2015

Twitter Sentiment Analysis – NBA

PROJECT – ISEN 613
TEAM - 13

Bolishetty, Manjusha
Gourishetti, Santhosh
Neelagiri, Nitish
Virupakshi, PhaniBhushan

Contents

Executive Summary

Statement of the Importance

Twitter Sentiment Analysis has grown widely over the past few years and this field of science offers a great advantage to companies or individuals to know more about positivity and negativity in their tweets which might affect their business. We would like to understand the application of classification algorithms such as Naïve Bayes, Maximum Entropy and Support Vector Machines on the tweet data collected on multiple sport teams and predict the accuracy of the models on the test data.

Statement of objectives

We chose to analyze tweets by fans on each team of NBA which can aid sponsorship decision for companies. Our objective is to create a model specifically for NBA (basketball). This sentiment analysis gives richer understanding of fan base sentiment on team or specific players.

Gap in Literature

Sentiment analysis is relatively new and broad approach in understanding sentiment of a particular group of people. There can be huge number of models pertaining to each topic. In this project we focused on a group of people who watch NBA and express their emotions via microblogging (Twitter).

Methods Used

Major approach in this project is to take each word from tweets and create a "Bag-of-Words" which acts as features for predicting the overall sentiment of each tweet. For predicting we tried to implement algorithms like SVM, Naive Bayes Classifier, Random Forest, and Boosting.

Challenges

Sentiment ambiguity is a real problem for most tools. Text with sentiment ambiguity is often classified as neutral. For example, *"The rooms at this hotel are priced higher than others in the area"* doesn't contain any charged words, although it's clearly a negative comment. Most Sentiment Analysis tools have no way to capture context. A positive or negative sentiment word can have the opposite connotation depending on context (e.g. *“This is a great hotel, if you like the smell of urine”).* We would like to address these problems by using a wide manually classified dataset to train the algorithm. This will create an accurate training data set which captures the context of the tweets.

Introduction

Importance of the problem

Sentiment Analysis refers to “the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral” {1}. With social media serving up millions of data points by way of comments, tweets, and status updates, it’s becoming increasingly important to be able to take all that subjective text and turn it into quantitative data that can be integrated with other metrics.

According to a survey by Rival IQ, an analytics company,

- 50% of shoppers have made purchases based on recommendations found through a social network
- 79% of Twitter users are more likely to recommend brands they follow
- 67% of Twitter users are more likely to buy from brands they follow

Even sports teams and players can use social media to grow their individual brands. Twitter sentiment analysis will provide sports teams opportunities to develop some richer descriptions of their fan bases. The idea behind sentiment is that we look at the “tone” of tweets surrounding each team. In this study, we are examining the distribution of positive versus negative tweets for sports teams.

Objectives & Scope of work

In this project, we would like to perform twitter sentiment analysis on sport teams and analyze positive and negative sentiment tweets using the data. We would like to understand the application of classification algorithms such as Naïve Bayes, Maximum Entropy and Support Vector Machines on the tweet data collected on multiple sport teams and predict the accuracy of the models on the test data.

The dataset is extracted from R using searchTwitter() method of the twitteR package. We plan to extract data for the previous year (2014) for training and the current year (2015) for testing the accuracy of the classification algorithms.

The analysis that we are conducting has code available but there are few cleaning and transformation tasks that we are doing it by ourselves. We will be using unigram feature selection while categorizing tweets as positive or negative. Unigram will use only one keyword and bigram will use two keywords for categorization.

Scope of this project is to train algorithms to classify each tweets positive or negative and measure accuracy of each algorithm to pick a best one.

Literature review

Paper 1

Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." *CS224N Project Report, Stanford* 1 (2009): 12.

Review by Phani Bhushan Virupakshi

Go et al. (2009) introduced a novel approach for automatically classifying the sentiment of Twitter messages. They presented the results of machine learning algorithms for classifying the sentiment of Twitter messages using distant supervision. They showed that machine learning algorithms (Naive Bayes, Maximum Entropy, and SVM) have accuracy above 80% when trained with emoticon data.

Approach

Go et al.'s (2009) approach was to use different machine learning classifiers and feature extractors. The machine learning classifiers are Naive Bayes, Maximum Entropy (MaxEnt), and Support Vector Machines (SVM). The feature extractors are unigrams, bigrams, unigrams and bigrams, and unigrams with part of speech tags. They built a framework that treats classifiers and feature extractors as two distinct components.

Go et al.'s (2009) approach took advantage of different twitter properties to reduce the feature space. Usernames, Usage of links and repeated letters are reduced. They were able to shrink the feature set down to 45.85% of its original size.

Evaluation

Authors considered the twitter data from April 6, 2009 to June 25, 2009 as their training data and it is post-processed with the filters. Emoticons are stripped off, tweet containing both positive and negative emoticons are removed, Retweets are removed, Tweets with ":P" are removed and Repeated tweets are removed. After post-processing the data, study took the first 800,000 tweets with positive emoticons, and 800,000 tweets with negative emoticons, for a total of 1,600,000 training tweets.

Results

Table: 3. Results

Features	Keyword	Naïve Bayes	Max Ent	SVM
Unigram	65.2	81.3	80.5	82.2
Bigram	N/A	81.6	79.1	78.8
Unigram+Bigram	N/A	82.7	83.0	81.6
Unigram+POS	N/A	79.9	79.9	81.9

Takeaways

The paper discussed on approaches of using unigram, bigram and parts of speech. However bigram approach involves highly sparse matrix and results are inferior to unigram approach when computations are taken into consideration . Also parts of speech (POS) tags are also not so useful as the results are almost similar to unigram approach.

Paper 2

Kouloumpis, Efthymios, Theresa Wilson, and Johanna Moore. "Twitter sentiment analysis: The good the bad and the omg!." *Icwsm* 11 (2011): 538-541.

Review by Phani Bhushan Virupakshi

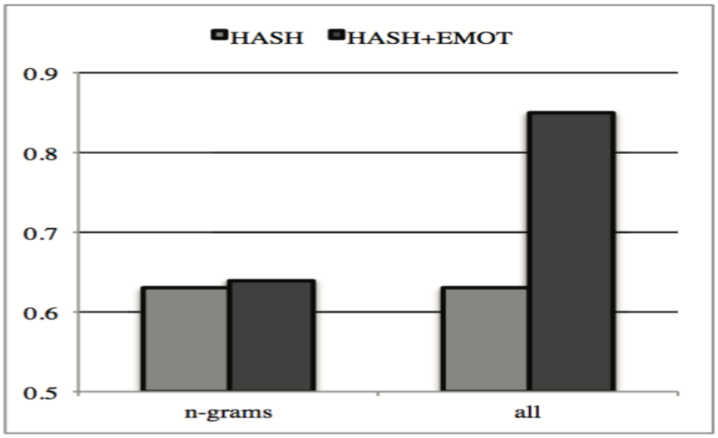
In Kouloumpis et al.'s (2011) summary article investigating the linguistic features for detecting the sentiment of twitter messages, authors took a supervised approach to the problem, but leverage existing hashtags in the Twitter data for building training data. Study found that using hashtags to collect training data did prove useful, as did using data collected based on positive and negative emoticons (Kouloumpis et al, 2011).

Approach

Kouloumpis et al (2011) used three different corpora of Twitter messages in our experiments. For development and training, we use the the hashtagged data set (HASH), which we compile from the Edinburgh Twitter corpus1, and the emoticon data set (EMOT) . The hashtagged data set is a subset of the Edinburgh Twitter corpus. The Emoticon data set was created by Go et al. (2009). The iSieve data contains approximately 4,000 tweets. It was collected and hand-annotated by the iSieve Corporation.

They processed this data in three steps, 1) tokenization, 2) normalization, and 3) part-of-speech (POS) tagging. Variety of features were used for classification experiments ie.. n-gram features, Lexicon features, Part-of-speech features, Micro-blogging features.

Results



The best performance on the evaluation data comes from using the n-grams together with the lexicon features and the microblogging features.

Takeaways

Inclusion of parts of speech may not be useful for sentiment analysis in microblogging domain. Presence of emoticons and abbreviations were useful.

Paper 3

Pak, Alexander, and Patrick Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining." *LREC*. Vol. 10. 2010.

Review by Santhosh Gourishetti

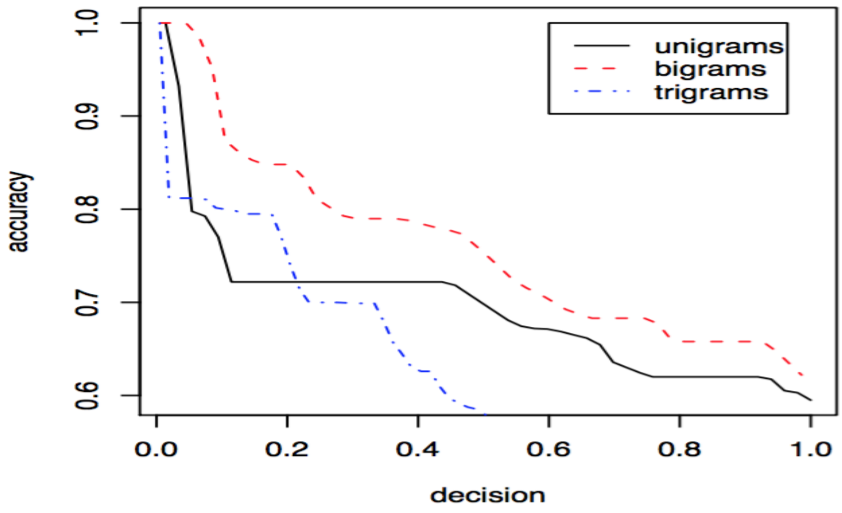
Pak et al. (2010) studied how microblogging can be used for sentiment analysis purposes. They used twitter as a corpus for sentiment analysis and opinion mining and performed a linguistic analysis of twitter corpus (having 300000 text posts). They also demonstrated how to build a sentiment classifier that uses the collected corpus as training data.

