feature. The popularity of this machine increased with practical machine learning research and their related algorithms. We came across experimental results in our study in which people had used Random Forest for Opinion mining and have found impressive accuracy in classification of their data sets. The Major advantages of this algorithm can be listed out as follows:

- Easy to interpret and understand
- Non-parametric so you dont have to worry about the linearity of the input data set.
- If parameters are there they can be easily entered thus eliminating the need for pruning the trees.
- The classification model is fast and scalable
- Importance and relevance of text/tokens in a class is automatically generated
- Robust to irrelevant text present in document.

Features	Naive Bayes	Max Entropy	Boosted Trees	Random Forest
Based On	Bayes Theorem	Feature Based Classifier	Decision Tree Learning	Decision Tree Aggregation
Simplicity	Very Simple	Hard	Moderate	Simple
Performance	Better	Good	Good	Excellent
Accuracy	Great	High	Poor	Excellent
Memory requirement	Low	High	Low	High
Other Application	Spam Detection, Document Classification	Diagnosis Tests in Pathology labs	Classifying Cardiovascular Outcomes	Biomedical Applications
Result Accuracy Over a Period of Time	Variable	Consistent	Incremental	Incremental
Time Required For Training Classifier	Less	Moderate	High	Recurrent Learning with every novel Dataset

Table 1.1: Comparision of classifiers

A disadvantage that we came across in our study is that the random forest classifier easily over fits its class. (This over fit can be solved. Since data sets have more number of trees and vague links or the data sets that you have provided are too small then the model over fits. To reduce the over fits reduce the number of trees in a random forest classifier as well as decrease the vague links that are present.)

1.6.5 Support Vector Machines

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

1.7 Findings

Study makes it pretty evident that every kind of classification model has its own benefits and drawbacks. Selection of classification models can be decided on the basis of resources, accuracy requirement, training time available etc. Considering sentiment analysis the Random Forest classifier clearly has an upper hand with high accuracy and performance, simplicity in understanding, and improvement in results over a period of time. This makes the classifier best fit for situations like sentiment analysis. Though it requires high training time and processing power the improved accuracy due to aggregation of decision trees, more than makes up for other shortcomings. Random Forest is also very well supported in terms of implementation. Many libraries are available in programming languages like java, python and R which makes it easy to use as well. We can conclude that if accuracy is at our highest priority then we must prefer a classifier model like Random Forest that consumes high learning time but has best accuracy. If processing power and memory is an issue then the Naive Bayes classifier should be selected due to its low memory & processing power requirements. If less training time is available but you have powerful processing system and memory then Max Entropy proves to be a worthy alternative. Selection of a classification model should be done wisely in sentiment analysis systems because this decision will influence the precision of your system and your end product.

Chapter 2

Related Work/Literature Survey

Glivia et. al. [1] assesses the usefulness of twitter hashtags in sentiment analysis. They analyzed 10,173,382 tweets related to the Brazilian Presidential elections in 2010. They analyzed these tweets and observed that the positive behavior of the tweeters across time was in accordance to the hypothesis that hashtags sentiments match the overall population sentiment. They also verified that the information propagation in twitter follows a cascade model where people make their decisions consciously or not, based on someone elses sentiments and choices. Alec Go et. al. [4] introduced a method for classifying twitter messages. First, the query term is normalized so that the query term by itself is not biased. Positive and negative tweets are collected by using :) / :-) and :(/ :-(. 80,000 positive and 80,000 negative tweets were collected as training set. The tweets are then preprocessed. The emoticons are then stripped from the training data because emoticons have a negative impact on the accuracies of SVM and Maximum Entropy classifiers but little effect on Naive Bayes classifier. They explored the usage of unigrams, bigrams, unigrams and bigrams, and Parts of Speech features. The following were the accuracies observed using different classifiers.

Table 1 Accuracies observed using different classifiers

	Naive Bayes	Maximum Entropy	SVM
Unigram	81,4%	80.4%	82.9%
Unigram and Bigram	82.7%	82.7%	81.9%
Parts of speech	81.5%	81.9%	80,4%

Apoorv Agarwal et. al. [8] have built models for two classification tasks: a binary task of classifying sentiment into positive and negative classes and a

3-way task of classifying sentiment into positive, negative and neutral classes. Experimentation is done with unigram model, feature based model and tree kernel based model. For the tree kernel based model they designed a new tree representation for tweets. They used the unigram model for sentiment analysis for Twitter data, as their baseline. Their experiments show that a unigram model is indeed a hard baseline achieving over 20% over the chance baseline for both classification tasks. Their feature based model that uses only 100 features achieves similar accuracy as the unigram model that uses over 10,000 features. Their tree kernel based model outperforms both these models by a significant margin. They also experimented with a combination of models: combining unigrams with features and combining features with the tree kernel. Both these combinations were found to outperform the unigram baseline by over 4% for both classification tasks.

Alexander Pak et. al. [6] used Twitter API to collect twitter datasets in three categories namely positive, negative and neutral. A corpus analysis is performed by checking the distribution of word frequencies in the corpus. The plot was found to obey Zip's law and hence confirmed a proper characteristic of the collected corpus. The corpus analysis is done in one more method. The parts of speech of each word are tagged using TreeTagger (Schmid, 1994) and interpreted the difference of tag distributions between sets of text (positive, negative, neutral or subjective, objective). The sentiments are classified using both Naive Bayes and SVM classifiers. Naive Bayes was found to perform better. Two Bayes classifiers were trained, one based on n-gram and another based on parts of speech distribution. To increase the accuracy some common n-grams are discarded that do not strongly indicate any sentiment. The best performance was achieved when a bigram was used.

Oshini Goonetilleke et. al. [21] address the issues around Big data nature of twitter and the need for new data management which limits the use of existing systems. For this reason, the system basically involves components like Focused Crawling which is used in effective retrieval and better coverage (twiiterecho is an open distributed crawler for twitter), pre-processing of tweets can be done using Tokenization and Stemming. Twitterzombie is a platform used for gathering the data and analyzing. To provide scalability and efficiency of processing large amounts of data, Tred Miner can be used for real time analysis of tweets. Tweepql allows the resulting tweets to be collected in batches and then stores them in relational data base for further identification of sentiments.

Martha Arias et. al. [23] have focused on study of hashtag level of sentiment classification. The main aim of this task is to automatically generate the overall sentiment for a given hashtag in twitter. It is found that out of

0.6 million randomly selected tweets 14.6% contained at least one hashtag. A two stage SVM classifier is used to determine the polarity of sentiment and finally, it is found that performance can be invariably increased by Boosting loopy belief propagation with accuracy prediction up to 77.72%. A simple baseline approach is developed on the results of tweets using simple voting strategy where a binary value was given to corresponding positive, negative and neutral tweets.

Po-Wei Liang et. al. [3] use Twitter API to collect twitter data. The training data falls in three different categories (camera, mobile phone and movie). The data is labeled as positive, negative and non-opinions instead of utilizing data that contains emoticons to identify the sentiment. This is because emoticons are not always consistent with the sentiment. Pre-processing is done on the considered tweets. In the next step, the tweets containing opinions are filtered. This is done by using Naive Bayes classifier on the training set. Unigram Naive Bayes model is implemented and the Naive Bayes simplifying independence assumption is employed. In the next step, useless features are eliminated by using the Mutual Information and Chi square feature extraction method. Next, the orientation of an opinion sentence is predicted. i.e. positive or negative. It is found that tweets are classified as opinionated and non-opinionated with a 76.8 %. Using feature selection, an accuracy of 96.6% is obtained. The test data set is classified with an accuracy of 90.17% with the focus on only positive and negative data. Another training set is generated using emoticons to denote positive or negative tweets. Using the emoticon trained data set, the tweets are classified with an accuracy of 58.65%. So, it was concluded that using the emoticons to collect training data is not always accurate.

