**Twitter Sentiment Analysis**

Contents

[Abstract 3](#_Toc534034656)

[Introduction 3](#_Toc534034657)

[Tools and Methods 5](#_Toc534034658)

[Background Theory 5](#_Toc534034659)

[Machine Learning 5](#_Toc534034660)

[Natural Language Processing 12](#_Toc534034661)

[Tools 13](#_Toc534034662)

[Scikit-learn 13](#_Toc534034663)

[Pandas 14](#_Toc534034664)

[Problem Statement 14](#_Toc534034665)

[Related Work 15](#_Toc534034666)

[Dataset details 16](#_Toc534034667)

[Data Pre-processing 18](#_Toc534034668)

[Approach 20](#_Toc534034669)

[Metrics 21](#_Toc534034670)

[Results 23](#_Toc534034671)

[Conclusion 25](#_Toc534034672)

[References 25](#_Toc534034673)

# Abstract

Investigation of open data from internet-based expressions and opinions could yield fascinating outcomes and bits of knowledge into the universe of popular feelings about any item, administration or identity. The blast of Web 2.0 has prompted expanded action in Podcasting, Blogging, Tagging, Contributing to RSS, Social Bookmarking, and Social Networking. Subsequently there has been a sudden increase of enthusiasm for individuals to mine these tremendous assets of information for suppositions. Sentiment analysis or Opinion Mining is mining of sentiment polarities from online social media. In this project we will talk about a procedure which permits use and understanding of twitter information for sentiment analysis. We perform several steps of text pre-processing, and then experiment with multiple classification mechanisms. Using a dataset of 50000 tweets and TFIDF features, we comparison the accuracy obtained using various classifiers for this task. We find that linear SVMs provide us the best accuracy results among the various classifiers tried. Sentiment analysis classifier could be useful for many applications like market analysis of different features of a new product or public opinion for a new movie or speech by a political candidate.

# Introduction

With the recent growth of mobile information systems and the increased availability of smart phones, social media has become a large part of daily life in most societies. This development has entailed the creation of massive amounts of data: data which when analysed can be used to extract valuable information about a variety of subjects.

Sentiment analysis (SA), also known as opinion mining is the process of classifying the emotion conveyed by a text, for example as negative, positive or neutral. The data made available by social media has contributed to a burst of research activity within SA in recent times and a shift in the focus of the field towards this type of data. Information gained from applying SA to social media data has many potential usages, for instance, to help marketers evaluate the success of an ad campaign, to identify how different demographics have received a product release, to predict user behaviour, or to forecast election results.

A popular social medium is Twitter, a micro-blogging site that allows users to write textual entries of up to 140 characters, commonly referred to as tweets. As of June 2015, Twitter has over 302 million monthly active users according to their homepage, whereof approximately 88 % have their tweets freely readable. Additionally, over 84% of the users also have their location specified in their profiles [Beevolve, 2012], enabling the possibility of performing drill-down on geographic locations. Data created by Twitter is made available through Twitter’s API, and represents a realtime information stream of opinionated data. Tweets can be filtered both by location and the time they were published. This has paved the way for a new sub-field of SA: Twitter sentiment analysis (TSA).

Performing natural language processing on textual data from Twitter presents new challenges because of the informal nature of this data. Tweets often contain misspellings, and the constrictive limit of 140 characters encourages slang and abbreviations. Unconventional linguistic means are also used, such as capitalization or elongation of words to show emphasis. Additionally, tweets contain special features like emoticons and hashtags that may have an analytical value. Hashtags are labels used for search and categorization, and are included in the text prepended by a “#”. Emoticons are expressions of emotion, and can either be written as a string of characters e.g., “:-)”, or as a unicode symbol. Finally, if a tweet is a reply or is directed to another Twitter user, mentions can be used by prepending a username with “@”.

The linguistic phenomenon of negation, has been shown to play a significant role in SA. Councill et al. [2010] tested a sentiment classifier and found that including their negation classifier provided a 29.5 % improvement in F1 score when classifying positive sentiment, and an 11.4 % improvement when classifying negative sentiment. Kiritchenko et al. [2014] included a sophisticated solution for handling negated terms in their SemEval-2014 entry by creating tweet-specific sentiment lexica containing individual scores for terms in affirmative and negated contexts, but the state-of-the-art systems in TSA still employ a very simple solution for identifying which terms are negated, by marking as negated all words from a negation cue term to the next punctuation symbol.

Twitter as of now has three unique variants of APIs accessible, to be specific the REST API, the Search API, and the Streaming API. With the REST API, engineers can assemble status information and client data; the Search API enables designers to inquiry explicit Twitter content, while the Streaming API can gather Twitter content progressively. Additionally, engineers can blend those APIs to make their very own applications. Subsequently, conclusion examination appears having a solid fundament with the help of monstrous online information. In any case, those kinds of online information have a few imperfections that possibly upset the procedure of slant investigation.