Approach

Pak et al. (2010) approached the problem of opinion mining as follows

1. They presented a method to automatically collect an arbitrarily large corpus of twitter data containing positive and negative comments as well as objective text.
2. Performed statistical linguistic analysis on the corpus data.
3. Used the corpora to build the classifier.
4. Conducted experiments on the test data to prove the efficiency of the technique.

Results



Bigrams gave the best performance which is a good balance between coverage(unigrams) and sentiment expression(trigrams).

Takeaways

Tweets can be assumed as a single sentence and each word in the sentence corresponds to same sentiment the user want to express. Very rarely tweets have multiple emotions. By this assumptions it is safe to classify the tweets as either positive or negative.

Paper 4

Boulis, Constantinos, and Mari Ostendorf. "Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams."*Proc. of the International Workshop in Feature Selection in Data Mining*. 2005.

Review by: Santhosh Gourishetti

Approach

The paper throws light on various techniques available for text classification like bag-of-words vector, more complex methods of using bigrams and parts of speech tags. The new features give best results when used all alone and when used them in combination with other methods. Searching for optimal feature subsets might be expensive given the order of vocabulary in tens of thousands. Finally the most widely used techniques of Support vector machines and Naïve Bayes theorem are used. There are three application of feature augmentation to 3 text corpora mentioned as below.

- 20News- groups corpus,
- A collection of postings to electronic forums
- WebKB corpus, a collection of web pages

In section 2 the paper describes about adding relevant and non – redundant bigrams and approaches to problem of feature selection by using filter approach and wapper approach. Section 3 describes the experiments carried out with details on the corpora used, learning methods and the evaluation of these methods. The feature selection is experimented using Naïve and SVM algorithms. The data was split in the ratio of 80/20 for training/test datasets. K-fold cross validation technique with K=10 was used to cross validate the results. Validation is carried out on the three corpora’s. The paper concludes with a note that the bag-of-words approach is assumed to handle all the complex words and other complex methods for text classification are hard to model and less accurate and hence text classification research is more focused on the learning method and not the representation of the vectors. Using the augmented space approach along with the SVM methods has a better results than using bag-of-words and individual space.

Takeaway

Other papers discussed on lesser impact of considering bigrams over unigram as features but this paper tried to improvise upon applying bigram approach and considering redundancy of these bigrams. Each bigram is added according to how much more information it brings compared to unigram.

Paper 5

Neethu, M. S., and R. Rajasree. "Sentiment analysis in twitter using machine learning techniques." Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on. IEEE, 2013.

Review by Nitish Neelagiri

Approach

The paper talks about Twitter Sentiment Analysis using knowledge based techniques like bag of words and lexical database and Machine Learning techniques.

The twitter sentiment analysis is difficult because the amount of content generated is large and it has a lot of slang text which needs to be first cleaned. The paper states the knowledge based techniques are difficult to implement. Machine Learning Algorithms like Naive Bayes (NB), Maximum Entropy (ME), and Support Vector Machines (SVM) are used to classify reviews. Some features that can be used for Sentiment classification are Term Presence, Term Frequency, negation, n-grams and Part-of-Speech. The solution proposed in the paper uses a Symbolic approach with a sentence level sentiment analysis carried out in 3 stages

1. Preprocessing stage and creation of the dataset: cleaning the data to remove all the URL, extra spaces, numbers and defining a set of words as positive, negative, a dictionary of slang words.

2. Creation of feature vector – After the cleansing of data they are classified as different vectors falling in positive, negative and emoticons which are classified as negative and positive. They are assigned weights of 1 and -1 respectively.

3. Sentiment classification using various Machine Learning Techniques – Naïve Bayes classification, SVM classifier, Maximum Entropy classifier, Ensemble classifier.

The paper concludes with the statement that Machine Learning techniques are simpler and efficient than Symbolic techniques and can be applied for twitter sentiment analysis. It states the issues while dealing with identifying emotional keyword from tweets having multiple keywords. Difficulties in handling misspellings and slang words which is handled by an efficient feature vector by doing feature extraction in two steps after proper preprocessing. In the first step, twitter specific features are extracted and added to the feature vector. The classification accuracy of the feature vector is tested using various classification algorithms and all the classifiers have almost similar accuracy for the new feature vector. This feature vector performs well for electronic products domain.

Takeaway

The paper discussed a technique to classify for training the algorithm which works like each word in tweet are looked up in a dictionary of positive and negative words, and given a score of 1 or -1.

Machine learning algorithms Naive Bayes (NB), Maximum Entropy (ME), and Support Vector Machines (SVM) gave out good accuracy for sentiment prediction.

Paper 6

Niu, Zhen, Zelong Yin, and Xiangyu Kong. "Sentiment classification for microblog by machine learning." *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on*. IEEE, 2012.

Review by Nitish Neelagiri

Approach

The paper speaks about three text classification methods in microblogging and enhancement of one of the techniques. The paper introduces various approaches to sentiment analysis namely – Machine Learning algorithms and Semantic Lexicon-based approach. Semantic Lexicon is an unsupervised learning which can classify text as absolute positive or negative with a greater accuracy while ML algorithms are more efficient in evaluating ambiguous or complicated sentiments. The paper highlights about various some background information where in people already have done some research on text analysis. Traditional ML algorithm can be applied for text classification with minor improvements in this method. Following are the approaches taken to improve the traditional ML algorithm for text classification.

- Calculating weights of feature words
- Using words' relative position in text
- Tagging part of speech

The complete process in divided into 5 stages.

- Extraction of key words as feature item- Selecting of feature words and calculating the weights for them - Remove words with lesser frequency.
- Calculating the weights of each feature word
- Training samples
- Sentiment classification
- Evaluation of performance

Following are the popular methods in sentiment classification mentioned in the paper

- Naïve Bayesian Text Classifier(NB)
- Maximum entropy (ME)
- Support Vector Machine (SVM)

Takeaway

This paper described the three algorithms mathematically and importance of each algorithm in particular scenarios. This also introduced terms like “precision”, “recall” & “balanced F-score” for measuring the accuracy of predictions.

Paper 7

Radovanović, Miloš, and Mirjana Ivanović. "Text mining: Approaches and applications." *Novi Sad J. Math* 38.3 (2008): 227-234.

Review by: Manjusha Bolishetty

Approach

In this paper, the authors review various ML approaches available for sentiment classification using a Bag of Words (BOW) document model. The BOW model for representing a piece of text is a simple and easy to generate model, but produces extremely high dimensional feature vectors. Such high dimensional features are problematic when trying to apply supervised or unsupervised Machine Learning techniques. In section 7, the authors study the effectiveness of different dimensionality reduction techniques in a sentiment classification application using a movie review dataset where reviews are classified into two categories: positive and negative, with a bag of words, 2-grams and 3-grams model. The dimensionality reduction techniques applied are Information Gain (IG), SVD, SIMPLS, SVD+LDA, LDA/QR, SRDA (spectral regression). Performance of these approaches are then compared using the movie review dataset using a kNN classifier. The naïve Bayes and the SVM classifiers are also applied using original high dimensional features. It is observed that the best performing approaches for dimensionality reduction of features are the SIMPLS and the SVD+LDA.

Takeaway

We got to know about the clustering and dimensionality reduction techniques which further improves accuracy of model along with time to predict.

Paper 8

Vinodhini, G., and R. M. Chandrasekaran. "Sentiment analysis and opinion mining: a survey." *International Journal* 2.6 (2012).

Review by: Manjusha Bolishetty

Approach

This paper is a survey on Sentiment Analysis and Opinion mining. The throws information on various sources for text mining in section 2, classification techniques in section 3, application and tools in section 4 and evaluation and discussions in section 5. There are various sources of data for sentiment analysis also referred to as opinion mining in the paper. Data from blogs – a popular means to express one personal opinion, review sites – that help user's make effective buying decisions, dataset – readily available data from certain companies for developers to work on and micro blogging like Twitter where users express opinion on different topics are the sources mentioned in the paper.

As per the paper, Sentiment analysis can be done at 3 levels – Document, sentence and attribute level. In addition to Semantic orientation process and Machine learning (ML) algorithms, natural language processing techniques(NPL) can also be used. The ML algorithms use two datasets – a training dataset used by an automatic classifier and the test dataset which validates the accuracy of the automatic classifier on the training classifier. The most well know ML algorithms spoken about in the paper are Naïve Bayes, Maximum Entropy, Support Vector Machine. The other NPL algorithms well known in text mining areas are the K-Nearest neighbourhood, ID3, C5, centroid classifier, winnow classifier, and the N-gram model. The paper introduces tools like

Review Seer, Web Fountain and various online tools like Twitrratr, Twendz ,Social mention, and Sentimetrics are available to track the sentiment in social networks.

The evaluation of various methods is done by metric like precision, recall and F-measure.

With the various performance measure used by various methods, the paper states that it is still difficult to to judge the best choice of classification method. Finally the paper concludes stating that there are a wide variety of a applications in information systems – classifying and summarizing review and other real time applications. It is found that the sentiment classifiers are very domain dependent. Neither of the classification methods mentioned outperform the other but are well suited and efficient in some scenarios only.

Some major challenges that are evident in Sentiment Analysis like dealing with negation expressions; produce a summary of opinions based on product features/attributes, complexity of sentence/ document , handling of implicit product features are mentioned in this reading.