Bowick et. al. [14] deals with distant supervision for topic identification of tweets and how this can be applied to analyzing Presidential Job Approval ratings from Twitter data.

1. The authors obtained seven months of tweets which contained around 476 million tweets labeled with usernames and timestamps, collected through the Twitter API.
2. Tweets were aligned with polling data using their timestamps.
3. Two trend lines for approval and disapproval ratings were created. The positive and negative sentiment scores were compared against these two trends.
4. Their results outperform previous work on Presidential Job Approval prediction (OConnor 2010). They presented two novel approaches to

the domain: a coupled distantly supervised system, and a topic-neutral baseline, both of which outperformed previous results.

Spencer et. al. [25] has presented a web based tool Sentimentor to classify live Twitter data into positive negative and objective tweets. It has an interface that allows users to easily analyze the word distributions and pictorial representation of the sentiments in tweets. Twitter API is used for data extraction process. The collected tweets were pre-processed. The POS tagging of each word was done and unigrams and bigrams were extracted. The Naive Bayes algorithm was used for classification with POS tags, unigrams and bigrams as features. It is found that the best accuracy of 52.31 % was obtained by using bigram without POS tagging. The use of POS tags has had a negative effect on the accuracy probably because of the use of summation of POS tags across a phrase rather than considering binary occurrences.

Barbosa et. al. [7] have presented effective sentiment detection approach for Tweets. They have used emoticons as noisy labels. The performance can be accredited to the following features: (1) An abstract representation of tweets is used instead of tweets as such and (2) the data source provides labels that are beneficial. The main disadvantage is when tweets have contradicting emotions. Modha et. al. [30] have developed a tool for automatic sentiment analysis for unstructured data. It talks about Big Data, its scope and challenges and opinion mining in big data. Instead of following the traditional approach of considering only subjective sentences and ignoring the objective ones, the paper considers both of them to be equally important. Two sets are considered - a document set $D = d_1, d_2, \dots, d_N$, and a sentence set $S = S_1, S_2, \dots, S_n$.

- In the first step, sentences or sentences of documents are classified into two categories Opinionated and Non Opinionated, regardless whether it is subjective or objective.
- In the second step, opinionated sentences are classified as subjective sentences and Objective sentences.
- In the third step, subjective sentences are classified into positive, negative or neutral sentences. For complex sentences, context and semantic orientation are attached. Finally, the fourth step involved classifying objective sentences into positive,negative or neutral.

Stephan Winkler et. al. [18] have presented an Ensemble modeling approach for sentimental analysis using Machine learning algorithms. The approach presented relies on analysis of words found in sentences and formation

of Heterogeneous models (i.e. Binary as well as Multi classification) that are calculated using machine learning methods. Applied machine learning techniques used are decision trees, random forest, neural networks and K-nearest neighbor and are used with boosting algorithm to increase the accuracy of prediction rule. For classifying sentiments based on positive and negative, Gaussian process is used. SVMs were used to select the best models from a set of models which can be increased by 60.4% and the ratio of wronged samples can be decreased up to 7.2% only if the threshold values is set to 1.0.

Tamilselvil et. al. [22] have considered basically two domains stock market and movie box revenue for which a decision tree is to be presented called summary tree. The study of sentiment analysis is done in three stages namely collection, cleaning and pre-processing. For the removal of irrelevant data, a probabilistic model called Latent Dirichlet Allocation is considered. Three datasets for stock market are considered. The stock market domain used the nonlinear models (SVMs and neural networks) along with sentiment indices did really predictions well whereas with movie box revenue with no sentiment indices did not do any specific predictions at all. A large volume of tweets along with SVMs are considered with other linear models to increase the forecasting of predictions.

Xiaolong Wang et. al. [24] have performed a linguistic analysis on collected data and discovered sentiments of it. A proposed sentiment classifier is built that is able to determine the sentiments in document. Here, basically the emoticons are used to indicate the users moods. They used SVMs and CRF learners to classify sentiments, SVM and Naive Bayes were able to obtain at least 70% accuracy together. Bayesian learning and turning opinion mining was used as models for sentiment classifiers. It is found that use of sentiment topics provided better predictions than semantic features. The sentiment topics provide accuracy up to 80.2% when compared with Naive Bayes used along with semantic topic features. It is concluded that sentiment topics give more accurate predictions even with less features.

Theresa et. al. [2] have observed that using a prior polarity alone for classifying contextual polarity of phrases gives an accuracy of only 48%. They analyzed that 76% of the errors resulted from using words with non-neutral prior polarity being used in neutral contextual polarity in the phrases. So, a two-step approach is proposed to classify such words. The first step used is a Neutral-Polar classifier which tries to determine if an instance is neutral or polar in context. Four types of learning algorithms, Boosting, Memory based learning, Rule learning and Support Vector learning are used. The performance of the neutral -polar classifier is compared based on two

baselines. The first baseline uses just word tokens. The second baseline uses both word tokens and polarity. The second step is the Polarity Classification where the polarity of all clues identified as polar in step one are classified.

Mandel et. al. [9] examined the response to the natural disaster Hurricane Irene on Twitter. They collected about 65,000 tweets over a span of 2 weeks and grouped them by location, gender in order to analyze them. Based on the level of concern, a sentiment classifier was trained to categorize the messages. First, the tweets were manually classified, then the classifier was trained on that data and then this classifier was used to label unlabeled data. The paper finds that the number of tweets from a particular region is directly proportional to the time hurricane hits that region and level of concern on the days leading up to the hurricanes arrival is dependent on the region.

Pak et. al. [12] presented a novel approach to collect a dataset with positive and negative sentiments, and a collection of objective texts (with no sentiments). This method allowed them to collect negative and positive sentiments such that no human effort was needed (and probably wont be needed) for classifying the documents. Objective texts were also collected automatically. A statistical linguistic analysis of the collected corpus was performed. The collected corpus was used to train a sentiment classifier. Experimental evaluations on a set of real microblogging posts were conducted to prove that this technique is efficient and performs better than previously proposed methods.

Luo et. al. [13] explains the challenges and an efficient technique to mine opinions from Twitter messages. Spam and wildly varying language makes opinion retrieval within Twitter challenging. They developed a unique ranking function which used social features and opinionated features of tweets for better opinion retrieval. They found out that their method optimized the BM25 baseline and the VSM baseline by improving MAP by 56.82% and 33.75% respectively. Their ranking model could achieve comparable performance with a method using manually tagged tweets. The experimental results showed their approach is still effective for opinion retrieval with TREC Tweets2011 dataset. Davidov et. al. [15] experimented with semi-supervised sarcasm identification on two different data sets consisting of 5.9 million tweets and 66000 product reviews from Amazon. A robust algorithm called SASI is used for recognition of sarcasm, to experiment with the tweets and reviews from Amazon dataset. Evaluating in various ways and with different parameters they achieved high precision on both datasets. The authors intend to design and develop a sarcasm recognition system for review ranking, summarization systems and brand monitoring systems. Sentiment

analysis requires a dictionary or lexicon for efficient classification of tweets.

