Report on

Twitter Sentiment Analysis

M Meher Vishal 315126510103

MTA-IIT and Verzeo

Contents:

Abstract

This project addresses the problem of sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive or negative. As the data being generated is growing rapidly at a scale petabytes per day in various forms one of the main sources that generates such large data is social media platforms with over tens of millions of people active per day, if only we could use this data to extract useful information for analyzing the current business needs, their reach and customer satisfaction towards the product and the company. It could contribute to meet constantly changing requirements and also analyze other competitors' performance and change one's business strategies accordingly to be on the top. In this project, we are going to take data generated by users of one of the top microblogging websites Twitter, which has over 100 million daily active users and we are going to implement sentiment analysis on the tweets. This paper produces the output in the form of graphical representation of various tweets describing the total sentiment score of the tweets and as well as it also produces the individual score of each tweet.

1. Introduction

R is a popular programming language which is generally embraced by information researchers. In any case, normal R must be executed in a solitary machine environment. As the volume of accessible information proceeds to quickly develop from an assortment of sources, versatile and execution investigation arrangements have turned into a fundamental device to upgrade business profitability and income. Existing information examination situations, for example, R, are compelled by the span of the fundamental memory and can't scale in numerous applications. Information representation is turning into an undeniably imperative part of investigative in the time of huge information.

The steps required for analyzing data are:

• The need for Meeting speed

• Understanding the various types of data

• Addressing the data quality

• Display the correct understandable results

The R environment provides enormous built-in functions in the package “base”, most of which are generally required for elementary data analysis (e.g., linear modeling, graph plotting, basic statistics). However, the real beauty of R is its almost versatility and infinite expandability. Approximately 2,500 packages have been developed for R by the active R community. These bundles serve to enlarge R's regular abilities in information examination and frequently concentrate on improvements in different scientific fields, and additionally systems utilized as a part of very specific information investigations. Sentiment analysis: It is likewise alluded to as the Opinion Mining, Which suggests separating different feelings, feelings and estimations in content. As you can envision, a standout amongst the most well known uses of opinion examination is to track mentalities and emotions on the web, particularly to tack items, administrations, marks or even individuals. An principle thought is figure out if are emphatically or adversely by a given gathering of people. The reason for Text Mining is to prepare literary data, extricate significant numeric records from the content, and, in this way, make the data contained in the content open to the different information mining calculations.

Data can be separated to determine synopses for the words contained in the archives or to process rundowns for the reports in view of the words contained in them. Henceforth, you can investigate words, bunches of words utilized as a part of archives, or you could break down records and decide likenesses between them or how they are identified with different factors of enthusiasm for the information mining venture. In the most broad terms, content mining turns

"content into number" which can then be joined in different examinations. Utilizations of content Mining are breaking down open-finished study reactions, programmed preparing of messages, messages, and so on., dissecting guarantee or protection claims, demonstrative meetings, and so forth., researching contenders by slithering their sites.

The packages used in this project are:

• twitteR :It is an interface to access Twitter API.

• plyr: This package is a collection of tools which can solve general set of problems like when we need to break down a large problem to various pieces and each piece is operated separately and then all the pieces put back together.

• Stringr: It is a collection of simple wrappers which help us in doing basic operations on strings like removing special characteristics , converting uppercase alphabets to lower case alphabet and by not considering spaces.

• ggplot2 : Used to plot graphics using grammar in R. Each plot can be build up step by step from various different sources

Domain Introduction

This project of analyzing sentiments of tweets comes under the domain of “Pattern Classification” and “Data Mining”. Both of these terms are very closely related and intertwined, and they can be formally defined as the process of discovering “useful” patterns in large set of data, either automatically (unsupervised) or semi automatically (supervised). The project would heavily rely on techniques of “Natural Language Processing” in extracting significant patterns and features from the large data set of tweets and on “Machine Learning” techniques for accurately classifying individual unlabelled data samples (tweets) accord ing to whichever pattern model best describes them. The features that can be used for modeling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word “excellent” has a strong positive connotation while the word “evil” possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment. Parts of Speech tagging, on the other hand, is a syntactical approach to the problem. It means to automatically identify which part of speech

each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb, interjection, etc. Patterns can be extracted from analyzing the frequency distribution of these parts of speech (either individually or collectively with some other part of speech) in a particular class of labeled tweets. Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word lengthening, question marks, presence of url in tweets, exclamation marks, internet emoticons and internet shorthand/slangs. Classification techniques can also be divided into a two categories: Supervised vs. unsupervised and non-adaptive vs. adaptive/reinforcement techniques. Supervised approach is when we have pre-labeled data samples available and we use them to train our classifier. Training the classifier means to use the pre-labeled to extract features that best model the patterns and differences between each of the individual classes, and then classifying an unlabeled data sample according to whichever pattern best describes it. For example if we come up with a highly simplified model that neutral tweets contain 0.3 exclamation marks per tweet on average while sentiment -bearing tweets contain 0.8, and if the tweet we have to classify does contain 1 exclamation mark then (ignoring all other possible features) the tweet would be classified as subjective, since 1 exclamation mark is closer to the model of 0.8 exclamation marks. Unsupervised classification is when we do not have any labeled data for training. In addition to this adaptive classification techniques deal with feedback from the environment. In our case feedback from the environment can be in form of a human telling the classifier whether it has done a good or poor job in classifying a particular tweet and the classifier needs to learn from this feedback. There are two further types of adaptive techniques: Passive and active. Passive techniques are the ones which use the feedback only to learn about the environment (in this case this could mean improving our models for tweets belonging to each of the three classes) but not using this improved learning in our current classification algorithm, while the active approach continuously keeps changing its classification algorithm according to what it learns at real-time. There are several metrics proposed for computing and comparing the results of our experiments. Some of the most popular metrics include: Precision, Recall,Accuracy, F1-measure, True rate and False alarm rate (each of these metrics is calculated individually for each class and then averaged for the overall classifier performance.)