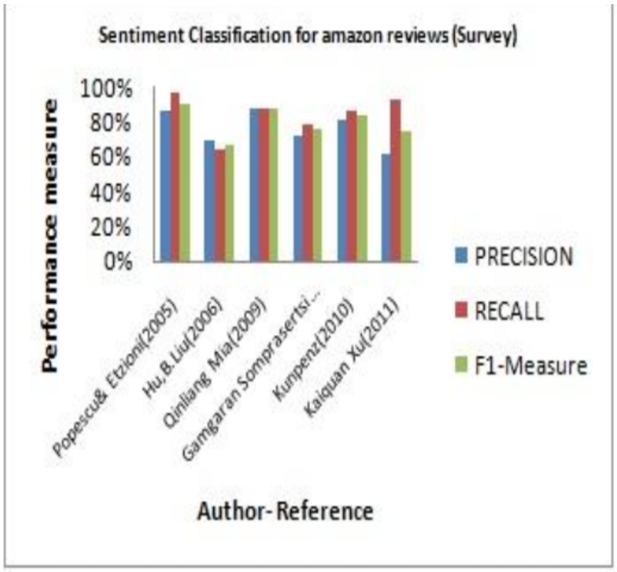


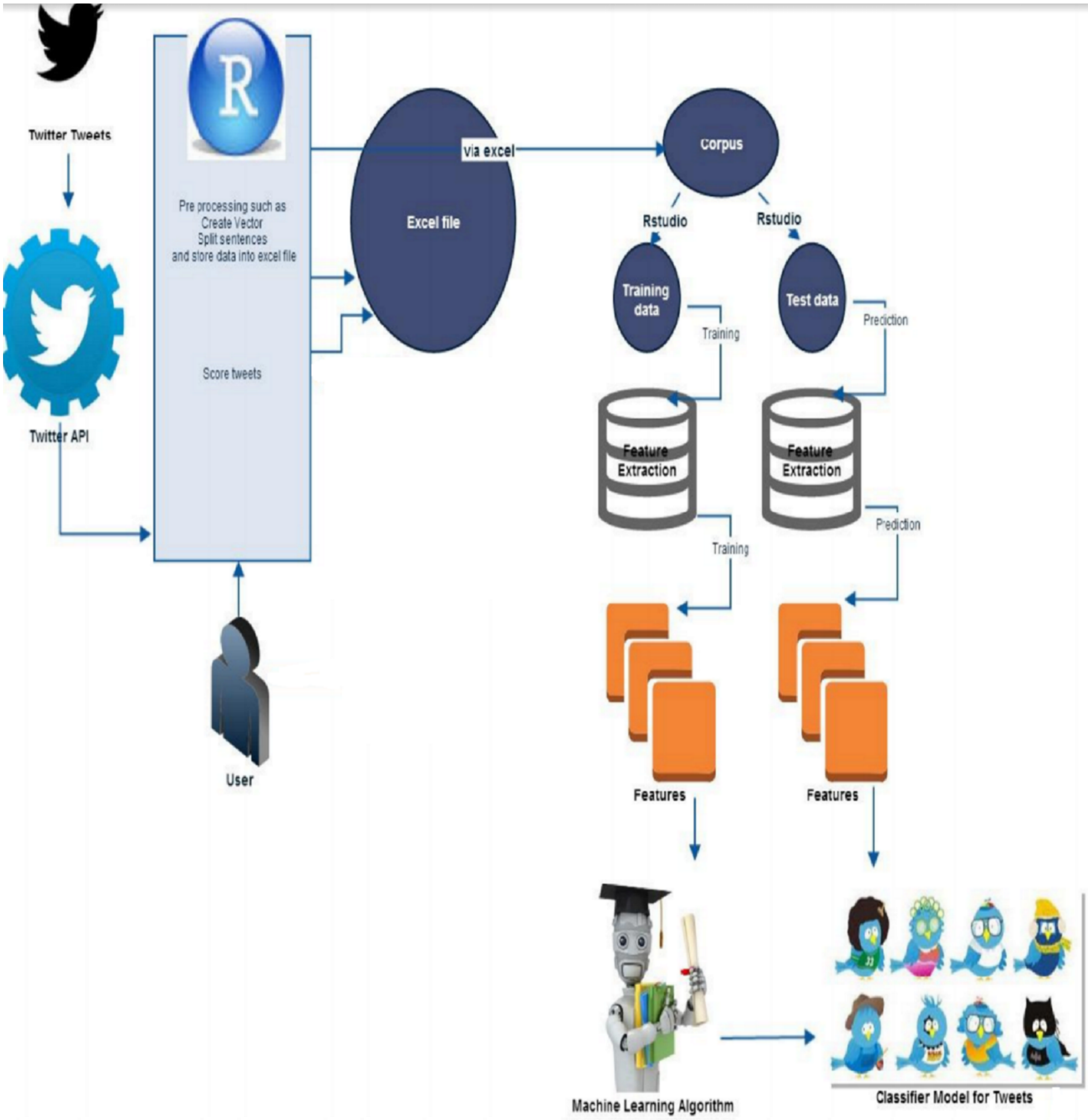
Fig 1. Sentiment Classification for Amazon reviews (Survey)

Takeaway

Neither classification outperforms the other and different types of features have distinct distribution. Also different types of features and algorithms can be combined in order to benefit from merits of each and overcome individual drawbacks.

Project Approach

Flow chart of implementation



Description of the new technique

- The process starts with authenticating with Twitter API and providing keys to establish a handshake connection with Twitter API for a while.
- Initially tweets from different teams are gathered and cumulated into a single variable.
- Cleaning techniques such as eliminating retweets, excluding replies, removing punctuations, web urls, illegal characters, tab spaces, numerics have been followed. The resulting data contained set of words for each tweet that actually have been used to show any sentiment.
- Each tweet has been manually provided a sentiment. This has been done to eliminate inaccurate interpretation of words and use human intelligence in sentiment classification for better model building.
- The entire data has then been stored in a csv file. This data then has been used to generate a corpus which in turn is used to generate a term document matrix. This matrix contains all the features as columns and the count of those features in each tweet as observations.
- We have divided the matrix into two sets, one for training and the other for testing the results.
- Now, training matrix and the response of the training set is sent to each machine learning algorithm that first extracts all the features from the matrix and selects the best features available.
- The selection of features is different for different algorithms and the most efficient algorithms such as Random Forest and SVM does the process more efficiently. We have also performed steps such as removing correlation among features, using algorithms specific parameters such as type, gamma, cost, kernel etc. to train the efficient algorithm.
- The team has reduced the sparsity of the matrix using inbuilt methods. It decreases the number of features. The team also identified few inaccurate words and removed them from the model building as they would affect the accuracy of the model. Pruning methods have been followed on tree models, but they did not improve from the already existing models, which are very efficient.

- Then we used predict function to test the model on test data and generate prediction values. These values are then compared with the actual classified values to determine mean accuracy of each model.
- Then several cross validations have been performed on the data to identify misclassification error rate on both training and test data.
- Word Cloud has been generated, after, that shows the importance of each word in a plot. The larger the word, more significant it is.
- Q Plot has also been generated to show the number of positive, negative and neutral tweets in the entire data.

Implementation details

This section needs to be divided into multiple sections, each for different methods you have attempted

“twitteR” library was used to extract tweets of few NBA teams which include (as their twitter account names) - @warriors, @dallasmavs, @Lakers,@nyknicks, @LAClippers, @cavs, @HoustonRockets, @Suns, @celtics, @Bucks. All the text (tweets) were extracted using twitter api by using function “searchTwitter”. We did not consider retweets as including same tweet hundreds of times does not convey the right emotion.

#Code for extracting tweets

```
APIKey="XXXXX"
APISecret="XXXXX"
AccessToken="XXXXX"
AccessTokenSecret="XXXXX"
Auth<-setup_twitter_oauth(APIKey,APISecret,AccessToken,AccessTokenSecret)
1
tweets <- searchTwitter("@Bucks", 10000, lang="en")
twitterdf <- do.call("rbind", lapply(tweets, as.data.frame))
twitterdf <- twitterdf[twitterdf$isRetweet == "FALSE",]
```

After extracting tweets, they were exported to csv file where we manually assigned sentiment to each tweet as positive (P), negative (N), neutral (O).

Next, we cleaned the tweets by eliminating punctuations, numbers (sometimes this include score), http links, extra spaces, tabs, etc, to make the tweet plane text for feeding to algorithms.

In total we were able to extract 11,200 tweets and classify each tweet.

Before feeding this data to machine learning algorithms, tweets were converted to “Bag-of-Words” by using the corpus class and then to term document matrix where each tweet is a row and columns are the words from all the tweets. Each cell in this term document matrix have either 1 or 0 indicating whether each tweet (row) has that particular word (column).

For example if the tweet is “spurs great end to the quarter”, term document matrix looks like

	great	job	go	end	to	the	final	quarter	back
tweet	1	0	0	1	1	1	0	1	0

Next the tweets are further cleaned to eliminate few words which does not add to emotion like “the”, “to” etc. by including them in list of stopwords.

The next step of cleaning include a concept called “Stemming”, in which words used in different verb forms are modified to a common string. For example the words “update”, “updated”, “”updating” all convey the same meaning in broad perspective. By stemming all these words in tweets are modified to a common string “update”

#code for corpus & term document matrix

```
twitterCorpus <- Corpus(VectorSource(df_Raptors$text))
twitterCorpus <- tm_map(twitterCorpus,removeWords,stopwords("en"))
twitterCorpus <- tm_map(twitterCorpus, content_transformer(tolower))
Corpus_copy <- twitterCorpus
twitterCorpus <- tm_map(twitterCorpus, stemDocument)
stemCompletion_mod <- function(x,dict=Corpus_copy) {
  PlainTextDocument(stripWhitespace(paste(stemCompletion(unlist(strsplit(as.character(x),"
")),dictionary=dict),sep="", collapse=" ")))
}
twitterCorpus <- as.VCorpus(lapply(twitterCorpus, stemCompletion_mod))
Tdm=TermDocumentMatrix(twitterCorpus, control=list(wordLengths=c(1,Inf)))
```

As a last step to clean the data, we eliminated few words (columns) from term document matrix which are used infrequently. This part is cleaning is done by “removeSparseTerms” function to remove sparsity to desired extent.

After all cleaning and formatting, we were left with a total of 985 columns for 11200 rows of tweets.