Rao et. al. [16] discuss about building an emotional dictionary for efficient sentiment analysis of news on twitter. Each word in the news can be compared to the words in the emotional dictionary and then can be classified according to the sentiments. They presented various algorithms like Acc@1 algorithm and AP algorithm for constructing a word level and topic level emotional dictionary. This approach is different from previous methods because it is automatic, language independent and unlimited in volume. Three pruning strategies are developed for effective refining of the emotional dictionary. One drawback of their method was that it is less effective on news headlines.

Shawn OBanion et. al. [19] presented a novel approach in mining preference data from natural language experience in social media. The approach considered the voting decision of twitter users in 2012 US election. Statistical analyses model were used to predict election outcomes and campaign decision. The main goal was to know the preferences about the people who never expressed their opinions. All models were compared against 2 simple baselines. The first predicted Obama if the user used the #Obama2012 or #Romney2012, the second predicted if the user is friends with official @BrackObama twitter or @Romney. On the day of elections the baselines performed surprisingly well.

Pablo et. al. [5] presented a family of Naive Bayes classifiers for detecting the polarity of English tweets. Two different Naive Bayes classifiers have been built namely Baseline (trained to classify the tweets as positive, negative and neutral), and Binary (makes use of a polarity lexicon and classifies as positive and negative. Neutral tweets are not considered). The features considered by the classifiers are Lemmas (nouns, verbs, adjectives and adverbs), Multiword, and Polarity Lexicons from different sources and Valence Shifters. The training data set of tweets is obtained from SemEval Organization -2014 and additional annotated tweets from external sources. Many combinations of the above mentioned strategies and features are implemented. It is also concluded that performance is best when binary strategy is used with multiword and valence shifters features.

Marc Egger et. al. [17] have investigated about text based User - Generated-Content which is useful and relevant for corporate companies. They have considered the processes along three stages collection, analysis (positive, negative or neutral) and visualization, along with top down approach is used beginning with techniques operating on document and drilling down to lower levels of information extraction. Here topic modeling techniques such as Probabilistic Latent Semantic Analysis or Latent Dirichlet allocation are considered, which can be used to uncover abstract topics within

document. Construction of Decision trees is found to be more Complex for abbreviation dictionaries for this Nave Bayes can be used to detect the end of sentence. To map each word of text onto parts of speech tagger Distance based approach is used. Kumar et. al. [26] retrieve twitter data using Twitter API. They preprocess the tweets and add weightage according to the number of exclamation marks and the adjectives, verbs and adverbs are tagged in each tweet. Emoticons are replaced by their polarity. Adjectives and negative words are taken into account to calculate the polarity of the whole phrase. Polarity of the tweet is calculated based on a formula. The system proposed had characteristics of perceiving the sentiments in tweets.

Kouloumpis et. al. [27] performed sentiment analysis using Twitter hash-tags (e.g., #bestfeeling, #newphone, #androidwhat) to identify positive, negative, and neutral tweets. They used three different corpora of Twitter messages in their experiments - a hashtagged data set, an emoticon data set and a manually annotated data set ISIEVE. Their goal for these experiments was two -fold.

- First, they wanted to evaluate whether the training data with labels derived from hashtags and emoticons is useful for training sentiment classifiers for Twitter.
- Second, they wanted to evaluate the effectiveness of the tweets after pre-processing for sentiment analysis in Twitter data.

It was concluded that the experiments on twitter sentiment analysis show that part -of-speech features may not be useful for sentiment analysis in the microblogging domain. Davidov et. al. [28] proposed a supervised sentiment classification framework which used data from Twitter. By utilizing 50 Twitter tags and 15 smileys (:-), :-o) as sentiment labels, this framework avoids the need for labor intensive manual annotation, allowing identification and classification of diverse sentiment types of short texts. The paper also evaluated previously developed frameworks for sentiment classification and showed that their framework successfully identified sentiment types of sentences that were untagged. The quality of the sentiment identification was also confirmed by actual manual classification and cross validation. Dependencies and overlap between different sentiment types represented by smileys and Twitter hashtags was also explored. In this study, four different feature types (punctuation, words, n-grams and patterns) for sentiment classification were used and the contribution of each feature type for this task was evaluated.

Narret al., [29] examine a language independent sentiment classifier. The tweets in four languages, English, French, German and Portuguese are con-

sidered. Emoticons as labels are used to collect training data because they believed that in short messages like tweets, the emoticons are often consistent with the overall sentiment of the tweet. Nave Bayes classifier is used from NLTK Natural Language Processing Toolkit. 10000 tweets are used for testing which was manually annotated using the help Mechanical Turk workers into positive, negative, neutral or irrelevant. They used various combinations of the classifier. With Unigram classifier an accuracy of 81.3% was achieved in English tweets. Best accuracies observed for the four languages are: English - 81.3%, French - 74.9%, German - 79.8% and Portuguese - 64.9%. Krushikanth et. al. [10] intend to apply data mining tools to generate interesting patterns for predicting box office performance of movies using data collected from multiple social media and web sources including Twitter, YouTube and the IMDb movie databases. The box office predictions are based on the factors derived from a historical movie database, count of the Twitter followers and sentiment analysis of the comments of YouTube viewers. The movies are identified as Hit, Neutral or Flop using Wekas K-Means clustering tool specifically Wekas J48.

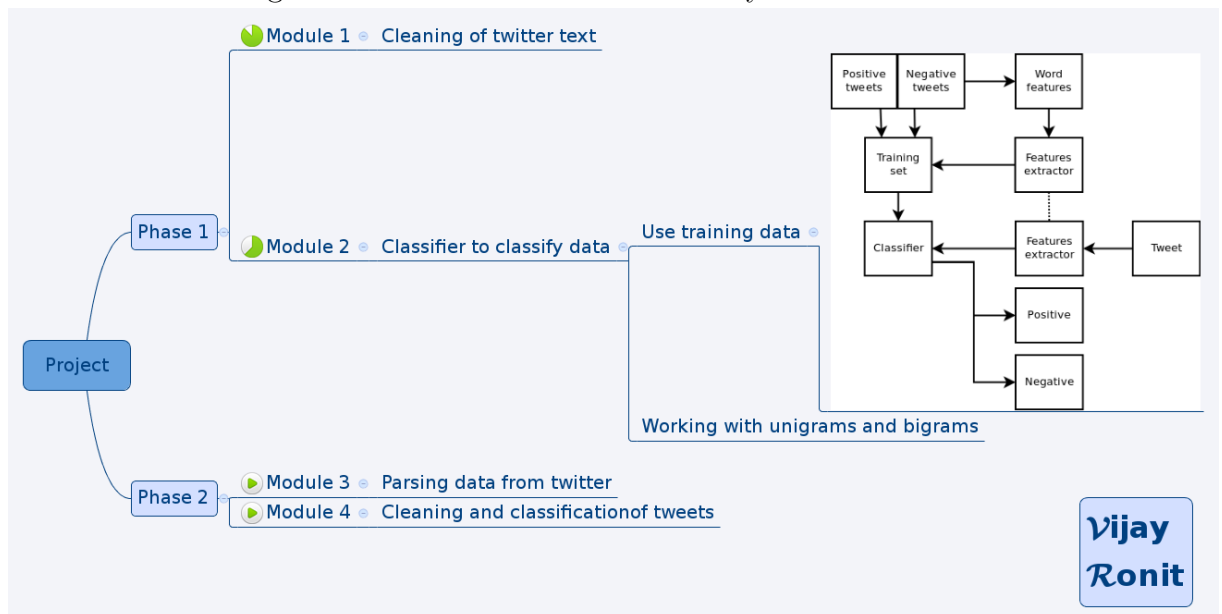
Burnap et. al. [11] built models that predicted information flow size and survival on Twitter following the terrorist event in Woolwich, London (2013). Information flow is the propagation of information over time posted on Twitter via the action of retweeting. After several trial and errors, they decided to use the zero -truncated negative binomial (ZTNB) regression method. Cox regression technique was used to model survival because it best estimates proportional hazard rates for independent measures. From about 427,330 tweets they analyzed that the sentiment expressed in the tweet is statistically predictive of both size and survival of information flows. Other press related information that the people were subjected to also was a significant predictor of size. Using the results from this paper, the authors intend to test the predictive efficiency of this model on other cases that exhibited similar characteristics.