The primary imperfection is that since individuals can unreservedly post their own substance, the nature of their sentiments can't be ensured. For instance, rather than imparting theme related insights, online spammers post spam on gatherings. Some spam are negligible by any stretch of the imagination, while others have immaterial conclusions otherwise called phony sentiments.

The second defect is that ground truth of such online information isn't constantly accessible. A ground truth is progressively like a tag of a specific feeling, showing whether the sentiment is certain, negative, or unbiased. The Stanford Sentiment 140 Tweet Corpus is one of the datasets that has ground truth and is likewise open accessible. The corpus contains 1.6 million machine-labeled Twitter messages.

Ventures discover this zone valuable to examine popular sentiment of their organization furthermore, items, or to examine consumer loyalty. Associations use this data to accumulate criticism about recently discharged items which supplements in enhancing further structure.

# Tools and Methods

This section contains a presentation of the background theory and technological tools relevant to this project, as well as the external data used in the conducted experiments.

## Background Theory

Sentiment analysis (SA) comprises many concepts common to the whole field of natural language processing in addition to many concepts from machine learning: the most relevant of these are described in this section.

### Machine Learning

Machine learning has become a cornerstone in the field of SA. Most wellperforming systems incorporate some form of supervised machine learning. Here, we give a description of several machine learning algorithms relevant to the current state-of-the-art in sentiment analysis.

#### Support Vector Machines

The Support Vector Machine (SVM) classification algorithm was formally described by Cortes and Vapnik [1995]. The algorithm considers data points based on their spatial location, and attempts to split the feature space into optimal class segments. This division of the feature space is referred to as training the machine. A trained SVM can then be used for classification of new examples by assigning them a class based on which segment of the feature space they are located in.

In its basic form, it is an algorithm for linear classification of binary problems. When dealing with two linearly separable classes, the feature space is divided into class segments by creating a hyperplane with the largest possible margin between the two classes. This margin maximization is the essential concept of SVMs. The closest data points of both classes, parallel to the vector defining the hyperplane, constitute the support vectors. These give the algorithm its name. The figure below shows the hyperplane and support vectors in a two dimensional linear classification problem.

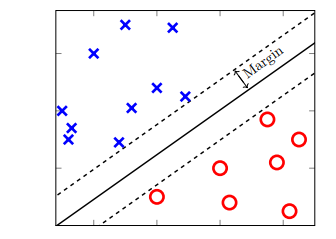


Figure: An SVM linearly separating two classes.

The algorithm can also be applied to problems where the examples are not linearly separable by allowing misclassified points. If no hyperplane exists to split all examples, the soft margin method introduces a slack variable that gives a penalty for each misclassified example [Cortes and Vapnik, 1995]. This penalty increases with the distance from the example’s support vector. The slack variable governs the trade-off between classification errors and margin size. To allow for non-linear classification, one can map the data into a higher dimensionality space by using the kernel trick. This is done by applying a kernel function to the data. The process is well explained in Fletcher [2009]. Popular kernel functions include radial basis function (RBF), Sigmoidal Kernel, and Polynomial Kernel. The figure below shows a data remapping done with an RBF kernel. Using the RBF kernel function, there are two parameters that require adjustment for good performance: the regularization parameter C and the influence radius parameter γ. C is a penalty for misclassification, and controls the complexity of the decision surface. A low C makes a complex hyperplane that attempts to correctly classify most of the training samples (increasing the chance of overfitting), while increasing the C regularizes the hyperplane, resulting in a more generalized classifier. The γ controls the radius of influence for each training example, in an inverse manner. This means that a low γ makes each training example have an influence over a larger area, and a higher γ will result in the training examples only influencing themselves. These two parameters are highly dependent on each other.

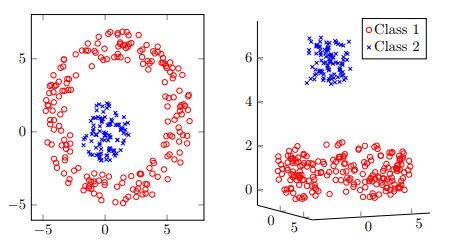


Figure: Dichotomous data remapped using an RBF kernel function

Additionally, the SVM algorithm can be applied to classification problems featuring more than two classes. Commonly used methods include one-against-one and one-against-the-rest. Hsu and Lin [2002] provide a thorough comparison of these methods.

#### Naïve Bayes Classifier

Naïve Bayes (NB) classifiers, also known as Naïve Bayes Learners, are a relatively simple group of probabilistic classifiers based on Bayes’ Theorem. For classification in certain domains, their performance has been shown to be comparable to much more complex machine learning algorithms, like neural networks or decision-tree learners [Mitchell, 1997]. Given a document d, the task of assigning a class c to the document is done by looking at the probabilities of each class given the document, and finding the maximum a posteriori (MAP) probability estimate:



The formula for each class can be rewritten using Bayes’ Theorem:



Because the probability of the document is the same across all classes, the class that maximises the numerator is the same as the class that maximises the whole expression. The denominator can then be dropped:



The document is represented as a vector of feature attributes:



This gives the approach used by the NB classifier:



The NB classifier assumes that all attribute values are conditionally independent. Despite this assumption, the classifier performs well on text classification tasks in practice.