Implementation of SVM

Most of the articles we read considered SVM as a reliable algorithm to predict text sentiment and we invested majority of time in bringing out best model by varying gamma & cost along with kernels which include linear, polynomial & radial. The main deciding factor to select best model is accuracy of prediction.

#code for SVM

```
model=svm(train,train_resp,type = "C-classification", kernel = "radial",gamma = 0.0001, cost = 10)
pred=predict(model,test)
table(predict=pred,truth=test_resp)
mean(pred==test_resp)
confusionMatrix(pred, test_resp)
```

Cross Validation was also being performed to avoid the overfitting of data as below

```
meth=c("polynomial","linear", "radial")
gam=c(0.0001,0.01,1,10)
cos=c(1,10,100)
for (m in meth){
  for (g in gam){
    for (c in cos){
      b=paste("for method=",m," gamma=",g,"cost=",c)
      print(b)
      SVM_CV <- cross_validate(container, 3, "SVM",method = "C-classification", kernel = m,cross = g, cost = c)
    }
  }
}
```

Implementation of Naive Bayes

Naive Bayes has been used as a basic algorithm for text classification. The accuracy of Naive Bayes has been good in many cases of text classification. But, for the tweet classification for our data, we were not able to realize a great deal of accuracy from naiveBayes algorithm.

Code:

```
#Generate training and testing data
train <- sample(nrow(documentMatrix), 70*nrow(documentMatrix)/100)
trainData <- documentMatrix[train,]
testData <- documentMatrix[-train,]
response <- as.matrix(allTweets$response)
trainResponse <- response[train,]
testResponse <- response[-train,]

model=naiveBayes(trainData,trainResponse,laplace = 0, threshold = 0.9, eps = 1)
pred=predict(object = model,newdata = testData, type="raw")
response <- rep(NA, nrow(pred))
for(i in 1:length(response))
   response[i] <- names(which(pred[i,] == max(pred[i,])))
table(response, testResponse)
mean(response[!is.na(response)]==testResponse[!is.na(response)])
```

Results

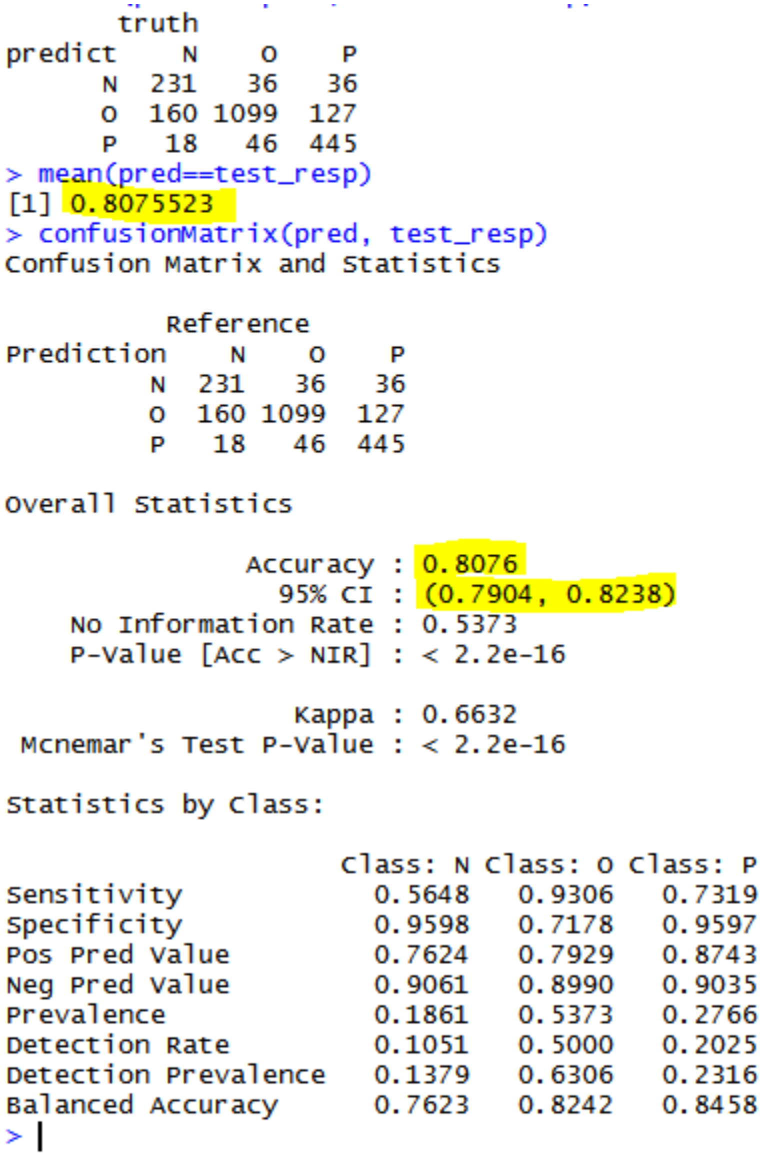
Results from best SVM model

Parameters for this model are

Kernel: Radial

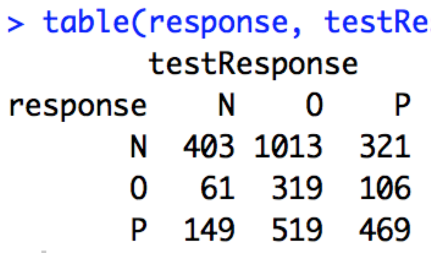
Gamma: 0.001

Cost: 10



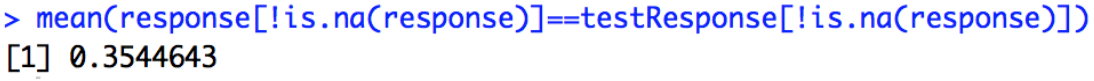
```
        truth
predict    N    O    P
      N  231   36   36
      O  160 1099  127
      P   18   46  445
> mean(pred==test_resp)
[1] 0.8075523
> confusionMatrix(pred, test_resp)
Confusion Matrix and Statistics

          Reference
Prediction    N    O    P
         N  231   36   36
         O  160 1099  127
         P   18   46  445

Overall Statistics

               Accuracy : 0.8076
                 95% CI : (0.7904, 0.8238)
    No Information Rate : 0.5373
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6632
 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: N Class: O Class: P
Sensitivity            0.5648   0.9306   0.7319
Specificity            0.9598   0.7178   0.9597
Pos Pred Value         0.7624   0.7929   0.8743
Neg Pred Value         0.9061   0.8990   0.9035
Prevalence             0.1861   0.5373   0.2766
Detection Rate         0.1051   0.5000   0.2025
Detection Prevalence   0.1379   0.6306   0.2316
Balanced Accuracy      0.7623   0.8242   0.8458
> 
```

Results from best fit of Naive Bayes



```
> table(response, testResponse)
          testResponse
response    N    O    P
       N  403 1013  321
       O   61  319  106
       P  149  519  469
```

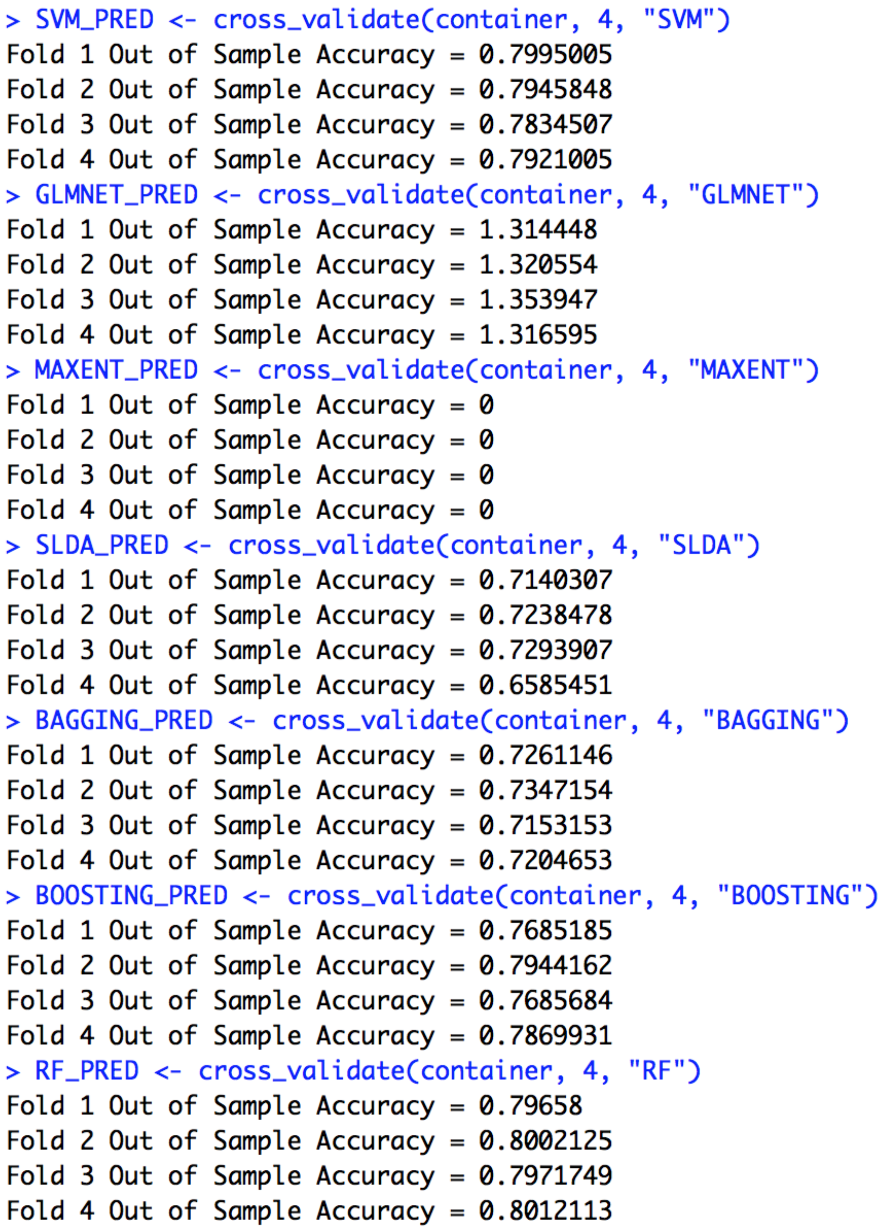
Above mentioned table is the confusion matrix that categorizes the sentiment based on test data and prediction data.