Sentiment Analysis

Sentiment analysis is a method for gauging opinions of individuals or groups, such as a segment of a brand's audience or an individual customer in communication with a customer support representative. Based on a scoring mechanism, sentiment analysis monitors conversations and evaluates language and voice inflections to quantify attitudes, opinions, and emotions related to a business, product or service, or topic. Sentiment analysis is sometimes also referred to as opinion mining. As part of the overall speech analytics system, sentiment analysis is the integral component that determines a customer's opinions or attitudes.

How Sentiment Analysis Works

Sentiment analysis is often driven by an algorithm, scoring the words used along with voice inflections that can indicate a person's underlying feelings about the topic of a discussion. Sentiment analysis allows for a more objective interpretation of factors that are otherwise difficult to measure or typically measured subjectively, such as:

- The amount of stress or frustration in a customer's voice
- How fast the individual is speaking (rate of speech)
- Changes in the level of stress indicated by the person's speech (such as in response to a solution provided by a customer support representative) In customer service and call center applications, sentiment analysis is a valuable tool for monitoring opinions and emotions among various customer segments, such as customers interacting with a certain group of representatives, during shifts, customers calling regarding a specific issue, product or service lines, and other distinct groups. Sentiment analysis may be fully automated, based entirely on human analysis, or some combination of the two. In some cases, sentiment analysis is primarily automated with a level of human oversight that fuels machine learning and helps to refine algorithms and processes, particularly in the early stages of implementation.

Examples of Sentiment Analysis

Sentiment analysis is used across a variety of applications and for myriad purposes. For instance, sentiment analysis may be performed on Twitter to determine overall opinion on a particular trending topic. Companies and brands often utilize sentiment analysis to monitor brand reputation across social media platforms or across the web as a whole. One of the most widely used applications for sentiment analysis is for monitoring call center and customer support performance. As companies seek to keep a finger on the pulse of their audiences, sentiment analysis is increasingly utilized for overall brand monitoring purposes. Sentiment analysis has been used by political candidates and administrations to monitor overall opinions about policy changes and campaign announcements, enabling them to fine-tune their approach and messaging

to better relate to voters and constituents. In brand reputation management applications, overall trends in sentiment analysis enables brands to identify peaks and valleys in overall brand sentiment or shifts in attitudes about products or services, thus enabling companies to make improvements perfectly in-tune with customer demands.

Benefits of Sentiment Analysis

A relative sentiment analysis score provides insight into the effectiveness of call center agents and customer support representatives and also serves as a useful measurement to gauge the overall opinion on a company's products or services. When sentiment analysis scores are compared across certain segments, companies can easily identify common pain points, areas for improvement in the delivery of customer support, and overall satisfaction between product lines or services. By monitoring attitudes and opinions about products, services, or even customer support effectiveness continuously, brands are able to detect subtle shifts in opinions and adapt readily to meet the changing needs of their audience.

Challenges and Best Practices for Sentiment Analysis

Language is complex, and as a process for quantifying and scoring language, sentiment analysis is equally complex. What is relatively easy for humans to gauge subjectively in face-to-face communication, such as whether an individual is happy or sad, excited or angry, about the topic at hand, must be translated into objective, quantifiable scores that account for the many nuances that exist in human language, particularly in the context of a discussion. For instance, a word that otherwise carries a positive connotation used in a sarcastic manner could easily be misinterpreted by an algorithm if both context and tone are not taken into consideration. Given these challenges, sentiment analysis solutions must consider acoustic measurements (the rate of speech, stress in a caller's voice, and changes in stress signals) in the context of the conversation. Additionally, integrating machine learning into the mix enables sentiment analysis to become more accurate over time, as algorithms learn and adapt to the commonalities in conversations and how the context of conversations can change outcomes. In call center and customer support applications, sentiment analysis that operates in real-time provides crucial feedback to agents and representatives, allowing them to respond appropriately to impact the outcome of the interaction. When used in conjunction with powerful features like Semantic Building Blocks, companies can easily monitor agent performance by identifying positive patterns such as frustration or dissatisfaction in the early stages of a call followed by positive sentiments in the later portion of the call, representing an agent's ability to talk-down frustrated customers. When utilizing the right technology tools and applying it to key business drivers, sentiment analysis is a powerful tool for steering companies and their individual business units to successful outcomes from every customer interaction.

2. Literature survey

The paper “Sentiment analysis of twitter published in 2012 introduces a machine learning approach to implement sentiment analysis on the data. They have performed sentiment classification of Twitter data where the classes are positive ,negative and neutral. Two sorts of models have been utilized: Tree part and highlight based models and both these models beat the unigram pattern. For the element based approach they performed include examination, Which uncovers that the most critical components are those that join the earlier extremity of words and their parts of discourse labels. In “The Twitter Sentiment Analysis: The Bad The Good and The OMG” paper, they have explored the utility of phonetic components for recognizing the assumption of twitter messages. They have known the value of existing of lexical assets and in addition includes that catch data about the casual and imaginative dialect utilized as a part of various social sites. An approach has been introduced to solve the problems. In This Paper “The Twitter Sentiment Classification using Discrete Supervision” published in 2009 introduces a novel approach for naturally grouping the feeling of various twitter message. These messages are either classified as The positive or negative with respect to the data . The paper describes the preprocessing of various steps in order to achieve very high accuracy. The principle commitment of this paper is to utilizing tweets with emoticons for far off regulated learning Diverse machine learning classifiers and highlight extractors have been utilized alongside the utilization of unigrams, bigrams, unigrams and bigrams, and parts of discourse as components.