There are several variants of Naive Bayes classifiers that are:

* The Multi­variate Bernoulli Model​: Also called binomial model, useful if our feature vectors are binary (e.g 0s and 1s). An application can be text classification with bag of words model where the 0s 1s are "word does not occur in the document" and "word occurs in the document" respectively.
* The Multinomial Model​: Typically used for discrete counts. In text classification, we extend the Bernoulli model further by counting the number of times a word $w\_i$ appears over the number of words rather than saying 0 or 1 if word occurs or not.
* The Gaussian Model​: We assume that features follow a normal distribution. Instead of discrete counts, we have continuous features

For text classification, the most used considered as the best choice is the Multinomial Naive Bayes.

The prior distribution P(c) can be used to incorporate additional assumptions about the relative frequencies of classes. It is computed by:



where N is the total number of training tweets and N is the number of training tweets in class c.

The likelihood P(w |c) is usually computed using the formula:



where count(w , c) is the number of times that word occurs within the training tweets of class j wj c , and |V | = ∑ the size of the vocabulary. This estimation uses the simplest smoothing j wj method to solve the zero­probability problem​, that arises when our model encounters a word seen in the test set but not in the training set, Laplace​or add­one since we use 1 as constant. We will see that Laplace smoothing method is not really effective compared to other smoothing methods used in language models.

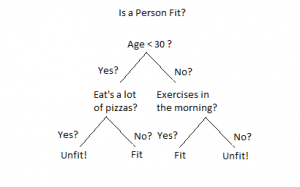
#### Logistic Regression

Logistic regression (multinomial logistic regression for more than two possible output values) is another probabilistic classification method. Based on the Principle of Maximum Entropy, it seeks the model that best represents the available data, which is the model with the maximum information entropy. Because of this, logistic regression is often called maximum entropy classification.

Compared to the NB classifier, logistic regression does not assume conditional independence among the features. This makes the classifier more suited to text classification problems, since the features consisting of words are not conditionally independent. The logistic regression classifier has been successfully applied to a wide range of text classification problems, including language detection, topic classification, and sentiment analysis [Vryniotis, 2013].

#### Decision Trees

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

[](https://www.xoriant.com/blog/wp-content/uploads/2017/08/Decision-Trees-modified-1.png)

An example of a decision tree can be explained using above binary tree. Let’s say you want to predict whether a person is fit given their information like age, eating habit, and physical activity, etc. The decision nodes here are questions like ‘What’s the age?’, ‘Does he exercise?’, ‘Does he eat a lot of pizzas’? And the leaves, which are outcomes like either ‘fit’, or ‘unfit’. In this case this was a binary classification problem (a yes no type problem).

To perform the splits, an impurity measure is used along with a gain measure. For example, information gain criteria chooses a split that maximizes the information gain. Information gain is computed as the entropy of the parent node minus the weighted entropy of the child nodes where the weight depends on the sizes of the child nodes. Also entropy is computed with respect to the class distribution. Other criteria are information gain ratio as well as gini gain. Popular decision tree algorithms include ID3, and CART.

#### Random Forests

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B: (1) Sample, with replacement, n training examples from X, Y; call these Xb, Yb. (2) Train a classification or regression tree fb on Xb, Yb.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x' or by taking the majority vote in the case of classification trees.

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated.

#### Boosted Trees

Boosting is a machine learning ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones. 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 weights are readjusted, known as "re-weighting". Misclassified input data gain a higher weight and examples that are classified correctly lose weight. Thus, future weak learners focus more on the examples that previous weak learners misclassified.

The main variation between many boosting algorithms is their method of weighting training data points and hypotheses. AdaBoost is very popular and the most significant historically as it was the first algorithm that could adapt to the weak learners.

AdaBoost can be used for face detection as an example of binary categorization. The two categories are faces versus background. The general algorithm is as follows:

1. Form a large set of simple features

2. Initialize weights for training images

3. For T rounds

a. Normalize the weights

b. For available features from the set, train a classifier using a single feature and evaluate the training error

c. Choose the classifier with the lowest error

d. Update the weights of the training images: increase if classified wrongly by this classifier, decrease if correctly

4. Form the final strong classifier as the linear combination of the T classifiers (coefficient larger if training error is small)

#### Nearest Neighbors

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

### Natural Language Processing

Natural language processing (NLP) is a field in the Human-Machine Interaction area concerned with the use of human natural languages for communication with computers. Among the many topics of NLP, the following are particularly relevant in this project.