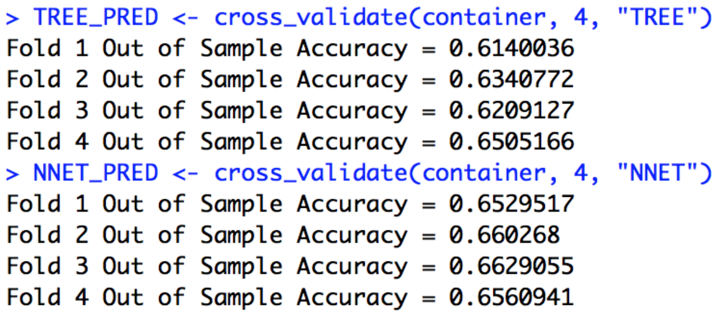
```
> mean(response[!is.na(response)]==testResponse[!is.na(response)])
[1] 0.3544643
```

Using RTextTools

Method 1: Document Matrix generated by the corpus

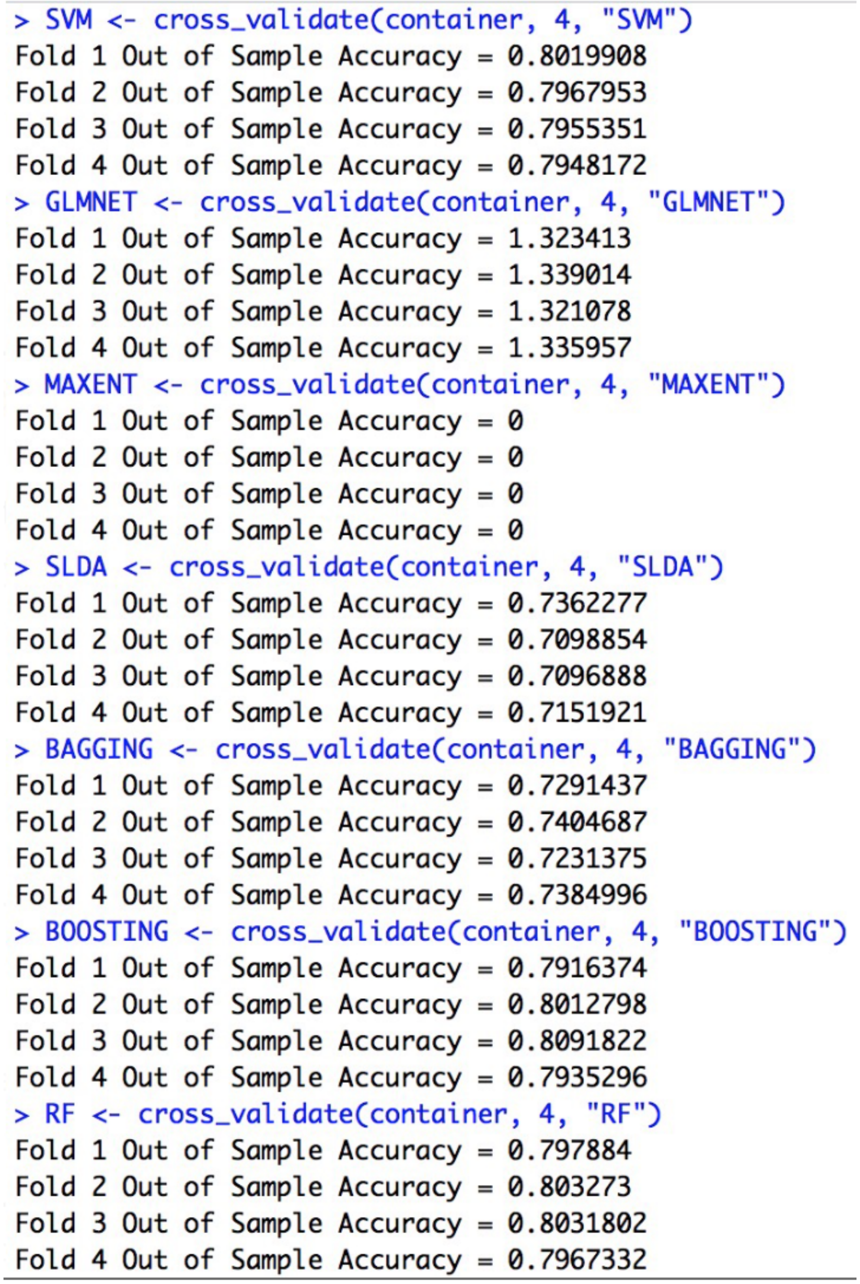


```
> SVM_PRED <- cross_validate(container, 4, "SVM")
Fold 1 Out of Sample Accuracy = 0.7995005
Fold 2 Out of Sample Accuracy = 0.7945848
Fold 3 Out of Sample Accuracy = 0.7834507
Fold 4 Out of Sample Accuracy = 0.7921005
> GLMNET_PRED <- cross_validate(container, 4, "GLMNET")
Fold 1 Out of Sample Accuracy = 1.314448
Fold 2 Out of Sample Accuracy = 1.320554
Fold 3 Out of Sample Accuracy = 1.353947
Fold 4 Out of Sample Accuracy = 1.316595
> MAXENT_PRED <- cross_validate(container, 4, "MAXENT")
Fold 1 Out of Sample Accuracy = 0
Fold 2 Out of Sample Accuracy = 0
Fold 3 Out of Sample Accuracy = 0
Fold 4 Out of Sample Accuracy = 0
> SLDA_PRED <- cross_validate(container, 4, "SLDA")
Fold 1 Out of Sample Accuracy = 0.7140307
Fold 2 Out of Sample Accuracy = 0.7238478
Fold 3 Out of Sample Accuracy = 0.7293907
Fold 4 Out of Sample Accuracy = 0.6585451
> BAGGING_PRED <- cross_validate(container, 4, "BAGGING")
Fold 1 Out of Sample Accuracy = 0.7261146
Fold 2 Out of Sample Accuracy = 0.7347154
Fold 3 Out of Sample Accuracy = 0.7153153
Fold 4 Out of Sample Accuracy = 0.7204653
> BOOSTING_PRED <- cross_validate(container, 4, "BOOSTING")
Fold 1 Out of Sample Accuracy = 0.7685185
Fold 2 Out of Sample Accuracy = 0.7944162
Fold 3 Out of Sample Accuracy = 0.7685684
Fold 4 Out of Sample Accuracy = 0.7869931
> RF_PRED <- cross_validate(container, 4, "RF")
Fold 1 Out of Sample Accuracy = 0.79658
Fold 2 Out of Sample Accuracy = 0.8002125
Fold 3 Out of Sample Accuracy = 0.7971749
Fold 4 Out of Sample Accuracy = 0.8012113
```

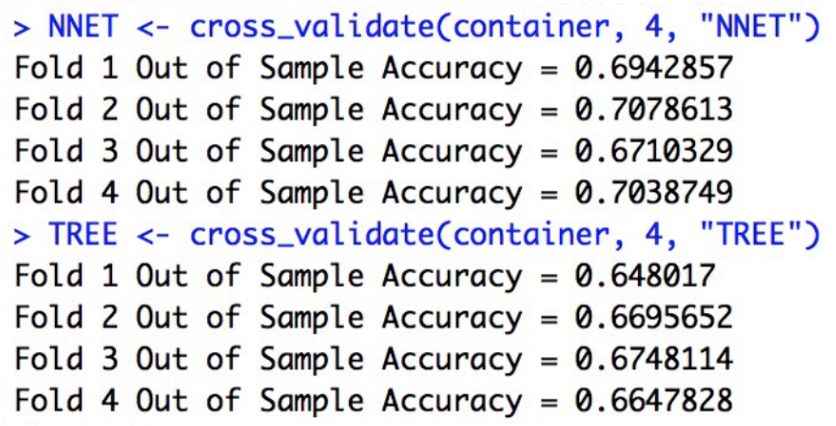


```
> TREE_PRED <- cross_validate(container, 4, "TREE")
Fold 1 Out of Sample Accuracy = 0.6140036
Fold 2 Out of Sample Accuracy = 0.6340772
Fold 3 Out of Sample Accuracy = 0.6209127
Fold 4 Out of Sample Accuracy = 0.6505166
> NNET_PRED <- cross_validate(container, 4, "NNET")
Fold 1 Out of Sample Accuracy = 0.6529517
Fold 2 Out of Sample Accuracy = 0.660268
Fold 3 Out of Sample Accuracy = 0.6629055
Fold 4 Out of Sample Accuracy = 0.6560941
```