3. Proposed Work

We have proposed a system that performs aspect level sentiment analysis on twitter data or tweets based on movies into two categories:

• Positive

• Negative The Following is a brief Description associated with the various tweets.

• Emoticons: The expressions which are used to describe the users conditions or feelings for an issue or his personal issues.

• Target: The Twitter users will use the special characters "@" symbol to simply refer to other users on the various micro blog which continuously alerts them

• Hashtags: The Users usually use The hash tags (#) to refer to various topics. This is to increase the views of their personal tweets. Aspect Level Sentiment Classification: Sentence level or document level sentiment classification is insufficient in many applications as it only reflects the overall opinion and does not evaluate all the aspects of an entity. Hence in order to understand the sentiment of each aspect, we perform aspect-level sentiment analysis or feature based opinion mining. This paper, proposes to perform sentiment analysis of multiple aspects of various entities related to movies, products, companies.

For example:

• "I love #hrithik so much, cant wait to see his film" When we want to find the tweets above the hero Hrithik. Let us consider the above tweet as the retrieved data . Now we apply the sentiment function to the above tweet.The word "Love" in the above tweet is a positive word . So the score of the tweet would be +1.

• "I abhor @hrithik movies. When we want to find the tweets above the hero Hrithik. Let us consider the above tweet as the retrieved data . Now we apply the sentiment function to the above tweet.The word "abhor" in the above tweet means negative word. So the score of tweet would be -1.

• " I love #hrithik so much, but I abhor his movies." Let us consider this tweet as the retrieved data , Now let us apply the sentiment function on the above tweet.The word "Love" and "abhor" are positive and negative words in the above tweet. So the score of the tweet would Zero. The following steps proposed in our paper are :

• Data collection using Twitter API: Large sets of twitter data is not available publicly. Hence we first extract the twitter data from the Twitter API.

• Data Preprocessing: This involves cleaning and simplifying the data by performing spell correction, punctuation handling etc. so as to remove noise from the data. • Applying Classification algorithms: The classification algorithms are applied on these tweets in order to categorize them. Different models provide different accuracy and we choose the model with

highest accuracy.

• Classified tweets: The results of the above step is classifies tweets which may belong to any of the three categories mentioned.

• Sentiments in graphical representation : The results of the sentiment analysis is provided using histograms

Henceforth prior polarities of individual words (whether the words generally carry positive or negative connotations) may alone not enough for the problem. The paper explores some other features which include grammar and syntactical relationships between words to make their classifier better at judging the contextual polarity of them phrase. The task of twitter sentiment analysis can be most closely related to phrase level sentiment analysis. A seminal paper on phrase level sentiment analysis was presented in 2005 by Wilson et al. which identified a new approach to the problem by first classifying phrases according to subjectivity (polar) and objectivity(neutral) and then further classifying the subjective classified phrases as either positive or negative. The paper noticed that many of the objective phrases used prior sentiment bearing words in them, which led to poor classification of especially objective phrases.It claims that if we use a simple classifier which assumes that the contextual polarity of the word is merely equal to its prior polarity gives a result of about 48%.

4. Functionality and Design

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

I. Data Acquisition II. Human Labelling III. Feature Extraction IV. Classification

Data Acquisition: Data in the form of raw tweets is acquired by using the python library "tweetstream" which provides a package for simple twitter streaming API [26]. This API allows two modes of accessing tweets: SampleStream and FilterStream. SampleStream simply delivers a small, random sample of all the tweets streaming at a real time. FilterStream delivers tweet which match a certain criteria. It can filter the delivered tweets according to three criteria:

•Specific keyword(s) to track/search for in the tweets

•Specific Twitter user(s) according to their user-id

•Tweets originating from specific location(s) (only for geo-tagged tweets). A programmer can specify any single one of these filtering criteria or a multiple combination of these. But for our purpose we have no such restriction and will thus stick to the SampleStream mode.

Since we wanted to increase the generality of our data, we acquired it in portions at different points of time instead of acquiring all of it at one go. If we used the latter approach then the generality of the tweets might have been compromised since a significant portion of the tweets would be referring to some certain trending topic and would thus have more or less of the same general mood or sentiment. This phenomenon has been observed when we were going through our sample of acquired tweets. For example the sample acquired near Christmas and New Year's had a significant portion of tweets referring to these joyous events and were thus of a generally positive sentiment. Sampling our data in portions at different points in time would thus try to minimize this problem. A tweet acquired by this method has a lot of raw information in it which we may or may not find useful for our particular application. It comes in the form of the python "dictionary" data type with various key-value pairs. A list of some key-value pairs are given below:

•Whether a tweet has been favorited

•User ID

•Screen name of the user

•Original Text of the tweet

•Presence of hashtags

•Whether it is a retweet

•Language under which the twitter user has registered their account

•Ge-tag location of the tweet

•Date and time when the tweet was created Since this is a lot of information we only filter out the information that we need and discard the rest. For our particular application we iterate through all the tweets in our sample and save the actual text content of the tweets in a separate file given that language of the twitter is user's account is specified to be English. The original text content of the tweet is given under the dictionary key "text" and the language of user's account is given under " lang". Since human labelling is an expensive process we further filter out the tweets to be labelled so that we have the greatest amount of variation in tweets without the loss of generality. The filtering criteria applied are stated below:

•Remove Retweets (any tweet which contains the string "RT")

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

•Remove non-English tweets (by comparing the words of the tweets with a list of 2,000 common English words, tweets with less than 15% of content matching threshold are discarded)

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

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

Human Labelling:

For the purpose of human labelling we made three copies of the tweets so that they can be labelled by four individual sources. This is done so that we can take average opinion of people on the sentiment of the tweet and in this way the noise and inaccuracies in labelling can be minimized. Generally speaking the more copies of labels we can get the better it is, but we have to keep the cost of labelling in our mind, hence we reached at the reasonable figure of three . We labelled the tweets in four classes according to sentiments expressed/observed in the tweets: positive, negative, neutral/objective and ambiguous.We gave the following guidelines to our labellers to help them in the labelling process:

•Positive: If the entire tweet has a positive/happy/excited/joyful attitude or if something is mentioned with positive connotations. Also if more than one sentiment is expressed in the tweet but the positive sentiment is more dominant. Example: "4 more years of being in shithole Australia then I move to the USA! :D".

•Negative: If the entire tweet has a negative/sad/displeased attitude or if something is mentioned

with negative connotations. Also if more than one sentiment is expressed in the tweet but the negative sentiment is more dominant. Example: “I want an android now this iPhone is boring :S”.

•Ambiguous : If more than one sentiment is expressed in the tweet which are potent with no one particular sentiment standing out and becoming more obvious. Also if it is obvious that some personal opinion is being expressed here but due to lack of reference to context it is difficult/ impossible to accurately decipher the sentiment expressed. Example: “I kind of like heroes and don’t like it at the same time...”. Finally if the context of the tweet is not apparent from the information available.

Example:“ That’s exactly how I feel about avengers haha”.

•<Blank>: Leave the tweet unlabelled if it belongs to some language other than English so that it is ignored in the training data. Besides this labellers were instructed to keep personal biases out of labelling and make no assumptions, i.e. judge the tweet not from any past extra personal information and only from the information provided in the current individual tweet. Once we had labels from four sources our next step was to combine opinions of three people to get an averaged opinion. The way we did this is through majority vote. So for example if a particular tweet had to two labels in agreement, we would label the overall tweet as such. But if all three labels were different, we labelled the tweet as “unable to reach a majority vote”. We arrived at the following statistics for each class after going through majority voting.

•Positive: 2543 tweets

• Negative: 1877 tweets

•Ambiguous: 451 tweets

•Unable to reach majority vote: 390 tweets

• Unlabelled non-English tweets: 369 tweets So if we include only those tweets for which we have been able to achieve a positive, negative or neutral majority vote, we are left with 8963 tweets for our training set. Out of these 4543 are objective tweets and 4420 are subjective tweets (sum of positive and negative tweets). We also calculated the human-human agreement for our tweet labelling task, results of which are as follows: Human 1: Human 2 Human 2: Human 3 Human 1: Human 3 Strict 58.9% 59.9%

62.5% Lenient 65.1% 67.1% 73.0%

In the above matrix the “strict” measure of agreement is where all the label assigned by both human beings should match exactly in all cases, while the “lenient” measure is in which if one person marked the tweet as “ambiguous” and the other marked it as something else, then this would not count as a disagreement. So in case of the “lenient” measure, the ambiguous class could map to any other class. So since the human-human agreement lies in the range of 60- 70% (depending upon our definition of agreement), this shows us that sentiment classification is inherently a difficult task even for human beings. We will now look at another table presented by Kim et al. which shows human-human agreement in case labelling individual adjectives and verbs. Adjectives Verbs Human 1: Human 2 Human 1: Human 3 Strict 76.19% 62.35% Lenient 88.96% 85.06%

Over here the strict measure is when classification is between the three categories of positive, negative and neutral, while the lenient measure the positive and negative classes into one class, so now humans are only classifying between neutral and subjective classes. These results reiterate our initial claim that sentiment analysis is an inherently difficult task. These results are higher than our agreement results because in this case humans are being asked to label individual words which is an easier task than labelling entire tweets.

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

•Tokenization: It is the process of breaking a stream of text up into words, symbols and other meaningful elements called “tokens”. Tokens can be separated by whitespace characters and/or punctuation characters. It is done so that we can look at tokens as individual components that make up a tweet.

•Url’s and user references (identified by tokens “http” and “@”) are removed if we are interested in only analyzing the text of the tweet.

•Punctuation marks and digits/numerals may be removed if for example we wish to compare the tweet to a list of English words.

•Lowercase Conversion: Tweet may be normalized by converting it to lowercase which makes it’s comparison with an English dictionary easier.

•Stemming: It is the text normalizing process of reducing a derived word to its root or stem. For example a stemmer would reduce the phrases “stemmer”, “stemmed”, “stemming” to the root word “stem”. Advantage of stemming is that it makes comparison between words simpler, as we do not need to deal with complex grammatical transformations of the word. In our case we employed the algorithm of “porter stemming” on both the tweets and the dictionary, whenever there was a need of comparison.

•Stop-words removal: Stop words are class of some extremely common words which hold no

additional information when used in a text and are thus claimed to be useless. Examples include “a”, “an”, “the”, “he”, “she”, “by”, “on”, etc. It is sometimes convenient to remove these words because they hold no additional information since they are used almost equally in all classes of text, for example when computing prior-sentiment-polarity of words in a tweet according to their frequency of occurrence in different classes and using this polarity to calculate the average sentiment of the tweet over the set of words used in that tweet.