Linguistic Negation

Linguistic negation is a grammatical concept that encompasses devices used to reverse the truth value of propositions in language. Givón [1993] defines two forms of grammatical negation: morphological negation, where individual words are negated with an affix, and syntactic negation, where a set of words is negated by a word or phrase. Negators in syntactical negation, known as negation cues or negation signals, function as operators, with an associated affected scope of words [Morante and Sporleder, 2012]. Syntactic negation is what is most relevant within NLP, as well as textual data mining in general, and is what we mean throughout this report when referring to negation.

Tottie [1991] provides an extensive study of negation in written English language, and splits syntactic negation — or clausal negation as she denotes it — into two main categories: rejections of suggestions and denials of assertions.

Polanyi and Zaenen [2006] describe valence as “positive or negative attitude communicated by a lexical item”. When looking at a segment of text, the segment’s valence can be equated to its sentimental orientation. In the context of SA, negators often function as valence shifters, because flipping the truth value of a proposition often also reverses, or significantly shifts, the valence it conveys. Valence shifters are terms that change the sentimental orientation of another set of terms, by changing the polarity and/or the evaluative intensity.

The most common linguistic negation cue in English is not, along with contractions created with it, such as couldn’t or isn’t [Tottie, 1991].

#### Bag-of-Words Model

A common way to represent text documents in a simplified manner is by using a bag-of-words model. The technique lists term occurrence and optionally the frequency of term occurrence, disregarding grammar and term order. Machine learning classifiers can use the resulting model directly as feature vectors.

#### Term Frequency-Inverse Document Frequency

Term frequency-inverse document frequency (TF-IDF) is a common term weighting scheme for the bag-of-words model, which lets us identify words in a collection of documents that can guide in deciding a document’s topic. A term will have a high TF-IDF score if it rarely appears in the whole corpus, but appears often in the document at hand. As a result, very common words such as “the”, “a”, and “is” in English will be weighted lower and have little impact, instead of shadowing rarer and more interesting terms [Manning et al., 2008].

TF-IDF is calculated as:



where tf(t, d) is the term frequency, and idf(t, D) is the inverse document frequency. There are several variants of both tf and idf, but in their simplest forms, tf is the number of times a term t occurs in a document d, and the idf is:



Here D is the entire corpus of documents, and N is the total number of documents in the corpus. As the number of documents where a term appears increases, the ratio inside the logarithm will decrease towards 1, making the idf approach 0.

## Tools

### Scikit-learn

Scikit-learn [Pedregosa et al., 2011] is a framework for the Python programming language that offers machine learning models as well as tools for performing preprocessing and data analysis. The Scikit-learn project is focused on providing state-of-the-art implementations for a wide range of machine learning methods, with particular attention to performance, consistent API and good documentation. The documentation is simplified in order to present to inexperienced readers the key points of a topic, while pointing experts to more in-depth information. The following highlights some key aspects of the framework.

### Pandas

Pandas [McKinney, 2012] is an open-source library providing high-performance data structures and data analysis tools for the Python programming language. It also includes tools for efficiently reading and writing data between in-memory data structures and different textual file formats, such as comma-separated value files.

# Problem Statement

Sentiment analysis, also refers as opinion mining, is a machine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive or negative. It is a really useful analysis since we could possibly determine the overall opinion about a selling objects, or predict stock markets for a given company like, if most people think positive about it, possibly its stock markets will increase, and so on. Sentiment analysis is actually far from to be solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar,...) but it is also why it is very interesting to working on.

In this project I choose to try to classify tweets from Twitter into “positive” or “negative” sentiment by building a model based on probabilities. Twitter is a microblogging website where people can share their feelings quickly and spontaneously by sending a tweets limited by 140 characters. You can directly address a tweet to someone by adding the target sign “@” or participate to a topic by adding an hastag “#” to your tweet. Because of the usage of Twitter, it is a perfect source of data to determine the current overall opinion about anything.

In summary, given a tweet, the aim is to build a system which can predict its sentiment polarity as positive or negative.

# Related Work

Sentiment analysis has been studied for a long time in the research community. While the simplest version of the problem is to classify a document as positive or negative, a more difficult version is to assign a rating, say on a scale of 1-10. Further, in some settings, finding the source and target of the sentiment is useful. In some other settings, fine grained moods like happy, peaceful, calm, etc. are more useful to discover from the documents. Typically researchers have used classification based approaches for sentiment analysis. There has been quite some work on extracting interesting feature values for the task, including simple frequency counts to TFIDF to scaled likelihood. There have also been efforts to build sentiment lexicons. Quite a few manually generated sentiment lexicons are publicly available. SentiWordNet is an automatically generated lexicon which is quite popular as well. There have also been semi-supervised approaches proposed for the sentiment analysis task to increase sentiment lexicon size. These include using simple heuristics like (a) words separated by “and” have same sentiment polarity while words separated by “but” have different sentiment polarity. (b) dictionaries and thesaurus can be used to include synonyms of positive words in positive set and antonyms of positive words in negative set, and (c) words frequently co-occurring with extremely positive words are positive, while words frequently cooccurring with extremely negative words are negative.