Method 2: Document Matrix generated by create_matrix

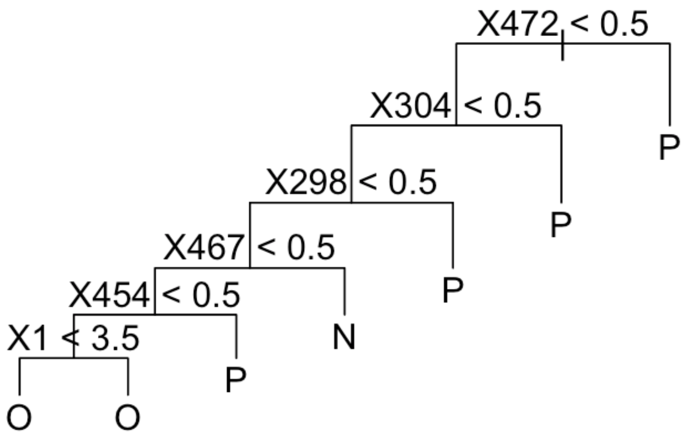


```
> SVM <- cross_validate(container, 4, "SVM")
Fold 1 Out of Sample Accuracy = 0.8019908
Fold 2 Out of Sample Accuracy = 0.7967953
Fold 3 Out of Sample Accuracy = 0.7955351
Fold 4 Out of Sample Accuracy = 0.7948172
> GLMNET <- cross_validate(container, 4, "GLMNET")
Fold 1 Out of Sample Accuracy = 1.323413
Fold 2 Out of Sample Accuracy = 1.339014
Fold 3 Out of Sample Accuracy = 1.321078
Fold 4 Out of Sample Accuracy = 1.335957
> MAXENT <- cross_validate(container, 4, "MAXENT")
Fold 1 Out of Sample Accuracy = 0
Fold 2 Out of Sample Accuracy = 0
Fold 3 Out of Sample Accuracy = 0
Fold 4 Out of Sample Accuracy = 0
> SLDA <- cross_validate(container, 4, "SLDA")
Fold 1 Out of Sample Accuracy = 0.7362277
Fold 2 Out of Sample Accuracy = 0.7098854
Fold 3 Out of Sample Accuracy = 0.7096888
Fold 4 Out of Sample Accuracy = 0.7151921
> BAGGING <- cross_validate(container, 4, "BAGGING")
Fold 1 Out of Sample Accuracy = 0.7291437
Fold 2 Out of Sample Accuracy = 0.7404687
Fold 3 Out of Sample Accuracy = 0.7231375
Fold 4 Out of Sample Accuracy = 0.7384996
> BOOSTING <- cross_validate(container, 4, "BOOSTING")
Fold 1 Out of Sample Accuracy = 0.7916374
Fold 2 Out of Sample Accuracy = 0.8012798
Fold 3 Out of Sample Accuracy = 0.8091822
Fold 4 Out of Sample Accuracy = 0.7935296
> RF <- cross_validate(container, 4, "RF")
Fold 1 Out of Sample Accuracy = 0.797884
Fold 2 Out of Sample Accuracy = 0.803273
Fold 3 Out of Sample Accuracy = 0.8031802
Fold 4 Out of Sample Accuracy = 0.7967332
```

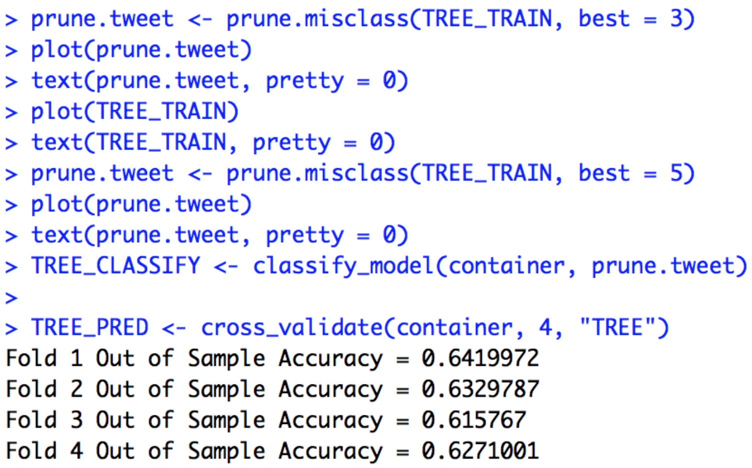


```
> NNET <- cross_validate(container, 4, "NNET")
Fold 1 Out of Sample Accuracy = 0.6942857
Fold 2 Out of Sample Accuracy = 0.7078613
Fold 3 Out of Sample Accuracy = 0.6710329
Fold 4 Out of Sample Accuracy = 0.7038749
> TREE <- cross_validate(container, 4, "TREE")
Fold 1 Out of Sample Accuracy = 0.648017
Fold 2 Out of Sample Accuracy = 0.6695652
Fold 3 Out of Sample Accuracy = 0.6748114
Fold 4 Out of Sample Accuracy = 0.6647828
```

Tree Outcome

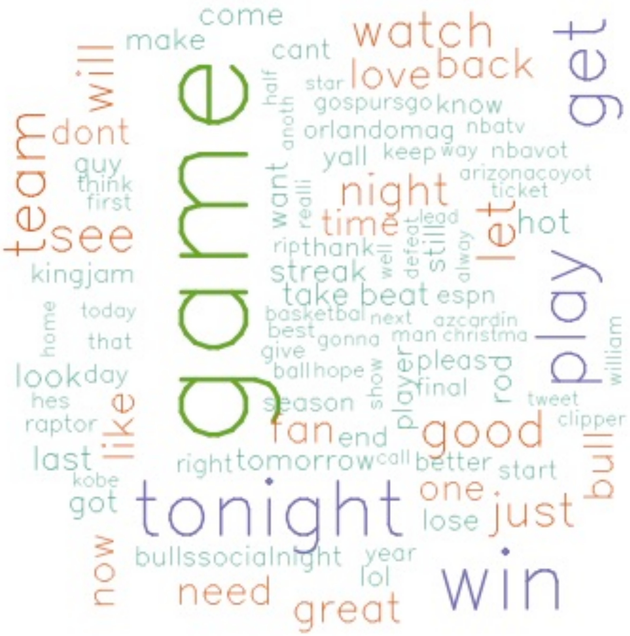


This shows the tree generated by the RTextTools package. This is the efficient tree with minimum number of terminal nodes.



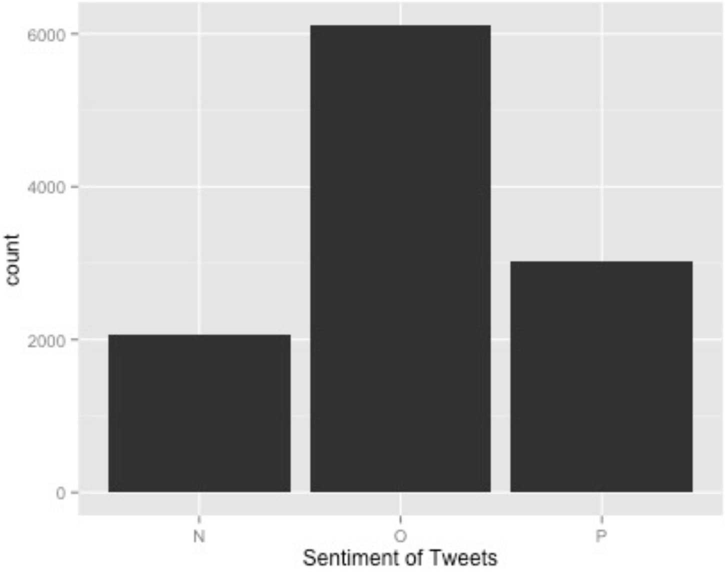
```
> prune.tweet <- prune.misclass(TREE_TRAIN, best = 3)
> plot(prune.tweet)
> text(prune.tweet, pretty = 0)
> plot(TREE_TRAIN)
> text(TREE_TRAIN, pretty = 0)
> prune.tweet <- prune.misclass(TREE_TRAIN, best = 5)
> plot(prune.tweet)
> text(prune.tweet, pretty = 0)
> TREE_CLASSIFY <- classify_model(container, prune.tweet)
>
> TREE_PRED <- cross_validate(container, 4, "TREE")
Fold 1 Out of Sample Accuracy = 0.6419972
Fold 2 Out of Sample Accuracy = 0.6329787
Fold 3 Out of Sample Accuracy = 0.615767
Fold 4 Out of Sample Accuracy = 0.6271001
```

Word Cloud



Word Cloud shows the significance of each word.

Tweet Significance



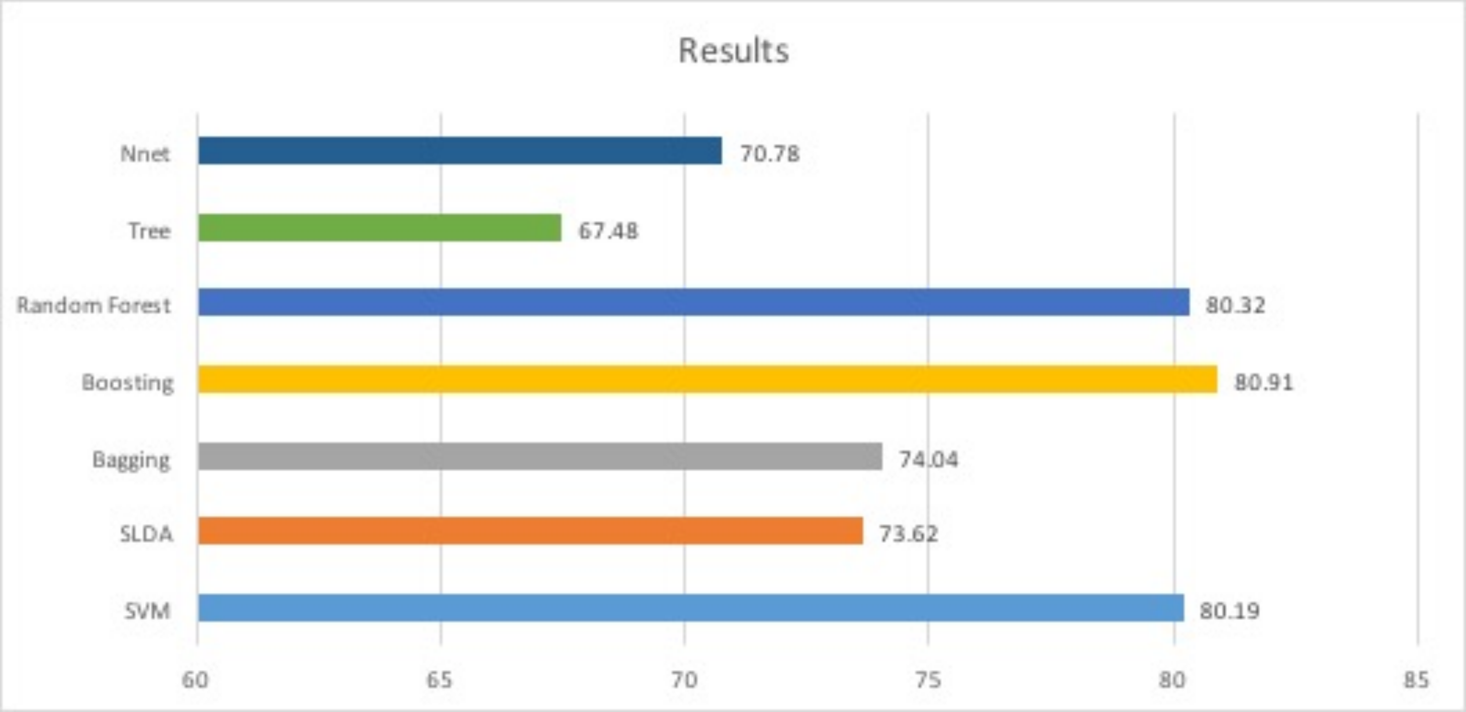
Most tweets are neutral, but positive sentiment in the tweets is visible.