Makazhanov et. al. study the problem of predicting the political preference of users on twitter and showing the predictions of users interacting with other parties. The paper evaluated the work on Alberta state elections and showed that model performed on the basis of F -measure and Sentiments. The goal is to predict which party, if any user is likely to vote for, for this a party is trained with binary classifier and it provides corresponding confidence value. The polarity of sentiments can be identified using Supervised and Unsupervised methods. They added a new feature to know about two kinds of users. It is predicted that logistic regression did better than SVMs when spammers were included who had lower F -measure.

Chapter 3

System Design

Figure 3.1: The Overall view of the sytem



3.1 Naive Bayes Classifier

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods. Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. An advantage of naive Bayes is that it only requires a small amount of training data to estimate the parameters necessary for classification. There are several variations in Nave Bayes classifier:

- Multinomial Nave Bayes - Used when Multiple Occurrences of Word Matter a lot in Text Classification problems. Such an example is when we try topic classification.
- Binarized Multinomial Nave Bayes - Used when frequencies of the words dont pay a key role in our classification. Such an example is Sentiment analysis where it doesnt matter how many times someone enters the word bad or good but rather only the fact that he does
- Bernoulli Nave Bayes - This is used when in our problem the absence of a particular word matters For example Bernoulli is commonly used in Spam or Adult Content Detection with very good results.

3.2 Implementation Details

We will be using Python (2.x or 3.x) along with the Natural Language Toolkit (nltk) and libsvm libraries to implement the classifiers. If you are using Ubuntu, you can get all of these with a single command as below.

```
sudo apt-get install python python-nltk python-libsvm python-yaml python-webpy python-oauth2
```

As we will describe in the next section, the main features of the model are lemmas extracted using lemmatization. Given that the language of microblogging requires a special treatment, we propose a pre-processing task to correct and normalize the tweets before lemmatizing them. The main preprocessing tasks we considered are the following:

1. Stop words - a, is, the, with etc. The full list of stop words can be found at [Stop Word List](#). These words don't indicate any sentiment and can be removed.
2. Removing urls, references to usernames, and hashtags.
3. Reduction of replicated characters (e.g. looooveeee love)

```
1 from cleaner import *
2 import csv
3 import pickle
4
5 featureList=[]
6 #start_extract_features
7 def extract_features(tweet):
8     global featureList
9     tweet_words = set(tweet)
10    features = {}
11    for word in featureList:
12        features['contains(%s)' % word] = (word in tweet_words)
13    return features
14 #end
```

Listing 3.1: Feature extractor

We import the module csv to work with large csv files and mainly for the csv.reader() function. The Pickle module allows us to save the trained classifier to a pickle file(dump) and gives us the capability to load the classifier back when we need it. This helps us to avoid retraining the classifier every time we run the application.

```
1 def testerFunc(testTweet):
2     f = open('final_classifier.pickle')
3     classifier = pickle.load(f)
4     f.close()
5
6     processedTestTweet = cleanerFunc(testTweet)
7     print testTweet+' , '+classifier.classify(extract_features(
8         processedTestTweet))
9
10 def main():
11     inpTweets = csv.reader(open('tester.csv', 'rb'), delimiter=',',
12                             ')
13     tweets=[]
14     global featureList
15     for row in inpTweets:
16         sentiment = row[0]
17         tweet = row[1]
18         processedTweet=cleanerFunc(tweet)
19
20         featureVector=processedTweet
21         featureList.extend(featureVector)
22         tweets.append((featureVector, sentiment))
23
24     # Remove featureList duplicates
25     featureList = list(set(featureList))
```



```
26     print tweets
27
28     training_set = nltk.classify.util.apply_features(
extract_features , tweets)
29     print training_set
30     #train the classifier
31     NBClassifier = nltk.NaiveBayesClassifier.train(training_set)
32
33     print NBClassifier.show_most_informative_features(30)
34
35     testTweet='horrible'
36     processedTestTweet = cleanerFunc(testTweet)
37     print processedTestTweet
38     print NBClassifier.classify(extract_features(
processedTestTweet))
39
40
41     f = open('final_classifier.pickle', 'wb')
42     pickle.dump(NBClassifier, f)
43     f.close()
```

Listing 3.2: TesterFunc and classifier.main()

Feature vector is the most important concept in implementing a classifier. A good feature vector directly determines how successful your classifier will be. The feature vector is used to build a model which the classifier learns from the training data and further can be used to classify previously unseen data.

In tweets, we can use the presence/absence of words that appear in tweet as features. In the training data, consisting of positive, negative and neutral tweets, we can split each tweet into words and add each word to the feature vector. Some of the words might not have any say in indicating the sentiment of a tweet and hence we can filter them out. Adding individual (single) words to the feature vector is referred to as 'unigrams' approach.

Some of the other feature vectors also add 'bi-grams' in combination with 'unigrams'. For example, 'not good' (bigram) completely changes the sentiment compared to adding 'not' and 'good' individually. Here, for simplicity, we will only consider the unigrams. Before adding the words to the feature vector, we need to preprocess them in order to filter, otherwise, the feature vector will explode.

The entire feature vector will be a combination of each of these feature words. For each tweet, if a feature word is present, we mark it as 1, else marked as 0. Instead of using presence/absence of feature word, you may also use the count of it, but since tweets are just 140 chars, I use 0/1. Now, you can think of each tweet as a bunch of 1s and 0s and based on this pattern, a tweet is labeled as positive, neutral or negative.