Recently there has also been work on extracting sentiment lexicons per domain, extracting sentiment carrying phrases, and also performing aspect based sentiment analysis. [Pang and Lee, 2008] and [Liu 2012] are good books on research done in the field of sentiment analysis. Further, we redirect the reader interested in reading more about previous work on sentiment analysis for Twitter data to [Alexander and Paroubek, 2010], [Kouloumpis et al., 2011], [Agarwal et al., 2011], and [Jørgen and Reitan, 2011].

# Dataset details

To gather the data many options are possible. Some people build a program to collect automatically a corpus of tweets based on two classes, “positive” and “negative”, by querying Twitter with two type of emoticons:

* Happy emoticons, such as “:)”, “:P”, “:­)” etc.
* Sad emoticons, such as “:(“, “:’(”, “=(“.

Others make their own dataset of tweets my collecting and annotating them manually which very long and fastidious.

Additionally to find a way of getting a corpus of tweets, we need to take a balanced data set, meaning we should have an equal number of positive and negative tweets, but it needs also to be large enough. Indeed, more the data we have, more we can train our classifier and more the accuracy will be.

After many researches, I found a dataset of 1600000 tweets in English coming from two sources: Kaggle and Sentiment140. It is composed of various columns that are ItemID, Sentiment, date, user and SentimentText. We are only interested in the Sentiment column corresponding to our label class taking a binary value, 0 if the tweet is negative, 4 if the tweet is positive and the SentimentText columns containing the tweets in a raw format.

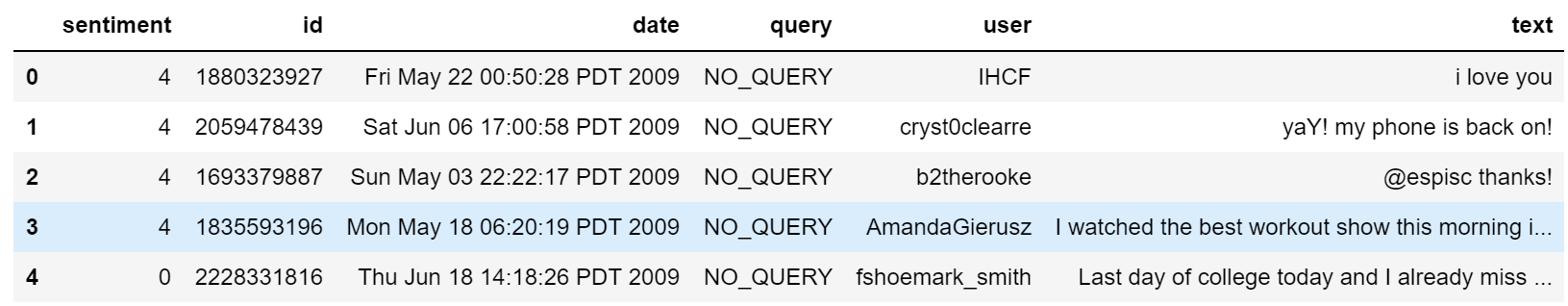


Table: Example of twitter posts annotated with their corresponding sentiment, 0 if it is negative, 4 if it is positive.

In the Table above, we show the first five twitter posts. We can already notice some particularities and difficulties that we are going to encounter during the pre-processing steps.

● The presence of acronyms​"bf" or more complicated "APL". Does it means apple ? Apple (the company) ?

● The presence of sequences of repeated characters​such as "Juuuuuuuuuuuuuuuuussssst", "hmmmm". In general when we repeat several characters in a word, it is to emphasize it, to increase its impact.

● The presence of emoticons​, ":O", "T\_T", ":­|" and much more, give insights about user's moods.

● Spelling mistakes ​and “urban grammar​” like "im gunna" or "mi".

● The presence of nouns​ such as "TV", "New Moon".

Furthermore, we can also add,

● People also indicate their moods, emotions, states, between two such as, \*\cries\*, \*hummin\*, \*sigh\*.

● The negation, “can't”, “cannot”, “don't”, “haven't” that we need to handle like: “I don’t like chocolate”, “like” in this case is negative.

We could also be interested in the grammar structure of the tweets, or if a tweet is subjective/objective and so on. As you can see, it is extremely complex ​to deal with languages and even more when we want to analyse text typed by users on the Internet because people don’t take care of making sentences that are grammatically correct and use a ton of acronyms and words that are more or less english in our case.

We have exactly 800000 positive tweets and 800000 negative tweets which signify that the dataset is well ­balanced. There are also no duplicates.

Finally, let’s recall the Twitter terminology since we are going to have to deal with in the tweets:

● Hashtag: A hashtag is any word or phrase immediately preceded by the # symbol. When you click on a hashtag, you’ll see other Tweets containing the same keyword or topic.