Interpret the outcomes

- RTextTools package has a sophisticated set of algorithms that perform inbuilt optimization to text data and delivers the model which can be used to predict with good accuracy.
- Packages such as e1071 and klaR, though are good, have not been able to replicate the same level of efficiency of RTextTools package.
- There is no improvement in accuracy in decision tree even after it is pruned. This is because the tree has already been reduced to its best size. Thus pre-pruning has occurred with the use of RTextTools package.
- Cost and gamma parameters in svm has a huge bearing on the outcome. While using svm from e1071 package cost = 10 and gamma = 0.001 has resulted in the best accuracy. But using RTextTools, gamma = 0 and cost = 100 has resulted in the best accuracy.
- Removing correlated features has improved the accuracy of Naive Bayes from 27% to 35%. But it has not improved other models. Thus, we believe that the implementation of Naive Bayes is inferior to that of other highly accurate supervised algorithms and their implementation in RTextTools taken into account the correlation among the different features.

Compare outcomes from different methods

- SVM, Random Forest and boosting has the prediction accuracy of about 80%. This is good considering the average accuracy in text classification is 60 - 70% which we observed in many papers.
- Neural network, trees, SLDA and bagging offer relatively closer accuracy to 70%.


Results
Nnet
70.78
Tree
67.48
Random Forest
80.32
Boosting
80.91
Bagging
74.04
SLDA
73.62
SVM
80.19
60
65
70
75
80
85

Conclusions

Major takeaways

- Team has identified SVM, Random Forest and Boosting as the best models for text classification.
- There still needs a lot of testing of the models as the features change quickly in the real world and it takes a thorough research on best feature selection processes to come up with a model that can classify majority of tweets with a proven consistency.
- Few models such as Naive Bayes, GLM have not shown good results on basketball teams tweet data. This does not necessarily mean those algorithms do not suit all the tweets.
- Spell check is very difficult to implement manually and the auto suggest feature in spell check does not provide best word in many occasions. Thus it could add new features every time which are inaccurate and increase the complexity of model and decrease the accuracy.
- Team has become familiar with major text classification techniques and has gone out of their course curriculum to explore newer methods of data cleaning such as stemming, spell check, and stop words and tuning.

Individual Roles & Responsibilities

Name	Contribution	Roles & Responsibilities
Santhosh Gourishetti	25%	Collect & clean data, apply algorithms Review Papers to collect information on application of algorithms
Nitish Neelagiri	25%	Collect & clean data, apply algorithms Review Papers to collect information on application of algorithms
Phani Bhushan Virupakshi	25%	Collect & clean data, apply algorithms Review Papers to collect information on application of algorithms
Manjusha Bolishetty	25%	Collect & clean data, apply algorithms Review Papers to collect information on application of algorithms

References

1. Christine Day, The Importance of Sentiment Analysis in Social Media Analysis - https://www.linkedin.com/pulse/importance-sentiment-analysis-social-media-christine-day

2. Jurka, Timothy P., et al. "Rtexttools: A supervised learning package for text classification." (2011).

3. https://scholarblogs.emory.edu/esma/category/twitter/

4. Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." *CS224N Project Report, Stanford* 1 (2009)

5. Kouloumpis, Efthymios, Theresa Wilson, and Johanna Moore. "Twitter sentiment analysis: The good the bad and the omg!." *Icwsm* 11 (2011): 538-541.

6. Pak, Alexander, and Patrick Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining." *LREC*. Vol. 10. 2010

7. Boulis, Constantinos, and Mari Ostendorf. "Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams."*Proc. of the International Workshop in Feature Selection in Data Mining*. 2005.

8. Neethu, M. S., and R. Rajasree. "Sentiment analysis in twitter using machine learning techniques." Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on. IEEE, 2013.

9. Niu, Zhen, Zelong Yin, and Xiangyu Kong. "Sentiment classification for microblog by machine learning." *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on*. IEEE, 2012.

10. Radovanović, Miloš, and Mirjana Ivanović. "Text mining: Approaches and applications." *Novi Sad J. Math* 38.3 (2008): 227-234.

11. Vinodhini, G., and R. M. Chandrasekaran. "Sentiment analysis and opinion mining: a survey." *International Journal* 2.6 (2012).

Appendix

Make sure the TA or I can copy the data and code into a single folder and everything runs correctly, and the results we obtain are consistent with those you report

```
setwd("~/")
library(twitteR)
library(wordcloud)
library(RColorBrewer)
library(plyr)
library(ggplot2)
library(httr)
library(stringr)
library(e1071)
library(tm)
library(RTextTools)
library(utils)
library(qdap)
library(ggplot2)
library(SnowballC)
library(caret)

##Please do not include - Commented
##Establishing a handshake with Twitter Application.
oauth_endpoints("twitter")
api_key <- "XXXXXXXXXXXXX"
api_secret <- "XXXXXXXXXXXXXX"
access_token <- "XXXXXXXXXXXXXX"
access_token_secret <- "XXXXXXXXXXXXXXX"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

extractLimit <- 20000
tweets_raw <- searchTwitter('Lakers', n = extractLimit, since = '2015-01-01')

tweets_transformed <- do.call("rbind", lapply(tweets_raw, as.data.frame))
```

```
allTweets <- tweets_transformed[tweets_transformed$isRetweet == "FALSE",]

##Cleaning the tweet data : Remove punctuation, digits, urls, tab spaces, duplicates
allTweets$text = gsub("[[:punct:]]", "", allTweets$text)
allTweets$text = gsub("[[:digit:]]", "", allTweets$text)
allTweets$text = gsub("http\\w+", "", allTweets$text)
allTweets$text = gsub("[ \t]{2,}", "", allTweets$text)
allTweets$text = gsub("^\\s+|\\s+$", "", allTweets$text)
allTweets$text = gsub("[^[:alnum:]///' ]", "", allTweets$text)
#allTweets$text = gsub("\B[@#]\S+\b/", "", allTweets$text)
allTweets=allTweets[!duplicated(allTweets$text),]
allTweets$text = gsub("[<*>]", "", allTweets$text)
##Please do not Include – Commented


## Actual Code
##Extract the data from saved csv file
allTweets <- read.csv("Final_Data.csv")

##Exclude stop words from the tweets
exclude=scan("New Text Document (3).txt", what="", sep="\n")
stopwords=c(stopwords("en"), exclude)

##Method for Stem Completion
stemCompletion_mod <- function(x,dict=Corpus_copy) {
  PlainTextDocument(stripWhitespace(paste(stemCompletion(unlist(strsplit(as.character(x),"
")),dictionary=dict),sep="", collapse=" ")))
}
```