Given any new tweet, we need to extract the feature words as above and we get one more pattern of 0s and 1s and based on the model learned, the classifiers predict the tweet sentiment

```
1 import nltk
2 import re
3 import sys
4 from nltk.corpus import stopwords
5 from nltk.stem import WordNetLemmatizer
6
7 inputString="It was the best game of my life #CWC2015 #awesome #
  SAVsNZ "
8
9 def cleanerFunc(inputString):
10     #Identifying tags associated with the tweet
11     #print "\nThe tags associated with the tweet"
12     tags=re.findall(r'#[A-Za-z0-9]* ',inputString , flags=0)
13     #print tags
14
15     #remove tags
16
17     inputStringCleaned=re.sub(r'#[A-Za-z0-9]* ',",",inputString)
18     #print "\nString with hashtags removed : "+
19     inputStringCleaned
20
21     #Convert www.* or https?:/*
22     inputStringCleaned= re.sub(r'((www\.[^\s]+)|(https?://[^\s
23     ]+))',' ',inputStringCleaned)
24     #Convert @username
25     inputStringCleaned= re.sub(r'@[^\s]+',' ',inputStringCleaned)
26     #remove recurring letters
27     inputStringCleaned=re.sub(r'(\.|\1+','\1\1',
28     inputStringCleaned)
29
30     stopwordsSet=set(stopwords.words("english"))
31
32     LoweredString=inputStringCleaned.lower()
33     LoweredString=nltk.word_tokenize(LoweredString)
34
35     LoweredString=set(LoweredString)
36
37     NoStopwords=[w for w in LoweredString if not w in
```

```
stopwordsSet ]
35
36
37 #After removal of stopwords
38 #print "String after removal of stopwords: "+" ".join(
NoStopwords)
39
40 return NoStopwords
```

Listing 3.3: Cleaner.py

Stopwords are removed and a list of only the feature words is returned. We have used the python 're' module to handle the demands of preprocessing. Regular expressions are extremely useful in text processing tasks. NLTK is the python module called Natural Language ToolKit. In addition to numerous text-processing functions, It also contains a corpus of stopwords.

```
1 def oauth_req(self , url , http_method="GET" , post_body=None ,
2               http_headers=None):
3     config = self.parse_config()
4     consumer = oauth2.Consumer(key=config.get('consumer_key') ,
secret=config.get('consumer_secret'))
5     token = oauth2.Token(key=config.get('access_token') ,
secret=config.get('access_token_secret'))
6     client = oauth2.Client(consumer , token)
7
8     resp , content = client.request(
9         url ,
10        method=http_method ,
11        body=post_body or '' ,
12        headers=http_headers
13    )
14    return content
```

Listing 3.4: Main oauth request

This is the authentication module for the tweetFetcher program. To be able to access the twitter search API, you are first required to create a developer account at dev.twitter.com. The account is then used to create an app which provides you with 4 tokens.

- A Consumer Key
- A Consumer Secret Key
- An Access token
- An Access token secret

Chapter 4

Testing and result analysis

4.1 Training Corpus

In our preliminary experiments we have used the training dataset of tweets provided by Sentiment 140 organization (training.csv). This set contains 1,600,000 tweets, which were tagged with the following polarity values: Positive, Negative. In addition to fulfill the needs of the task, we also used a selection of annotated tweets (namely 5,050 positive and negative ones), which were compiled from an external source. We trained the classifier using different sets of tweets and tested it too.

4.2 Retrieving tweets for a particular topic

When you build a twitter sentiment analyzer, the input to your system will be a user enter keyword. Hence, one of the building blocks of this system will be to fetch tweets based on the keyword within a selected time duration.

The most important reference to achieve this is the Twitter API Documentation for Tweet Search. There are a lot of options that you can set in the API query and for the purpose of demonstrating the API, we have used only the simpler options.

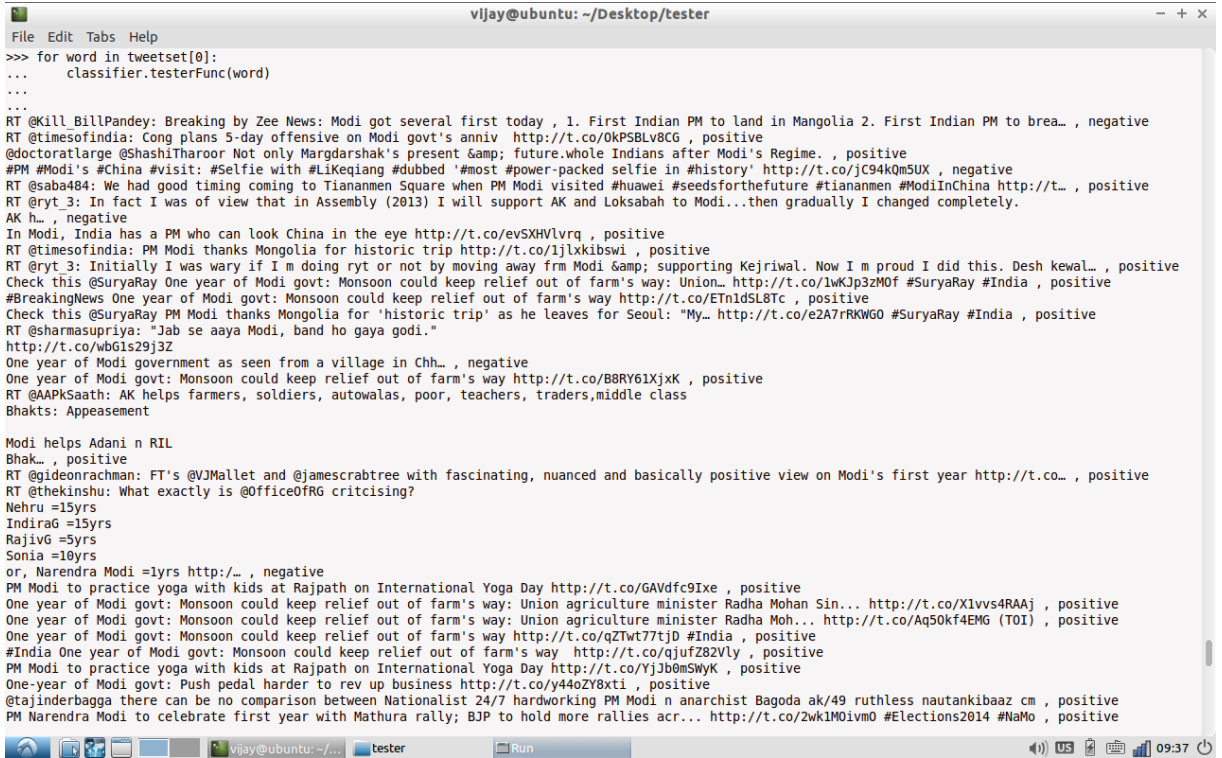
You can retrieve the tweets given a particular keyword. You need to specify config.json as defined below so that oauth requests can be made.

4.3 Most Informative Features

Table 4.1: Most informative features found after using 500 tweets for training

contains(:) = True	positi : negati = 2.2 : 1.0
contains(like) = True	positi : negati = 1.7 : 1.0
contains(-) = True	positi : negati = 1.7 : 1.0
contains(recommend) = True	positi : negati = 1.7 : 1.0
contains(frickin) = True	positi : negati = 1.7 : 1.0
contains(n't) = False	positi : negati = 1.0 : 1.0
contains(good) = False	negati : positi = 1.0 : 1.0
contains(love) = False	negati : positi = 1.0 : 1.0
contains(best) = False	positi : negati = 1.0 : 1.0
contains()) = False	positi : negati = 1.0 : 1.0
contains(;) = False	positi : negati = 1.0 : 1.0
contains(fallen) = False	positi : negati = 1.0 : 1.0
contains(:) = False	positi : negati = 1.0 : 1.0
contains(really) = False	positi : negati = 1.0 : 1.0
contains(cool) = False	positi : negati = 1.0 : 1.0
contains(fuming) = False	positi : negati = 1.0 : 1.0
contains(deserving) = False	positi : negati = 1.0 : 1.0
contains(bummed) = False	positi : negati = 1.0 : 1.0
contains(frothing) = False	positi : negati = 1.0 : 1.0
contains(behind) = False	positi : negati = 1.0 : 1.0