● @username: A username is how you’re identified on Twitter, and is always preceded immediately by the @ symbol. For instance, Katy Perry is @katyperry.

● MT: Similar to RT (Retweet), an abbreviation for “Modified Tweet.” Placed before the Retweeted text when users manually retweet a message with modifications, for example shortening a Tweet.

● Retweet: RT, A Tweet that you forward to your followers is known as a Retweet. Often used to pass along news or other valuable discoveries on Twitter, Retweets always retain original attribution.

● Emoticons: Composed using punctuation and letters, they are used to express emotions concisely, ";) :) ...".

Now we have the corpus of tweets, we need to use other resources to make easier the pre­processing step.

Note that while the dataset is huge, such a large dataset requires a lot of processing power when one wants to build complex classifiers like boosting and random forests, specifically. Hence, we obtained a random sample of 50000 tweets from this original dataset, with 24864 negative tweets and 25136 positive tweets.

# Data Pre-processing

In this project, we perform the following pre-processing steps.

1. Cleaning up the HTML entities, and any other HTML tags etc. Sometimes some HTML entities may be present in the text. E.g., &lt; for < etc. We use BeautifulSoup Python library to get rid of such HTML.

HTML entities are characters reserved in HTML. We need to decode them in order to have characters entities to make them understandable.



Figure: A tweet before processing HTML entities.



Figure: A tweet after processing HTML entities.

1. Remove @mentions from the text. The @mentions correspond to usernames in twitter preceded by “@” symbol. It is used to address a tweet to someone or just grab the attention. But they are not very important for sentiment analysis.

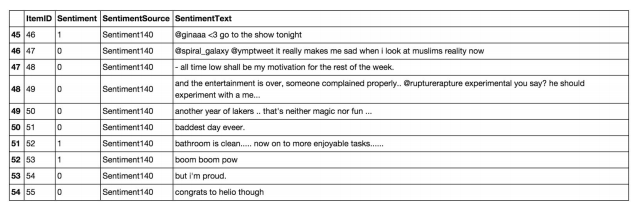


Table: Tweets before processing @mentions.

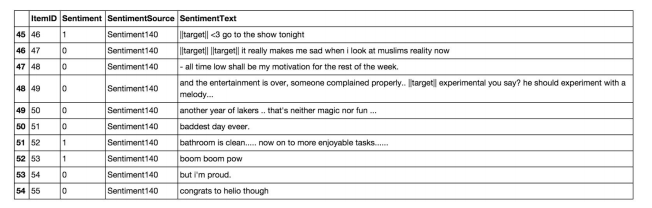


Table: Tweets after processing @mentions.

1. Remove URLs from tweets.

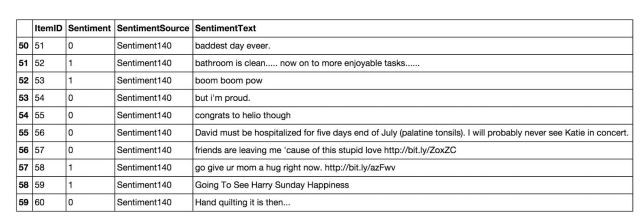


Table: Tweets before processing URLs.

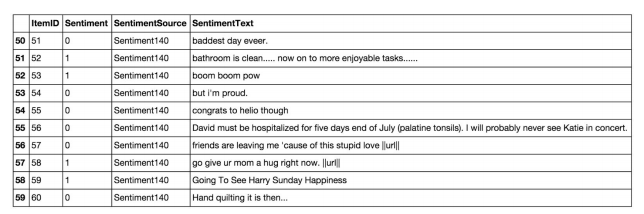


Table: Tweets after processing URLs.

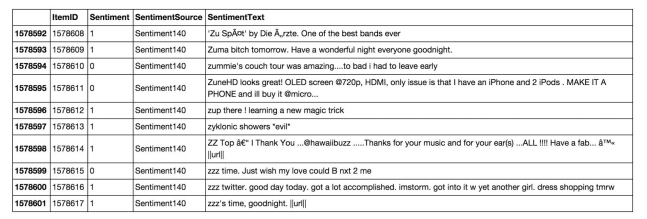
1. Convert short forms of negation words to their full forms. "isn't":"is not", "aren't":"are not", "wasn't":"was not", "weren't":"were not", "haven't":"have not", "hasn't":"has not", "hadn't":"had not", "won't":"will not", "wouldn't":"would not", "don't":"do not", "doesn't":"does not", "didn't":"did not", "can't":"can not", "couldn't":"could not", "shouldn't":"should not", "mightn't":"might not", "mustn't":"must not"
2. Decode UTF8 encoded symbols.  
   

Table: Tweets before processing Unicode

1. Replace non-alphabetic characters to spaces.
2. Ignore words of size 1.
3. Remove punctuations
4. Perform word lemmatization: Lemmatization is the reverse process of making morphological changes to a word. It is the process of getting to the root form of the word which exists in the dictionary.