•Parts-of-Speech Tagging: POS-Tagging is the process of assigning a tag to each word in the sentence as to which grammatical part of speech that word belongs to, i.e. noun, verb, adjective, adverb, coordinating conjunction etc. Now that we have discussed some of the text formatting techniques employed by us, we will move to the list of features that we have explored. As we will see below a feature is any variable which can help our classifier in differentiating between the different classes. There are two kinds of classification in our system (as will be discussed in detail in the next section), the objectivity/ subjectivity classification and the positivity/ negativity classification. As the name suggests the former is for differentiating between objective and subjective classes while the latter is for differentiating between positive and negative classes. The list of features explored for objective/ subjective classification is as below:

•Number of exclamation marks in a tweet

•Number of question marks in a tweet

•Presence of exclamation marks in a tweet

•Presence of question marks in a tweet

•Presence of url in a tweet

•Presence of emoticons in a tweet

•Unigram word models calculated using Naive Bayes

•Prior polarity of words through online lexicon MPQA

•Number of digits in a tweet

•Number of capitalized words in a tweet

•Number of capitalized characters in a tweet

•Number of punctuation marks / symbols in a tweet

•Ratio of non-dictionary words to the total number of words in the tweet

•Length of the tweet

•Number of adjectives in a tweet

•Number of comparative adjectives in a tweet

•Number of superlative adjectives in a tweet

•Number of base- form verbs in a tweet

•Number of past tense verbs in a tweet

•Number of present participle verbs in a tweet

•Number of past participle verbs in a tweet

•Number of 3rd person singular present verbs in a tweet

•Number of non-3rd person singular present verbs in a tweet

•Number of adverbs in a tweet

•Number of personal pronouns in a tweet

•Number of possessive pronouns in a tweet

•Number of singular proper noun in a tweet

•Number of plural proper noun in a tweet

•Number of cardinal numbers in a tweet

•Number of possessive endings in a tweet

•Number of wh-pronouns in a tweet

•Number of adjectives of all forms in a tweet

•Number of verbs of all forms in a tweet

•Number of nouns of all forms in a tweet

•Number of pronouns of all forms in a tweet

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

•Overall emoticon score (where 1 is added to the score in case of positive emoticon, and 1 is subtracted in case of negative emoticon)

•Overall score from online polarity lexicon MPQA (where presence of strong positive word in the tweet increases the score by 1.0 and the presence of weak negative word would decrease the score by 0.5)

•Unigram word models calculated using Naive Bayes

•Number of total emoticons in the tweet

•Number of positive emoticons in a tweet

•Number of negative emoticons in a tweet

•Number of positive words from MPQA lexicon in tweet

•Number of negative words from MPQA lexicon in tweet

•Number of base- form verbs in a tweet

•Number of past tense verbs in a tweet

•Number of present participle verbs in a tweet

•Number of past participle verbs in a tweet

•Number of 3rd person singular present verbs in a tweet

•Number of non-3rd person singular present verbs in a tweet

•Number of plural nouns in a tweet

•Number of singular proper nouns in a tweet

•Number of cardinal numbers in a tweet

•Number of prepositions or coordinating conjunctions in a tweet

•Number of adverbs in a tweet

•Number of wh-adverbs in a tweet

•Number of verbs of all forms in a tweet Next we will give mathematical reasoning of how we calculate the unigram word models using Naive Bayes. The basic concept is to calculate the probability of a word belonging to any of the possible classes from our training sample. Using mathematical formulae we will demonstrate an example of calculating probability of word belong to objective and subjective class. Similar steps would need to be taken for positive and negative classes as well. We will start by calculating the probability of a word in our training data for belonging to a particular class: We now state the Bayes' rule. According to this rule, if we need to find the probability of whether a tweet is objective, we need to calculate the probability of tweet given the objective class and the prior probability of objective class. The term P(tweet) can be substituted with P(tweet| obj) + P(tweet| subj). Now if we assume independence of the unigrams inside the tweet (i.e. the occurrence of a word in a tweet will not affect the probability of occurrence of any other word in the tweet) we can approximate the probability of tweet given the objective class to a mere product of the probability of all the words in the tweet belonging to objective class. Moreover, if we assume equal class sizes for both objective and subjective class we can ignore the prior probability of the objective class. Henceforth we are left with the following formula, in which there are two distinct terms and both of them are easily calculated through the formula mention above.Now that we have the probability of objectivity given a particular tweet, we can easily calculate the probability of subjectivity given that same tweet by simply subtracting the earlier term from 1. This is because probabilities must always add to 1. So if we have information of P(obj | tweet) we automatically know P(subj | tweet). Finally we calculate P(obj | tweet) for every tweet and use this term as a single feature in our objectivity / subjectivity classification. There are two main potential