The team has faced issues converting the entire tweet data into a corpus and generating the document matrix. So, the team decided to generated the matrix, that will further be used in model building, in six phases, one for each data file.

```
allTweet1=read.csv("data1.csv")

twitterCorpus1 <- Corpus(VectorSource(allTweet1$text))
```

```
twitterCorpus1 <- tm_map(twitterCorpus1,removeWords,stopwords)
twitterCorpus1 <- tm_map(twitterCorpus1, content_transformer(tolower))
Corpus_copy <- twitterCorpus1
twitterCorpus1 <- tm_map(twitterCorpus1, stemDocument)
twitterCorpus1 <- lapply(twitterCorpus1, stemCompletion_mod)
twitterCorpus1 <- as.VCorpus(twitterCorpus1)

####################################################################################

allTweet2=read.csv("data2.csv")

twitterCorpus2 <- Corpus(VectorSource(allTweet2$text))
twitterCorpus2 <- tm_map(twitterCorpus2,removeWords,stopwords)
twitterCorpus2 <- tm_map(twitterCorpus2, content_transformer(tolower))
Corpus_copy <- twitterCorpus2
twitterCorpus2 <- tm_map(twitterCorpus2, stemDocument)
twitterCorpus2 <- lapply(twitterCorpus2, stemCompletion_mod)
twitterCorpus2 <- as.VCorpus(twitterCorpus2)

####################################################################################
###############

allTweet3=read.csv("data3.csv")

twitterCorpus3 <- Corpus(VectorSource(allTweet3$text))
twitterCorpus3 <- tm_map(twitterCorpus3,removeWords,stopwords)
twitterCorpus3 <- tm_map(twitterCorpus3, content_transformer(tolower))
Corpus_copy <- twitterCorpus3
twitterCorpus3 <- tm_map(twitterCorpus3, stemDocument)
twitterCorpus3 <- lapply(twitterCorpus3, stemCompletion_mod)
twitterCorpus3 <- as.VCorpus(twitterCorpus3)

####################################################################################
###############
```

```
allTweet4=read.csv("data4.csv")

twitterCorpus4 <- Corpus(VectorSource(allTweet4$text))
twitterCorpus4 <- tm_map(twitterCorpus4,removeWords,stopwords)
twitterCorpus4 <- tm_map(twitterCorpus4, content_transformer(tolower))
Corpus_copy <- twitterCorpus4
twitterCorpus4 <- tm_map(twitterCorpus4, stemDocument)
twitterCorpus4 <- lapply(twitterCorpus4, stemCompletion_mod)
twitterCorpus4 <- as.VCorpus(twitterCorpus4)
############################################################################################
############

allTweet5=read.csv("data5.csv")

twitterCorpus5 <- Corpus(VectorSource(allTweet5$text))
twitterCorpus5 <- tm_map(twitterCorpus5,removeWords,stopwords)
twitterCorpus5 <- tm_map(twitterCorpus5, content_transformer(tolower))
Corpus_copy <- twitterCorpus5
twitterCorpus5 <- tm_map(twitterCorpus5, stemDocument)
twitterCorpus5 <- lapply(twitterCorpus5, stemCompletion_mod)
twitterCorpus5 <- as.VCorpus(twitterCorpus5)
############################################################################################
############

allTweet6=read.csv("data6.csv")

twitterCorpus6 <- Corpus(VectorSource(allTweet6$text))
twitterCorpus6 <- tm_map(twitterCorpus6,removeWords,stopwords)
twitterCorpus6 <- tm_map(twitterCorpus6, content_transformer(tolower))
Corpus_copy <- twitterCorpus6
twitterCorpus6 <- tm_map(twitterCorpus6, stemDocument)
twitterCorpus6 <- lapply(twitterCorpus6, stemCompletion_mod)
twitterCorpus6 <- as.VCorpus(twitterCorpus6)
############################################################################################
############
```

```
##Generating final corpus
corpus <- c(twitterCorpus1, twitterCorpus2, twitterCorpus3, twitterCorpus4, twitterCorpus5, twitterCorpus6)

#Creates Document Matrix
termDocumentMatrix <- TermDocumentMatrix(corpus, control=list(wordLengths=c(1,Inf)))
transposeTDM <- t(termDocumentMatrix) #transpose
transposeTDM <- removeSparseTerms(transposeTDM, 0.999)

documentMatrix <- as.matrix(transposeTDM)

##Remove highly correlated features
correlatedMatrix <- cor(documentMatrix)
correlatedMatrix[upper.tri(correlatedMatrix)] <- 0
diag(correlatedMatrix) <- 0
documentMatrix <- documentMatrix[,!apply(correlatedMatrix,2,function(x) any(x > 0.5 | x < -0.5))]

#Generate training and testing data
train <- sample(nrow(documentMatrix), 70*nrow(documentMatrix)/100)
trainData <- documentMatrix[train,]
testData <- documentMatrix[-train,]
response <- as.matrix(allTweets$response)
trainResponse <- response[train,]
testResponse <- response[-train,]

#Model Building

##SVM
model=svm(trainData,trainResponse,type = "C-classification", kernel = "polynomial",gamma = 0.1, cost = 10)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)
```

```
model=svm(trainData,trainResponse,type = "C-classification", kernel = "polynomial",gamma = 0.1, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "polynomial",gamma = 1, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "polynomial",gamma = 10, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "linear",gamma = 10, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "linear",gamma = 100, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "linear",gamma = 1000, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "linear",gamma = 10000, cost = 10)
```

```
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "linear",gamma = 100000, cost = 10)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "linear",gamma = 100000, cost = 100)
pred=predict(model, testData)
confusionMatrix(pred, testResponse)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "radial",gamma = 10, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)



model=svm(trainData,trainResponse,type = "C-classification", kernel = "radial",gamma = 100, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "radial",gamma = 1, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "radial",gamma = 0.1, cost = 100)
pred=predict(model, testData)
```

```
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "radial",gamma = 0.01, cost = 100)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "radial",gamma = 0.01, cost = 10)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "radial",gamma = 0.001, cost = 1)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)

model=svm(trainData,trainResponse,type = "C-classification", kernel = "radial",gamma = 0.0001, cost = 10)
pred=predict(model, testData)
table(predict=pred,truth=testResponse)
mean(pred==testResponse)
confusionMatrix(pred, testResponse)

##Naive Bayes
model=naiveBayes(trainData,trainResponse,laplace = 0, threshold = 0.9, eps = 1)
pred=predict(object = model,newdata = testData, type="raw")

response <- rep(NA, nrow(pred))
for(i in 1:length(response))
   response[i] <- names(which(pred[i,] == max(pred[i,])))
table(response, testResponse)
```

```
mean(response[!is.na(response)]==testResponse[!is.na(response)])

##Using RTextTools

documentMatrix <- create_matrix(allTweets$text, language="english", removeNumbers=TRUE, stemWords = TRUE, removeSparseTerms = .998, removeStopwords = TRUE, minWordLength = 3, toLower = TRUE, removePunctuation = TRUE)
container <- create_container(documentMatrix, allTweets$response, trainSize=1:7839, testSize=7840:11198, virgin=FALSE)

##Model Building
SVM_TRAIN <- train_model(container,"SVM")
GLMNET_TRAIN <- train_model(container,"GLMNET")
MAXENT_TRAIN <- train_model(container,"MAXENT")
SLDA_TRAIN <- train_model(container,"SLDA")
BOOSTING_TRAIN <- train_model(container,"BOOSTING")
BAGGING_TRAIN <- train_model(container,"BAGGING")
RF_TRAIN <- train_model(container,"RF")
NNET_TRAIN <- train_model(container,"NNET")
TREE_TRAIN <- train_model(container,"TREE")

##Classification
SVM_CLASSIFY <- classify_model(container, SVM_TRAIN)
GLMNET_CLASSIFY <- classify_model(container, GLMNET_TRAIN)
MAXENT_CLASSIFY <- classify_model(container, MAXENT_TRAIN)
SLDA_CLASSIFY <- classify_model(container, SLDA_TRAIN)
BOOSTING_CLASSIFY <- classify_model(container, BOOSTING_TRAIN)
BAGGING_CLASSIFY <- classify_model(container, BAGGING_TRAIN)
RF_CLASSIFY <- classify_model(container, RF_TRAIN)
NNET_CLASSIFY <- classify_model(container, NNET_TRAIN)
TREE_CLASSIFY <- classify_model(container, TREE_TRAIN)

##Cross Validation
SVM_PRED <- cross_validate(container, 4, "SVM")
GLMNET_PRED <- cross_validate(container, 4, "GLMNET")
MAXENT_PRED <- cross_validate(container, 4, "MAXENT")
```

```
SLDA_PRED <- cross_validate(container, 4, "SLDA")
BAGGING_PRED <- cross_validate(container, 4, "BAGGING")
BOOSTING_PRED <- cross_validate(container, 4, "BOOSTING")
RF_PRED <- cross_validate(container, 4, "RF")
NNET_PRED <- cross_validate(container, 4, "NNET")
TREE_PRED <- cross_validate(container, 4, "TREE")
```

```
##Word Cloud
documentMatrix <- as.matrix(documentMatrix)
sortMatrix <- sort(colSums(documentMatrix),decreasing=TRUE)
wordCloudFrame <- data.frame(word = names(sortMatrix),freq=sortMatrix)
```

```
theme <- brewer.pal(8,"Dark2")
wordcloud(wordCloudFrame$word,wordCloudFrame$freq,         scale=c(8,.3),min.freq=2,max.words=100,
random.order=T, rot.per=.15, colors=theme, vfont=c("sans serif","plain"))
```

```
##Sentiment Plot
qplot(allTweets$response, xlab = "Sentiment of Tweets")
```

#From this we can observe that there was an overall positive sentiment in the tweets, but it is also important to note that many of the words are neutral. Thus, a good feature selection algorithm would help reduce terms that do not contribute to either positive or negative sentiment in a tweet.

##Testing with new Data. Create a corpus with the data using the aforementioned steps. Use below steps to create a test matrix and test the prediction accuracy.

```
tdmTest=TermDocumentMatrix(twitterCorpus4, control=list(wordLengths=c(1,Inf)))
mdtTest=t(tdmTest)
documentMatrixTest=as.matrix(mdtTest)
```

```
new=setdiff(colnames(documentMatrix),
intersect(colnames(documentMatrix),colnames(documentMatrixTest)))
xx=data.frame(documentMatrixTest[,intersect(colnames(documentMatrix),colnames(documentMatrixTest)
)])
a=data.frame(matrix(0, nrow=nrow(xx), ncol=ncol(documentMatrix)-ncol(xx)))
names(a) <- new
new_dataset=cbind(a,xx)
```

~~~
new_resp=as.matrix(allTweet4$response)
pred=predict(model,new_dataset)
mean(pred==allTweet4$response)

##Pruning the Tree
prune.tweet <- prune.misclass(TREE_TRAIN, best = 3)
plot(prune.tweet)
text(prune.tweet, pretty = 0)
plot(TREE_TRAIN)
text(TREE_TRAIN, pretty = 0)
prune.tweet <- prune.misclass(TREE_TRAIN, best = 5)
plot(prune.tweet)
text(prune.tweet, pretty = 0)
TREE_CLASSIFY <- classify_model(container, prune.tweet)
TREE_PRED <- cross_validate(container, 4, "TREE")

#For Cross Validation in SVM with varying parameters
meth=c("polynomial","linear", "radial")
gam=c(0.0001,0.01,1,10)
cos=c(1,10,100)
for (m in meth){
  for (g in gam){
    for (c in cos){
      b=paste("for method=",m," gamma=",g,"cost=",c)
      print(b)
      SVM_CV <- cross_validate(container, 3, "SVM",method = "C-classification", kernel = m,cross = g, cost = c)
    }
  }
}
```
~~~