# Approach

Once we have applied the different steps of the preprocessing part, we can now focus on the machine learning part.

We start by splitting the data randomly into two parts: train and test. We use 80% data for training and the remaining for testing.

We extract TFIDF features from tweets using TFIDF vectorizer. It converts a collection of raw documents to a matrix of TF-IDF features.

* TF(w) = (Number of times term w appears in a document) / (Total number of terms in the document)
* IDF(w) = log\_e(Total number of documents / Number of documents with term w in it)

Consider a document containing 100 words wherein the word 'Cauvery' appears 3 times.

1. The term frequency (tf) for 'Cauvery' is then TF = (3 / 100) = 0.03.
2. Now, assume we have 10 million documents and the word 'Cauvery' appears in 1000 of these. Then, the inverse document frequency (idf) is calculated as IDF = log(10,000,000 / 1,000) = 4.
3. Thus, the Tf-idf weight is the product of these quantities TF-IDF = 0.03 \* 4 = 0.12.

In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

Further, we use these features to learn multiple classifiers. For each type of classifier, we first train the classifier on train data and report the accuracy on test data. We also report cross validation accuracy with 10 folds.

The K­fold cross ­validation​: We split the data set into k parts, hold out one, combine the others and train on them, then validate against the held­out portion. We repeat that process k times (each fold), holding out a different portion each time. Then we average the score measured for each fold to get a more accurate estimation of our model's performance.

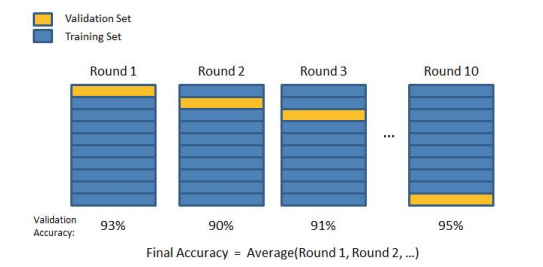
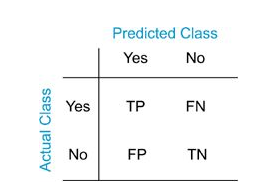


Figure: 10­-fold cross-­validation

# Metrics

A confusion matrix​ helps to visualize how the model did during the classification and evaluate its accuracy.



We use the typical definition of accuracy to score our models. Accuracy is defined as the fraction of true positives+true negatives with respect to the total number of instances.

# Results

|  |  |  |
| --- | --- | --- |
| **Classifier** | **Test Acc.** | **Cross Validation Acc.** |
| Naïve Bayes | 0.7623 | 0.7524 |
| Logistic Regression | 0.7776 | 0.7757 |
| Linear SVM | 0.7867 | 0.7842 |
| Decision Trees | 0.5663 | 0.5758 |
| Boosted Trees | 0.7284 | 0.7161 |
| Random Forests | 0.7637 | 0.7609 |
| Nearest Neighbors | 0.5347 | 0.5163 |

**Table: Accuracy Comparison across various Classifiers**

The above table shows the accuracy comparison across various classifiers both when tested using the train/test split method and using the 10-fold cross validation approach.

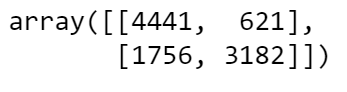
Note that we obtain the maximum accuracy using Linear SVM. However, also notice that logistic regression and random forests also lead to similar accuracy values. However, simple classifiers like decision trees and nearest neighbors somehow cannot provide good results at all.

We used default hyper parameter settings for all the classifiers as provided by scikit learn. Tuning those hyper parameters can lead to further accuracy boost and we plan to explore that as part of future work. For example, we could try logistic regression without any regularization or try say L1 regularization. For SVMs, we could try SVMs with various kinds of kernels or also vary the complexity parameter C and see the impact. For decision trees, we could try different levels of pruning. For random forests, we experimented with 100 trees, but we could vary the number of trees and check its impact on accuracy. Lastly, for nearest neighbors, we used k=5 and clearly changing the value of K should help us tune it better to get better accuracy values.

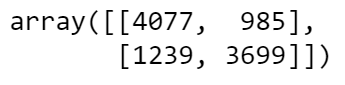
In the following, we show the distribution of true positives, false positives, true negatives and false negatives as we see across different classifiers. A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class. A false positive is an outcome where the model incorrectly predicts the positive class. And a false negative is an outcome where the model incorrectly predicts the negative class.

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). Hence, we present the information in the form of confusion matrices.