problems with this approach. First being that if we include every unique word used in the data set then the list of words will be too large making the computation too expensive and time-consuming. To solve this we only include words which have been used at least 5 times in our data. This reduces the size of our dictionary for objective / subjective classification from 11,216 to 2,320. While for positive / negative classification unigram dictionary size is reduced from 6,502 to 1,235 words. The second potential problem is if in our training set a particular word only appears in a certain class only and does not appear at all in the ot her class (for example if the word is misspelled only once). If we have such a scenario then our classifier will always classify a tweet to that particular class (regardless of any other features present in the tweet) just because of the presence of that single word. This is a very harsh approach and results in over-fitting. To avoid this we make use of the technique known as "Laplace Smoothing". In this formula "x" is a constant factor called the smoothing factor, which we have arbitrarily selected to be 1. How this works is that even if the count of a word in a particular class is zero, the numerator still has a small value so the probability of a word belonging to some class will never be equal to zero. Instead if the probability would have been zero according to the earlier formula, it would be replace by a very small non- zero probability. The final issue we have in feature selection is choosing the best features from a large number of features. Our ultimate aim is to achieve the greatest accuracy of our classifier while using least number of features. This is because adding new feature add to the dimensionality of our classification problem and thus add to the complexity of our classifier. This increase in complexity may not necessarily be linear and may even be quadratic so it is preferred to keep the features at a minimum low. Another issue we have with too many features is that our training data may be over-fit and it may confuse the classifier when doing classification on an unknown test set, so the accuracy of the classifier may even decrease. To solve this issue we select the most pertinent features by computing the information- gain of all the features under exploration and then selecting the features with highest information gain.

Classification:

Pattern classification is the process through which data is divided into different classes according to some common patterns which are found in one class which differ to some degree with the patterns found in the other classes. The ultimate aim of our project is to design a classifier which accurately classifies tweets in the following four sentiment classes: positive, negative, neutral and ambiguous. There can be two kinds of sentiment classifications in this area: contextual sentiment analysis and general sentiment analysis. Contextual sentiment analysis deals with classifying specific parts of a tweet according to the context provided, for example for the tweet "4 more years of being in shithole Australia then I move to the USA :D" a contextual sentiment classifier would identify Australia with negative sentiment and USA with a positive sentiment.

On the other hand general sentiment analysis deals with the general sentiment of the entire text (tweet in this case) as a whole. Thus for the tweet mentioned earlier since there is an overall positive attitude, an accurate general sentiment classifier would identify it as positive. For our particular project we will only be dealing with the latter case, i.e. of general (overall) sentiment analysis of the tweet as a whole. The classification approach generally followed in this domain is a two-step approach. First Objectivity Classification is done which deals with classifying a tweet or a phrase as either objective or subjective. After this we perform Polarity Classification (only on tweets classified as subjective by the objectivity classification) to determine whether the tweet is positive, negative or both (some researchers include the both category and some don't). This was presented by Wilson et al. and reports enhanced accuracy than a simple one-step approach. We propose a novel approach which is slightly different from the approach proposed by Wilson et al. We propose that in first step each tweet should undergo two classifiers: the objectivity classifier and the polarity classifier. The former would try to classify a tweet between objective and subjective classes, while latter would do so between the positive and negative classes. We use the short -listed features for these classifications and use the Naive Bayes algorithm so that after the first step we have two numbers from 0 to 1 representing each tweet. One of these numbers is the probability of tweet belonging to objective class and the other number is probability of tweet belonging to positive class. Since we can easily calculate the two remaining probabilities of subjective and negative by simple subtraction by 1, we don't need those two probabilities. So in the second step we would treat each of these two numbers as separate features for another classification, in which the feature size would be just 2. We can use and apply the following Machine Learning algorithms for this second classification to arrive at the best result:

- K-Means Clustering
- Support Vector Machine
- Logistic Regression
- K Nearest Neighbours
- Naive Bayes
- Rule Based Classifiers

However, the classifier we used in this project is Naive Bayes.

5. Implementation

The steps in implementation are :

a) Getting text from feeds

b) Defining text cleaning functions

c) Cleaning and splitting twitter feeds

d) Analyzing twitter feeds

e) Plotting high frequency negative and positive words

a) Getting text from feeds : Twitter sustains have huge amounts of extra fields and implanted pointless data. We utilize the gettext() capacity to remove the content fields and appoint the rundown to a variable tweetT. The capacity is connected to every one of the 5000 tweets. The code beneath likewise indicates consequences of extraction for the initial 5 sustains.tweetT=lapply(tweet,function(t)t$getText()) head(tweetT,5)

b) Defining text cleaning functions : In this progression, we compose a capacity which executes a progression of orders to clean content, removes punctuation, special characters, inserted HTTP joins, additional spaces, and digits. This function likewise changes capitalized characters to lower case string utilizing tolower() work. Ordinarily, the tolower() work stops startlingly as it experiences unique characters ceasing execution of the r code. To dodge this, we compose a blunder getting capacity, tryTolower, and install it in the code of the content cleaning function.

c) Cleaning and Splitting twitter feeds : In this step, we separate the tweets. The resultant feeds are stored in a list object.

d) Analyzing twitter feeds : Here we get into the actual task of analyzing feeds. We compare the twitter text feeds with the word dictionaries and retrieve out the matching words. To do this, we first define a function to count the positive and negative words that are matching with our database.

e)Plotting high frequency negative and positive words: The resultant output for the word hrithik is

Machine Learning :