Figure 4.1: The system in action



```

vjay@ubuntu: ~/Desktop/tester
File Edit Tabs Help
>>> for word in tweetset[0]:
...     classifier.testFunc(word)
...
...
RT @Kill_BillPandey: Breaking by Zee News: Modi got several first today , 1. First Indian PM to land in Mangolia 2. First Indian PM to brea... , negative
RT @timesofindia: Cong plans 5-day offensive on Modi govt's anniv http://t.co/0kPSBLv8CG , positive
@doctoratlarge @ShashiTharoor Not only Margdarshak's present & future.whole Indians after Modi's Regime. , positive
#PM #Modi's #China #visit: #Selfie with #Likeqiang #dubbed '#most #power-packed selfie in #history' http://t.co/jC94kQm5UX , negative
RT @saba484: We had good timing coming to Tiananmen Square when PM Modi visited #huawei #seedsforthe future #tiananmen #ModiInChina http://t... , positive
RT @ryt_3: In fact I was of view that in Assembly (2013) I will support AK and Loksabab to Modi...then gradually I changed completely.
AK h... , negative
In Modi, India has a PM who can look China in the eye http://t.co/evSXHVlvrq , positive
RT @timesofindia: PM Modi thanks Mongolia for historic trip http://t.co/1jlxkibswi , positive
RT @ryt_3: Initially I was wary if I m doing ryt or not by moving away frm Modi & supporting Kejriwal. Now I m proud I did this. Desh kewal... , positive
Check this @SuryaRay One year of Modi govt: Monsoon could keep relief out of farm's way: Union... http://t.co/1wKJp3zMof #SuryaRay #India , positive
#BreakingNews One year of Modi govt: Monsoon could keep relief out of farm's way http://t.co/ETnldSL8Tc , positive
Check this @SuryaRay PM Modi thanks Mongolia for 'historic trip' as he leaves for Seoul: 'My... http://t.co/e2A7rRKWGO #SuryaRay #India , positive
RT @sharmasupriya: "Jab se aaya Modi, band ho gaya godi."
http://t.co/wbG1s29j3Z
One year of Modi government as seen from a village in Chh... , negative
One year of Modi govt: Monsoon could keep relief out of farm's way http://t.co/B8RY61XjxK , positive
RT @AAPKsaath: AK helps farmers, soldiers, autowalas, poor, teachers, traders,middle class
Bhakt's: Appeasement

Modi helps Adani n RIL
Bhak... , positive
RT @gideonrachman: FT's @VJMallet and @jamescrabtree with fascinating, nuanced and basically positive view on Modi's first year http://t.co... , positive
RT @thekinsu: What exactly is @officeOfRG criticising?
Nehru =15yrs
IndiraG =15yrs
RajivG =5yrs
Sonia =10yrs
or, Narendra Modi =1yrs http://... , negative
PM Modi to practice yoga with kids at Rajpath on International Yoga Day http://t.co/GAVdfc9Ixe , positive
One year of Modi govt: Monsoon could keep relief out of farm's way: Union agriculture minister Radha Mohan Sin... http://t.co/Xlvvs4RAAj , positive
One year of Modi govt: Monsoon could keep relief out of farm's way: Union agriculture minister Radha Moh... http://t.co/Aq50kf4EMG (TOI) , positive
One year of Modi govt: Monsoon could keep relief out of farm's way http://t.co/qZTwt77tjD #India , positive
#India One year of Modi govt: Monsoon could keep relief out of farm's way http://t.co/qjufZ82Vly , positive
PM Modi to practice yoga with kids at Rajpath on International Yoga Day http://t.co/YjJb0mSWyK , positive
One-year of Modi govt: Push pedal harder to rev up business http://t.co/y44oZY8xtl , positive
@tajinderbagga there can be no comparison between Nationalist 24/7 hardworking PM Modi n anarchist Bagoda ak/49 ruthless nautankibaa cm , positive
PM Narendra Modi to celebrate first year with Mathura rally; BJP to hold more rallies acr... http://t.co/2wk1M0ivm0 #Elections2014 #NaMo , positive

```

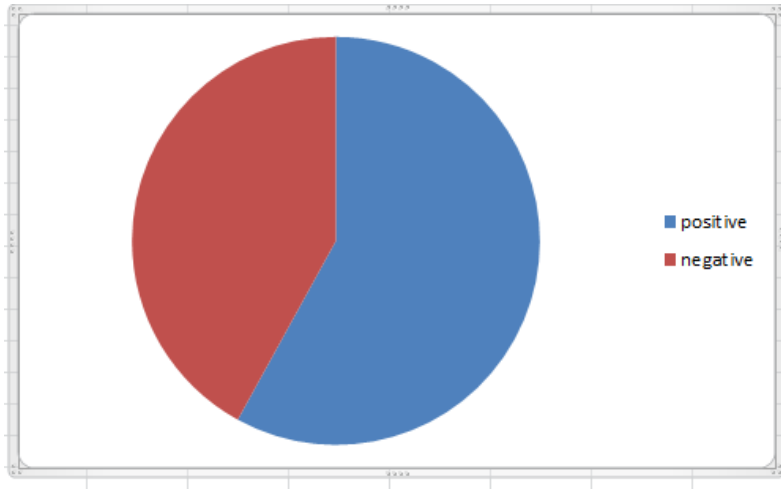
Table 4.2: Most informative features found after using 2000 tweets for training

contains(thanks) = True	positi : negati = 11.8 : 1.0
contains(lost) = True	negati : positi = 11.0 : 1.0
contains(sleep) = True	negati : positi = 8.2 : 1.0
contains(sad) = True	negati : positi = 7.9 : 1.0
contains(hate) = True	negati : positi = 6.7 : 1.0
contains(miss) = True	negati : positi = 6.6 : 1.0
contains(me.) = True	negati : positi = 6.3 : 1.0
contains(isn't) = True	negati : positi = 5.7 : 1.0
contains(man) = True	negati : positi = 5.7 : 1.0
contains(thats) = True	positi : negati = 5.7 : 1.0
contains(luck) = True	positi : negati = 5.7 : 1.0
contains(great) = True	positi : negati = 5.4 : 1.0
contains(bad) = True	negati : positi = 5.2 : 1.0
contains(bed) = True	negati : positi = 5.0 : 1.0
contains(feeling) = True	negati : positi = 5.0 : 1.0
contains(cry) = True	negati : positi = 5.0 : 1.0
contains(cant) = True	negati : positi = 5.0 : 1.0
contains(awesome) = True	positi : negati = 5.0 : 1.0
contains(!) = True	positi : negati = 5.0 : 1.0
contains(game) = True	negati : positi = 4.3 : 1.0
contains(heart) = True	negati : positi = 4.3 : 1.0
contains(shit) = True	negati : positi = 4.3 : 1.0
contains(killed) = True	negati : positi = 4.3 : 1.0
contains(cold) = True	negati : positi = 4.3 : 1.0
contains(today!) = True	negati : positi = 4.3 : 1.0
contains(hello) = True	positi : negati = 4.3 : 1.0
contains(eating) = True	positi : negati = 4.3 : 1.0
contains(beautiful) = True	positi : negati = 4.3 : 1.0
contains(thank) = True	positi : negati = 4.2 : 1.0
contains(best) = True	positi : negati = 4.1 : 1.0

