Domain classification and sentiment analysis of twitter tweets

Adarsh Kashyap, Amit Gupta, Gayathri Shivakumar, Kanishta Agarwal

Abstract— **In recent years, the interest among the research community in sentiment analysis has grown exponentially. This is largely supported by the huge number of scientific publications and forums or related conferences. The sharp rise in Twitter usage has made it a very important platform for this kind of analysis. A wide range of features and methods for training sentiment classifiers for Twitter datasets have been researched over years with varying results, but very little focus is laid upon the sentiment analysis with respect to the domain of the data. It is for this reason that this paper aims to analyze the performance of classifiers in sentiment analysis along with domain classification.**

I. INTRODUCTION

Sentiment analysis refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. It is a growing area with research ranging from document level classification (Pang and Lee 2008) to learning the polarity of words and phrases (Esuli and Sebastiani 2006). It is widely applied to microblogging websites and social media for a variety of applications, ranging from marketing to customer service. Domain Classification, on the other hand, refers to characterization of the data on the basis of its type. It involves segregating data based on what it describes rather than the sentiment involved. Domain classification combined with Sentiment Analysis proves to be very powerful in identifying the sentiment of users pertaining to a particular area, thus giving us deep insights into the trends prevailing in that sector. Not only does this emerge as a mechanism to review new products, it also helps in enhancing the marketing strategies and capturing the advancements in the corresponding industry.

Microblogging websites have evolved to become a source of this kind of varied information which can undergo domain classification and sentiment analysis to provide our necessary results. This is due to nature of microblogs on which people post real time messages about their opinions on a variety of topics, discuss current issues, complain, and express positive sentiment for products they use in daily life. In fact, companies manufacturing such products have started to poll these microblogs to get a sense of general sentiment for their product. Many a times these companies study user reactions and cater to their needs in respective fashion. One major bottleneck is to build technology to detect and summarize an overall sentiment in a particular domain. Given the character limitations on tweets, classifying the sentiment of Twitter messages is most similar to sentence-level sentiment analysis (Yu and Hatzivassiloglou 2003); however, the informal and specialized language used in tweets, as well as the very nature of the microblogging domain make domain classification and sentiment analysis a herculean task.

The domains available are diverse and for representation purposes, we chose two popular and significant domains- Sports and Politics. Not only are these domains popular, they are also one of the few domains which affect the masses to a large extent. Analyzing the Sports domain can help us in formulating better strategies for conducting events or even in understanding the interests of the audience. The politics domain paints a picture of the political situation of a country and can help in taking major decisions with regards to the policies pertaining to the people.

Classifiers are used to differentiate the data on any specific basis, be it sentiment or domain. We use Naïve Bayes classifier and Support Vector Machines for this purpose. Naïve Bayes is the most commonly used baseline classifier. Support Vector Machine is comparatively an advanced classifier. After training our classifiers appropriately, we pass our data, which is pre-processed, by removing stop words and unwanted words, to a space effective, probabilistic data structure Bloom Filter.

The filtered data is then passed to the trained Naïve Bayes and Support Vector Machine classifiers to segregate them on the basis of domain and sentiment. This, thus, helps us in achieving our aim of classifying the data on the basis of sentiment and domain and also allows analysis of the accuracy of such classification. We compare and contrast both the classifiers and conclude on the best classifier for our requirement. Let us understand

the complete details of the project in the further sections described below.

II. BACKGROUND

The last few years have experienced a huge growth in the use of microblogging platforms such as Twitter. Just in the past year there have been a number of papers looking at Twitter sentiment and buzz (Jansen et al. 2009; Pak and Paroubek 2010; O'Connor et al. 2010; Tumasjan et al. 2010; Bifet and Frank 2010; Barbosa and Feng 2010; Davidov, Tsur, and Rappoport 2010). Other researchers have begun to explore the use of part-of-speech features but results remain mixed. Spurred by the tremendous growth and importance, companies and media organizations are increasingly seeking ways to mine Twitter for information about what people think and feel about their products and services. Companies such as Twitratr (twitrratr.com), tweetfeel (www.tweetfeel.com), and Social Mention (www.socialmention.com) are just a few who advertise Twitter sentiment analysis as one of their services. Tools like Tweetstat and google analytics are also working in this area. While there has been a fair amount of research on how sentiments are classified, there hasn't been much emphasis on domain specific sentiment analysis. We intend to focus on this area of domain classification combined with sentiment analysis to get a clear picture of trends in a certain area.

III. DATA

For the domain classification, we trained the Naïve Bayes classifier and the Support Vector Machine classifier in layer one with around 500MB of data. Then, we used a humongous 32,50,000 tweets for training our Sentiment classifier. This amounts to about 600MB of data for training the Naïve Bayes classifier and the Support Vector Machine classifier in layer two. Furthermore, we have used 10,000 features in our process of domain classification and sentiment analysis. Finally, we used approximately 400,000 tweets as the test data to find our final accuracy and compare our classifiers on that basis.