Once we have applied the different steps of the preprocessing part, we can now focus on the machine learning part. There are three major models used in sentiment analysis to classify a sentence into positive or negative: SVM, Naive Bayes and Language Models (NGram). SVM is known to be the model giving the best results but in this project we focus only on probabilistic model that are Naive Bayes and Language Models that have been widely used in this field. Let's first introduce the Naive Bayes model which is well known for its simplicity and efficiency for text classification. 2.4.1 Naive Bayes In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive)independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum likelihood training can be done by evaluating a closed form expression (mathematical expression that can be evaluated in a finite number of operations), which takes linear time. It is based on the application of the Bayes rule given by the following formula: Formula 2.4.1.1: Bayes rule where D denotes the document and C the category (label), d and c are instances of D and C and $P(D = d) = \sum (D|C)P(C)$. We can simplify this expression by, $c \in C\ P = d = c = c$ Formula 2.4.1.2: Bayes rule simplified In our case, a tweet d is represented by a vector of K attributes such as $d = (w, w, ..., w)$. 1 2 k Computing P(d|c) is not trivial and that's why the Naive Bayes introduces the assumption that all of the feature values wj are independent given the category label c . That is, for $i =/ j$, wi and wj are conditionally independent given the category label c . So the Bayes rule can be rewritten as, Formula 2.4.1.3: Bayes rule rewritten 14 Based on this equation, maximum a posterior (MAP) classifier can be constructing by seeking the optimal category which maximizes the posterior P(c|d) : Formula 2.4.1.4: Classifier maximizing the posterior probability P(c|d) Note that P(d) is removed since it is a constant for every category c . 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: Formula 2.4.1.5: Prior distribution P(c) where N is the total number of training tweets and N is the number of training tweets in class i c . 15 The likelihood P(w |c) is usually computed using the formula: j Formula 2.4.1.6: Likelihood P(w |c) j 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 zero probability problem, that arises when our model encounters a word seen in the test set but not in the training set, Laplaceor add one 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.

The Naive Bayes Classifier:

Naive Bayes classifier is a simple classifier that has its foundation on the well known Bayes's theorem. Despite its simplicity, it remained a popular choice for text classification.

. In simplest form for event A and B , Bayes theorem relates two conditional probabilities as follows:

$$P(A \cap B)=P(A,B)=P(A)P(B|A)=P(B)P(A|B) \Longrightarrow P(B|A)=P(B)P(A|B) \,/\, P(A)$$

Frequency of tweets : Racist and Non-Racist

Word Cloud of the most frequent words:

Word Cloud of the non-Racist words:

Word Cloud of the Racist words:

Code

```
library(caret)
library(descr)
library(NLP)
library(tm)
library(SnowballC)

#reading the dataset
tweets <- read.csv("/Users/mallamehervishal/Downloads/tweets.csv")
#analysing the tweets dataset
#invokes the sprea-sheet for the object
View(tweets)
#retrieves the dimension of the object
dim(tweets)
#prints frequency of the object
freq(tweets)
#displays internal structure of th object
str(tweets)

#encoding tweets$labels into vector
tweets$label <- factor(tweets$label)
#cross classifies the object
table(tweets$label)

#converting tweets$twweet into documents for data cleaning
tweetscorpus <- Corpus(VectorSource(tweets$tweet))
#displays detailed information about CORPUS
inspect(tweetscorpus)

#converting all corpus data into LOWER CASE
tweetscorpus <- tm_map(tweetscorpus, tolower)

#removes stopwords from corpus
tweetscorpus <- tm_map(tweetscorpus, removeWords, stopwords("english"))

#removes punctuation
tweetscorpus <- tm_map(tweetscorpus, removePunctuation)

#removes stem words
tweetscorpus <- tm_map(tweetscorpus, stemDocument)

#removes numbers
tweetscorpus <- tm_map(tweetscorpus, removeNumbers)

#removes extra white spaces from the corpus
tweetscorpus <- tm_map(tweetscorpus, stripWhitespace)

#constructs term-document matrix
tweetscorpus_dtm <- DocumentTermMatrix(tweetscorpus) #,control = list(tolower = TRUE, removeNumbers = TRUE, stopwords =
 TRUE, removePunctuation = TRUE,stemming = TRUE))

#Invoke a spreadsheet-style data viewer for tweetscorpus_dtm.
View(tweetscorpus_dtm)
```

```
#splitting the converted corpus dataset
#train dataset
traint <- tweetscorpus_dtm[1:20000,]
dim(traint)
View(traint)
#test dataset
testt <- tweetscorpus_dtm[20001:31962,]
dim(testt)
View(testt)

#trainlabels dataset splitting
traintlabels <- tweets[1:20000,]$label
#invokes spread sheet for traintlabels object
View(traintlabels)

#testlabels dataset splitting
testtlabels <- tweets[20001:31962,]$label
#invokes spread sheet for testtlabels object
View(testtlabels)

#removes non-zero elements from traint
tweetscorpus_dtm_freq_train<-removeSparseTerms(traint,0.999)
#removes non-zero elements from testt
tweetscorpus_dtm_freq_test<-removeSparseTerms(testt,0.999)

#finds frequent terms which is repeated 3 or more times from the traint document
tweets_freq_words <- findFreqTerms(tweetscorpus_dtm_freq_train,3)
tweets_freq_words

#finds frequent terms which is repeated 3 or more times from the testt document
tweets_freq_words2 <- findFreqTerms(tweetscorpus_dtm_freq_test,3)
tweets_freq_words2

tweetscorpus_dtm_freq_train <- tweetscorpus_dtm_freq_train[ ,tweets_freq_words]
tweetscorpus_dtm_freq_test <- tweetscorpus_dtm_freq_test[ ,tweets_freq_words2]

#user defined function which counts the tems in the document
convert_counts<-function(x){
  x<-ifelse(x>0,"Yes","No")
}

#applies user-defined function to the margin to tweetscorpus_dtm_freq_train
tweets_train<-apply(tweetscorpus_dtm_freq_train, MARGIN=2, convert_counts)
View(tweets_train)
#applies user-defined function to the margin to tweetscorpus_dtm_freq_test
tweets_test<-apply(tweetscorpus_dtm_freq_test, MARGIN=2, convert_counts)
View(tweets_test)

#this library is used to import package "NAIVE BAYES"
library(e1071)
#creating model for naive bayes and laplace is used for smoothing
tweetnb <- naiveBayes(tweets_train,traintlabels, laplace = 1)
#predicting the label values from created model
tweetpred <- predict(tweetnb, tweets_test)

#confusion matrix between predicted and test labels
```

```
CrossTable(tweetpred, testtlabels, prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE, dnn = c('predicated','actual'))

#gives the accuracy of the predicted values
confusionMatrix(tweetpred, testtlabels)$overall
```