Table 4.3: Most informative features found after using 10000 tweets for training

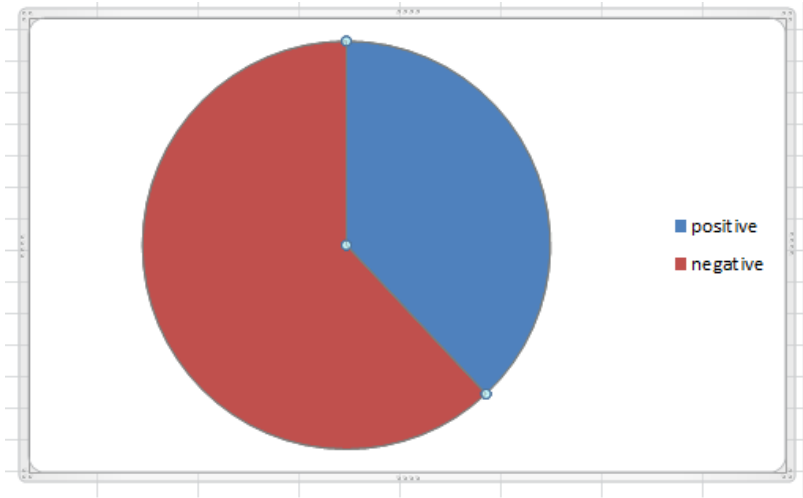
contains(jonas) = True	positi : negati = 25.7 : 1.0
contains(sad) = True	negati : positi = 16.7 : 1.0
contains(welcome) = True	positi : negati = 15.0 : 1.0
contains(cold) = True	negati : positi = 14.4 : 1.0
contains(brothers) = True	positi : negati = 14.3 : 1.0
contains(poor) = True	negati : positi = 12.4 : 1.0
contains(sleep.) = True	negati : positi = 12.2 : 1.0
contains(spring) = True	negati : positi = 11.7 : 1.0
contains(thanks!) = True	positi : negati = 10.6 : 1.0
contains(died) = True	negati : positi = 10.3 : 1.0
contains(xd) = True	positi : negati = 10.3 : 1.0
contains(hate) = True	negati : positi = 9.7 : 1.0
contains(season) = True	negati : positi = 9.7 : 1.0
contains(lost) = True	negati : positi = 9.4 : 1.0
contains(stupid) = True	negati : positi = 9.0 : 1.0
contains(concert) = True	positi : negati = 9.0 : 1.0
contains(stuck) = True	negati : positi = 8.7 : 1.0
contains(ugh) = True	negati : positi = 8.7 : 1.0
contains(hurts) = True	negati : positi = 8.6 : 1.0
contains(fuck) = True	negati : positi = 8.3 : 1.0
contains(traffic) = True	negati : positi = 8.3 : 1.0
contains(unfortunately) = True	negati : positi = 8.3 : 1.0
contains(hospital) = True	negati : positi = 8.3 : 1.0
contains(hills) = True	negati : positi = 8.3 : 1.0
contains(join) = True	positi : negati = 8.3 : 1.0
contains(green) = True	positi : negati = 8.3 : 1.0
contains(great!) = True	positi : negati = 8.3 : 1.0
contains(hurt) = True	negati : positi = 8.2 : 1.0
contains(pain) = True	negati : positi = 7.8 : 1.0
contains(killed) = True	negati : positi = 7.7 : 1.0

Figure 4.2: Distribution of sentiment for search keyword 'modi'



Out of 50 tweets retrieved from the twitter website, more than 50% were of positive sentiment. # Modi was trending at the time the sample was taken. We can enhance the accuracy of the sentiment distribution by obtaining larger data sets.

Figure 4.3: Distribution of sentiment for keyword ipl



For the keyword ipl, there are several negative tweets in the sample that we collected from the twitter website. One must be careful that, to get an accurate picture of sentiment for a particular object, person, place or thing, numerous factors need to be considered.

Chapter 5

Conclusion/Future Work

The Naive Bayes classifier is simple to implement and does not take a lot of time and resources to perform classification. On the other hand, classifiers such as Random Forest provide more accurate classification but at the expense of other resources (time, computing power). There are still many unsolved problems in sentiment analysis. If you're interested, you can help us by working on one of the problems below. You can also join the mailing list at [sentiment 140](#).

- **Building a classifier for subjective vs. objective tweets** We've focused mostly on classifying positive vs. negative correctly. We haven't looked at classifying tweets with sentiment vs. no sentiment very closely.
- **Handling negation.** Words like no, not, and never are difficult to handle properly. Although we have implemented a rudimentary algorithm to check negation, there is a lot of scope for improvement. Relevant papers:
 1. G. Councill, Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis.
 2. Potts, Christopher. 2010. On the negativity of negation.
- **Handling comparisons.** Our bag of words model doesn't handle comparisons very well. For example, in the phrase "Stanford is better than Berkeley", the tweet would be considered positive for both Stanford and Berkeley using our bag of words model because it doesn't take into account the relation towards "better".

- **The "aboutness" problem.** Given a tweet, automatically detect if the sentiment is towards an entity. Example: about the term [Google]: "I love Google." not about the term [Google]: "You should Google that." Relevant papers:

1. Target-dependent Twitter Sentiment Classification

- **Determine context switches.** Sometimes tweets contain two different ideas. It would be good to be able to segment these two different ideas out. Here's an example: "Just chomped my way through a massive apple, was pretty tasty. Now for work. Business revision."
- **Building an accurate parser for tweets.** Dependency parsers, like the Stanford Parser, doesn't handle ungrammatical text very well because they were trained on corpuses like the Wall Street Journal . It would be great to develop a parser that can handle informal text better.
- **Sarcasm detection.**
- **Topic classification for tweets.**
- **Tag clouds.** Given a list of positive and negative tweets, what are the most meaningful words to put in a tag cloud?
- Applying sentiment analysis to Facebook messages. Facebook messages don't have the same character limitations as Twitter, so it's unclear if our methodology would work on Facebook messages.
- **Internationalization.** We focus only on English sentences, but Twitter has many international users. It should be possible to use our approach to classify sentiment in other languages.

Chapter 6

References

- [1] Glivia A. R. Barbosa, Wagner Meira Jr, Ismael S. Silva, Raquel O. Prates, Mohammed J. Zaki, Adriano Veloso, Characterizing the Effectiveness of Twitter Hashtags to Detect and Track Online Population Sentiment, ACM SIGCHI Conference on Human Factors in Computing Systems, Juried Works-in-Progress, May 2012.
- [2] Theresa Wilson, Janyce Wiebe, Paul Hoffmann, Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis, Journal of Computational Linguistics archive, Volume 35, Issue 3, Sept 2009, pp 399-433, DOI:10.1162/coli.08-012-R1-06-90.
- [3] Po-Wei Liang, Bi-Ru Dai, Opinion Mining on Social Media Data, IEEE 14th International Conference on Mobile Data Management, Milan, Italy, June 3 - 6, 2013, pp 91-96, ISBN: 978-1-4673-6068-5
- [4] Alec Go, Richa Bayani, Lei Huang, Twitter Sentiment Classification Using Distant Supervision, Journal on Processing, 2009, pp1-6.
- [5] Pablo Gamallo, Marcos Garcia, Citius: A Naive-Bayes Strategy for Sentiment Analysis on English Tweets, 8th International Workshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland, Aug 23-24 2014, pp 171175.
- [6] Alexander Pak, Patrick Paroubek, Twitter as a Corpus for Sentiment Analysis and Opinion Mining, 7th conference on International Language Resources and Evaluation (LREC'10), Valletta, ISBN: 2-9517408-6-7, May 2010.
- [7] Luciano Barbosa, Junlan Feng, Robust Sentiment Detection on Twitter

from Biased and Noisy Data, 23rd International Conference on Computational Linguistics: Posters, Pages 36-44, 2010.