IV. METHOD

FETCHING AND CLEANING DATA:

We rely on the Twitter streaming API, Tweepy,to download twitter messages in real time. It is useful for obtaining a high volume of tweets, or for creating a live feed using a site stream or user stream. The streaming API pushes messages to a persistent session. This allows the streaming API to download more data in real time than could be done using the REST API.In Tweepy, an instance of tweepy.Stream establishes a streaming session and routes messages to StreamListener instance. The on_data method of a stream listener receives all messages and calls functions according to the message type.Streams not terminate unless the connection is closed, blocking the thread.

First, the data obtained from the Tweepy is cleaned before training the classifiers or predicting the classifications for the new incoming tweets. Initially, stop words that do not add any meaning to the sentiment of the tweet are pruned. Then, symbols like '#' and '@' from the hashtags and user tags that are present in tweets are removed. Then, special characters are trimmed, blank spaces at the beginning and end of the words are cleaned. Finally, slang words are converted into actual words. For instance, 'Hapyyyyy' gets converted to 'Happy'. We retain emoticons as they play a crucial role in depicting the sentiment of the tweet. We have added many more cleaning techniques to modify the data which can be optimally used for training our classifier.

TRAINING OF CLASSIFIERS:

We have implemented a dual layered classifier approach wherein we have 2 instances of Naïve Bayes classifier and 2 instances of Support Vector machine in each layer, 1 of each for Domain classification and the other for Sentiment analysis. The preprocessed data is passed through these layers of classifiers.

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. It is a popular baseline method for text categorization, the problem of judging documents as belonging to one category or the other with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem.

Support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. We also include TF-IDF (Term Frequency- Inverse Document Frequency) as a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The TF-IDF value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust

for the fact that some words appear more frequently in general.

FILTERING AND PREDICTION:

We initially filter our data using a probabilistic data structure called Bloom Filter. It supports two functions at its core: add() and contains(). As the name suggest add() function is used to add the element to bloom filter and contains() is used to test whether filter contains the particular element or not. The contains () function is not always 100% accurate., thus leading to false positives. If contains () returns false, you can be sure that this element is not present in the data structures but if it returns true, there are chances that the element is actually not present in bloom filter. Figure 1.0 shows how data is mapped to bits in the bit array.

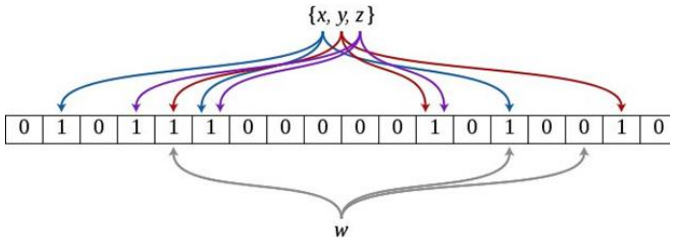

Fig.1.0: Representation of Bloom Filter

PREDICTION:

After filtering the data using Bloom filter we then pass it through our dual layer classifiers. Initially, the Naïve Bayes and Support Vector Machine classifiers in the first layer classify the data based on the domain. We have chosen Sports and Politics as our domains. So, the data which is filtered through is passed through the first layer of classifiers and is segregated into “Sports” and “Politics” domains respectively. This data is then passed through our next layer of classifier which analyses our data based on sentiment. The second layer classifies the domain classified data based on the sentiment.

We initially opted for Naïve Bayes owing to the fact that it is very simple. If the Naïve Bayes conditional independence assumption actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so we need less training data. Even if the Naïve Bayes assumption doesn’t hold, a Naïve Bayes classifier still does a great job in practice. It is definitely a good bet if want something fast and easy. But, Naïve Bayes can’t learn interactions between features. Another useful and effective classifier is the Support Vector Machine. It is highly accurate and with an appropriate kernel it can work well even if our data isn’t linearly separable in the base feature space. It is very popular in text classification problems where very high-dimensional spaces are the norm. It is memory intensive and hard to interpret.

We have used Spark as the data streaming platform and we are making use of the libraries Spark provides like MLib which contains the above mentioned classifiers. The MLib (pronounced "M-Lib") is a C software framework that significantly simplifies the development of highly reliable system-level libraries and programs (like daemons/services). It is small, efficient, robust and highly portable, and was designed primarily for embedded systems. The MLib is re-entrant and doesn't use static writable data, which makes it ideal for a wide array of embedded systems. It is possible to use several streams implementations concurrently and it also provides useful containers like hash tables, AVL trees, string dictionaries etc. in order to ease the development of sophisticated algorithms.