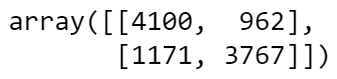
Naive Bayes:



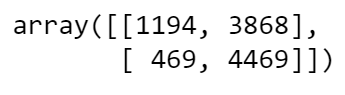
Logistic Regression:



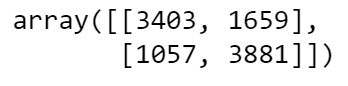
Linear SVM



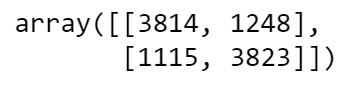
Decision Trees:



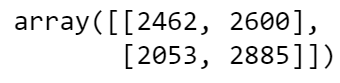
Boosted Trees:



Random Forests:



Nearest Neighbors:



# Conclusion

Sentiment analysis or opinion mining is a hot topic in machine learning. We are still far to detect the sentiments of s corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese. In this project we tried to show the basic way of classifying tweets into positive or negative category using various popular classification methods. We could further improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the classifiers, or trying more classifiers like deep learning architectures.

# References

* Beevolve. (2012, October 10). An exhaustive study of Twitter users across the world. Retrieved December 18, 2014, from http://www.beevolve.com/twitter-statistics/#c1
* Councill, I. G., McDonald, R. & Velikovich, L. (2010). What’s great and what’s not: learning to classify the scope of negation for improved sentiment analysis. In Proceedings of the workshop on negation and speculation in natural language processing (pp. 51–59). NeSp-NLP ’10. Uppsala, Sweden: Association for Computational Linguistics.
* Kiritchenko, S., Zhu, X. & Mohammad, S. M. (2014). Sentiment analysis of short informal texts. Journal of Artificial Intelligence Research, 50, 723–762.
* Cortes, C. & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20 (3), 273–297.
* Fletcher, T. (2009). Support vector machines explained, tutorial paper. http://www.tristanfletcher.co.uk/SVM%20Explained.pdf. Retrieved June 11, 2015.
* Hsu, C.-W. & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. IEEE Transactions on Neural Networks, 13 (2), 415–425.
* Mitchell, T. M. (1997). Machine learning (1st ed.). New York, NY, USA: McGraw-Hill, Inc.
* Vryniotis, V. (2013, November 20). Machine learning tutorial: the max entropy text classifier. Retrieved December 16, 2014, from http: //blog.datumbox.com/machine-learning-tutorial-the-maxentropy- text-classifier/
* Givón, T. (1993). English grammar: a function-based introduction. John Benjamins Publishing.
* Morante, R. & Sporleder, C. (2012). Modality and negation: an introduction to the special issue. Computational Linguistics, 38 (2), 223–260.
* Tottie, G. (1991). Negation in English speech and writing: a study in variation. Academic Press.
* Polanyi, L. & Zaenen, A. (2006). Contextual valence shifters (J. G. Shanahan, Y. Qu & J. Wiebe, Eds.). Springer.
* Manning, C. D., Raghavan, P. & Schütze, H. (2008). Introduction to information retrieval. Cambridge, UK: Cambridge University Press.
* Marcus, M. P., Marcinkiewicz, M. A. & Santorini, B. (1993). Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics, 19 (2), 313–330.
* Gimpel, K., Schneider, N., O’Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J. & Smith, N. A. (2011). Part-of-speech tagging for Twitter: annotation, features, and experiments. In Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies: short papers (Vol. 2, pp. 42–47). Association for Computational Linguistics. Portland, Oregon.
* Collins, M. (2003). Head-driven statistical models for natural language parsing. Computational Linguistics, 29 (4), 589–637.
* Hudson, R. (2010). An encyclopedia of word grammar and English grammar. Retrieved December 16, 2014, from http://tinyurl.com/wgencyc
* McKinney, W. (2012). Pandas: a Python data analysis library. Retrieved May 13, 2015, from http://pandas.pydata.org
* Kong, L., Schneider, N., Swayamdipta, S., Bhatia, A., Dyer, C. & Smith, N. (2014). A dependency parser for tweets. In Proceedings of the conference on empirical methods in natural language processing (pp. 1001– 1012). EMNLP ’14. Doha, Qatar.
* Owoputi, O., O’Connor, B., Dyer, C., Gimpel, K., Schneider, N. & Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In Proceedings of the 2013 conference of the North American Chapter of the Association for Computational Linguistics: human language technologies (pp. 380–390). Atlanta, GA, USA.
* Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
* Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." *Foundations and Trends® in Information Retrieval* 2, no. 1–2 (2008): 1-135.
* Liu, Bing. "Sentiment analysis and opinion mining." *Synthesis lectures on human language technologies* 5, no. 1 (2012): 1-167.
* Pak, Alexander, and Patrick Paroubek. "Twitter as a corpus for sentiment analysis and opinion mining." In *LREc*, vol. 10, no. 2010, pp. 1320-1326. 2010.
* Kouloumpis, Efthymios, Theresa Wilson, and Johanna D. Moore. "Twitter sentiment analysis: The good the bad and the omg!." *Icwsm* 11, no. 538-541 (2011): 164.
* Agarwal, Apoorv, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. "Sentiment analysis of twitter data." In *Proceedings of the workshop on languages in social media*, pp. 30-38. Association for Computational Linguistics, 2011.
* Faret, Jørgen, and Johan Reitan. "Twitter sentiment analysis." (2011).