[8] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau, Sentiment Analysis of Twitter Data, Workshop on Languages in Social Media, 2011, pp 30-38, ISBN: 978-1-932432-96-1.

[9] Benjamin Mandel, Aron Culotta, John Boulahanis, Danielle Stark, Bonnie Lewis, Jeremy Rodrigue, A Demographic Analysis of Online Sentiment during Hurricane Irene, 2nd Workshop on Language in Social Media, pp 27-36, 2012.

[10] Krushikanth R. Apala, Merin Jose, Supreme Motnam, C.-C. Chan, Kathy J. Liszka, Federico de Gregorio, Prediction of Movies Box Office Performance Using Social Media, IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2013, pp 1209-1214 ISBN: 978-1-4503-2240-9, DOI: 10.1145/2492517. 2500232.

[11] Pete Burnap, Matthew L. Williams, Luke Sloan, Omer Rana, William Housley, Adam Edwards, Vincent Knight, Rob Procter, Alex Voss, Tweeting the terror: modelling the social media reaction to the Woolwich terrorist attack, Social Network analysis and Mining, Springer 2 014, DOI 10.1007/s13278-014-0206-4.

[12] Alexander Pak, Patrick Paroubek, Twitter as a Corpus for Sentiment Analysis and Opinion Mining, 7th conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, 2010, ISBN 2-9517408-6-7

[13] Zhunchen Luo, Miles Osborne, Ting Wang, An effective approach to tweets opinion retrieval, Springer Journal on World Wide Web, Dec 2013, DOI: 10.1007/s11280-013-0268-7.

[14] Micol Marchetti-Bowick, Nathaneal Chambers, Learning for Microblogs with Distant Supervision: Political forecasting with Twitter, 13th Conference of the European Chapter of the Association for Computational Linguistics, 2012, Pages 603-612, ISBN: 978-1-937284-19-0.

[15] Davidov, Tsur, Rappoport, Semi- Supervised Recognition of Sarcastic Sentences in Twitter and Amazon, 14th Conference on Computational Natural Language Learning, 2010, pp 107-116, ISBN: 978-1-932432-83-1.

- [16] Yanghui Rao, Jingsheng Lei, Liu Wenyin, Qing Li, Mingliang Chen, Building emotional dictionary for sentiment analysis of online news, *Journal on World Wide Web*, Volume 17, Number 4, pp 723-742, June 2014, DOI 10.1007/s11280-013-0221-9.
- [17] Marc Egger, Andre Lang, Brief tutorial on how to generate user generated content, *German Journal on Artificial Intelligence*, pp 53-60, Feb 2013.
- [18] Stephan Winkler, Susanne Schaller, Viktoria Dorfer, Michael Affenzeller, Gerald Petz, Micha Karpowicz, Data-based prediction of sentiments using heterogeneous model ensembles, *Journal on Soft Computing*, July 2014, pp 1-12.
- [19] Shawn OBanion, Larry Birnbaum Knight Lab, Using Explicit Linguistic Expressions of Preference in Social Media to Predict Voting Behavior, *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Niagara Falls, Canada, Aug. 25 - 28 2013, ISBN: 978-1-4503-2240-9, pp 207-214, <http://doi.ieeecomputersociety.org>.
- [20] Aibek Makazhanov, Davood Rafiei, Predicting Political Preference of Twitter Users, *International Conference on Advances in Social Networks Analysis and Mining*, Niagara Falls, Ontario, Canada, 25-28 August 2013.
- [21] Oshini Goonetilleke, Timos Sellis, Xiuzhen Zhang, Saket Sathe, Twitter analytics: A Big data management perspective, *ACM SIGKDD Explorations Newsletter - Special issue on big data archive*, Volume 16, Issue 1, June 2014, Pages 11-20, DOI: 10.1145/2674026. 2674029.
- [22] A. Tamilselvi, M. ParveenTaj, Sentiment Analysis of Micro blogs using Opinion Mining Classification Algorithm, *International Journal of Science and Research (IJSR)*, Volume 2, Issue 10, October 2013, pp 196-202, ISSN (Online): 2319-7064. A Study on Sentiment Analysis using Tweeter Data (IJIRST/ Volume 1 / Issue 9 / 037) All rights reserved by www.ijirst.org 218
- [23] Marta Arias, Argimiro Arratia, Ramon Xuriguera, Forecasting with Twitter Data, *ACM Transactions on Intelligent Systems and Technology (TIST) -Special Section on Intelligent Mobile Knowledge Discovery and Management Systems and Special Issue on Social Web Mining archive*, Volume 5, Issue 1, December 2013, Article No. 8, DOI: 10.1145/2542182.2542190.

- [24] Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, Topic Sentiment Analysis in Twitter: A Graph-based Hashtag Sentiment Classification Approach, 20th ACM international conference on Information and knowledge management, Oct 2012, pp 1031-1040, DOI:10.1145/2063576.2063726.
- [25] James Spencer, Gulden Uchyigit. Sentimentor: Sentiment Analysis of Twitter Data, The 1st International Workshop on Sentiment Discovery from Affective Data (SDAD 2012) pp 56-66, Bristol, UK.
- [26] Akshi Kumar, Teeja Mary Sebastian, Sentiment Analysis on Twitter, International Journal of Computer Science Issues, Volume 9, Issue 4, No 3, July 2012, pp 372-378, ISSN (Online): 1694-0814.
- [27] Efthymios Kouloumpis, Theresa Wilson, Johanna Moore, Twitter Sentiment Analysis: The Good, Bad and the OMG!, 5th International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011, pp 538-541.
- [28] Dmitry Davidov, Oren Tsur, Ari Rappoport, Enhanced Sentiment Learning Using twitter Hashtags and Smileys, 23rd International Conference on Computational Linguistics, Beijing, pp 241-249, Aug 2010.
- [29] Sascha Narr, Michael Hulfenhaus, Sahin Albayrak, Language-Independent Twitter Sentiment Analysis, KDML workshop on knowledge discovery, data mining and machine learning, Dortmund, Germany, Sep 2012.
- [30] Jalaj S Modha, Gayatri S Pandi, Sandip J Modha, Automatic Sentiment Analysis for Unstructured Data, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 12, Dec 2013, pp 91-97, ISSN: 2277 128X.

Appendix A

Appendix

Related Articles

- Twitter sentiment analysis using Python and NLTK (my inspiration for this tech post) <http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/>
- <http://www.sentiment140.com>
- Text Classification for Sentiment Analysis Naive Bayes Classifier — StreamHacker <http://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-naive-bayes-classifier/>
- Wikipedia- The Free Encuclopedia — <http://en.wikipedia.org>
- how to build a twitter sentiment analyzer ? — <http://ravikiranj.net>

TextBooks

- NLTK Textbook — <http://nltk.org/book>
- Text Processing with Python CookBook