Prediction

Accuracy

```
Total Observations in Table:  11962


             | actual
   predicted |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |     10775 |       444 |     11219 |
             |     0.970 |     0.518 |           |
-------------|-----------|-----------|-----------|
           1 |       330 |       413 |       743 |
             |     0.030 |     0.482 |           |
-------------|-----------|-----------|-----------|
Column Total |     11105 |       857 |     11962 |
             |     0.928 |     0.072 |           |
-------------|-----------|-----------|-----------|


> # Creating  a truth table by tabulating the predicted class labels with the
actual class labels
> table("Prediction"=tw_test_pred2,"Actual" =tw_test_labels)
          Actual
Prediction     0     1
         0 10775   444
         1   330   413
> #using confusion matrix to find the accuracy of our model
> confusionMatrix(tw_test_pred2, tw_test_labels)$overall['Accuracy']*100
Accuracy
93.52951
>
```

6. Conclusion

The project helps us to analyze huge amount of data and process it. The data will be collected by the twitter streaming API. The data collected will be analyzed, based on score we analyze how users are feeling about the product or company etc. We can also use this to visualize the users' opinion towards other products in the market by drawing a bar graph. This project not only analyzes the sentiments of the users but can be very helpful in marketing sector.

The task of sentiment analysis, especially in the domain of micro-blogging, is still in the developing stage and far from complete. So we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance. Right now we have worked with only the very simplest unigram models; we can improve those models by adding extra information like closeness of the word with a negation word. We could specify a window prior to the word (a window could for example be of 2 or 3 words) under consideration and the effect of negation may be incorporated into the model if it lies within that window. The closer the negation word is to the unigram word whose prior polarity is to be calculated, the more it should affect the polarity. For example if the negation is right next to the word, it may simply reverse the polarity of that word and farther the negation is from the word the more minimized ifs effect should be. Apart from this, we are currently only focusing on unigrams and the effect of bigrams and trigrams may be explored. As reported in the literature review section when bigrams are used along with unigrams this usually enhances performance. However for bigrams and trigrams to be an effective feature we need a much more labeled data set than our meager 9,000 tweets. Right now we are exploring Parts of Speech separate from the unigram models, we can try to incorporate POS information within our unigram models in future. So say instead of calculating a single probability for each word like P(word | obj) we could instead have multiple probabilities for each according to the Part of Speech the word belongs to. For example we may have P(word | obj, verb), P(word | obj, noun) and P(word | obj, adjective). Pang et al. [5] used a somewhat similar approach and claims that appending POS information for every unigram results in no significant change in performance (with Naive Bayes performing slightly better and SVM having a slight decrease in performance), while there is a significant decrease in accuracy if only adjective unigrams are used as features. However these results are for classification of reviews and may be verified for sentiment analysis on micro blogging websites like Twitter. One more feature we that is worth exploring is whether the information about relative position of word in a tweet has any effect on the performance of the classifier. Although Pang et al. explored a similar feature and reported negative results, their results were based on reviews which are very different from tweets and they worked on an extremely simple model.

One potential problem with our research is that the sizes of the three classes are not equal. The objective class which contains 4,543 tweets is about twice the sizes of positive and negative classes which contain 2,543 and 1,877 tweets respectively. The problem with unequal classes is that the classifier tries to increase the overall accuracy of the system by increasing the accuracy of the majority class, even if that comes at the cost of decrease in accuracy of the minority classes. That is the very reason why we report significantly higher accuracies for objective class as opposed to positive or negative classes. To overcome this problem and have the classifier exhibit no bias towards any of the classes, it is necessary to label more data (tweets) so that all three of our classes are almost equal.

In this research we are focussing on general sentiment analysis. There is potential of work in the field of sentiment analysis with partially known context. For example we noticed that users generally use our website for specific types of keywords which can divided into a couple of di stinct classes, namely: politics/politicians,celebrities, products/brands, sports/sportsmen, media/ movies/music. So we can attempt to perform separate sentiment analysis on tweets that only belong to one of these classes (i.e. the training data would not be general but specific to one of these categories) and compare the results we get if we apply general sentiment analysis on it instead.

Last but not the least, we can attempt to model human confidence in our system. For example if we have 5 human labellers labelling each tweet, we can plot the tweet in the 2 -dimensional objectivity / subjectivity and positivity / negativity plane while differentiating between tweets in which all 5 labels agree, only 4 agree, only 3 agree or no majority vote is reached. We could develop our custom cost function for coming up with optimized class boundaries such that highest weightage is given to those tweets in which all 5 labels agree and as the number of agreements start decreasing, so do the weights assigned. In this way the effects of human confidence can be visualized in sentiment analysis.

7. References

1. R Documentation
2. Stack Overflow
3. https://www.tidytextmining.com/sentiment.htm
4. Sentiment analysis in twitter using machine learning techniques
https://ieeexplore.ieee.org/document/6726818/
5. RPubs
6. RBloggers