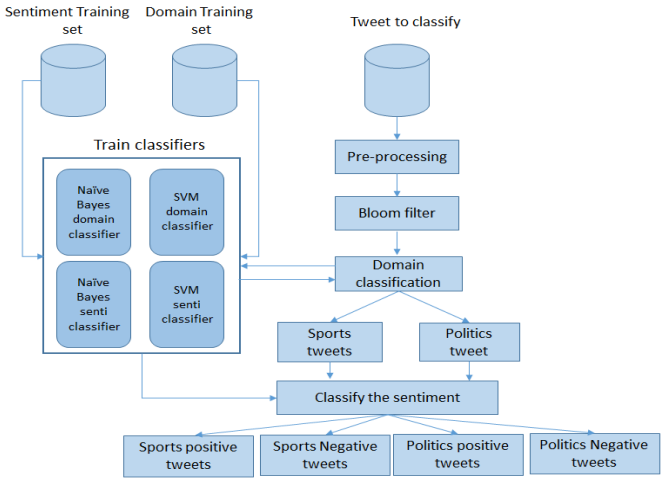

Fig.2.0: block diagram of the methods and flow of the project

As the flow chart depicts, the data we have is tested against the classifiers we have mentioned. Each of the classifier behaves in a different manner based on the data set and its underlying characteristic. We obtain varying accuracy for each of them and based on the difference, we understand the nuances of the classifiers and the extent to which our classification is correct.

V. RESULTS

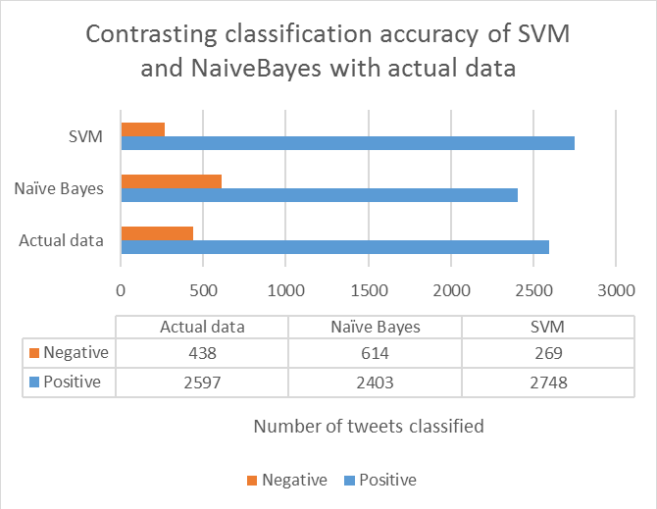

Fig.3.0: Small Data Set

We initially use a small data set where we know the classification details. We then test our trained classifiers on this data set. As we can observe from Fig.3.0, our classifiers fare moderately in comparison to the actual result owing to the small size of the data set.

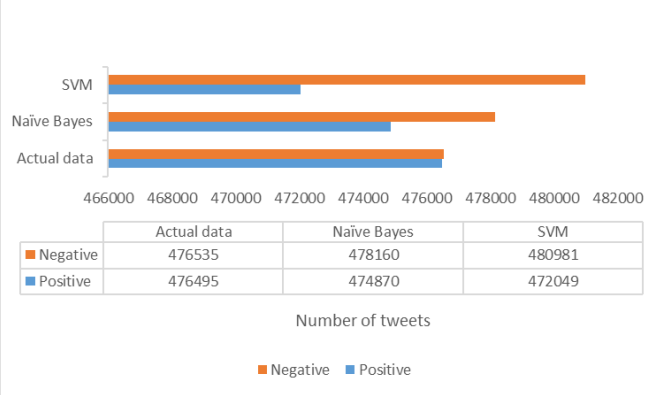

Fig.4.0: Large Data Set

We then take a larger corpus to train our classifiers, a data set which is nearly 300 times our small data set. As we can observe from Fig 4.0, our classifiers perform much better than earlier. We are able to classify the data much accurately.

```
Comparing the classifiers
-------------------------

Number of training tweets          :  6983
Number of test tweets              :  3017

                          Negative sentiment      Positive sentiment
                          ------------------      ------------------
Actual data             :       2579                    438
Naive predicted data    :       2403                    614
SVM predicted data      :       2748                    269

Naive Bayes classifier accuracy     :  79.11%
SVM classifier accuracy             :  87.63%
```

Fig.5.0: Accuracy comparison with small data set

Fig 5.0 portrays the accuracy of our classifiers with the small data set. We have trained on around seven thousand tweets and have tested the trained classifiers on approximately three thousand tweets. SVM classifier overshoots the performance of Naïve Bayes classifier by a large margin indicating that the former performs better with the limited data set.

```
Comparing the classifiers
-------------------------
Number of training tweets          :  2225956
Number of test tweets              :  953030

                          Negative sentiment      Positive sentiment
                          ------------------      ------------------
Actual data             :       476495                  476535
Naive predicted data    :       474870                  478160
SVM predicted data      :       472049                  480981

Naive Bayes classifier accuracy     :  74.19%
SVM classifier accuracy             :  75.48%
```

Fig.6.0: Accuracy comparison with large data set

We also compare the accuracy of our classifiers with the larger corpus we possess. As Fig.6.0 illustrates, Naïve Bayes performs nearly as well as SVM, though not better. The marginal difference in accuracy also points to the fact that Naïve Bayes classifier predicts more accurately when trained with huge data.

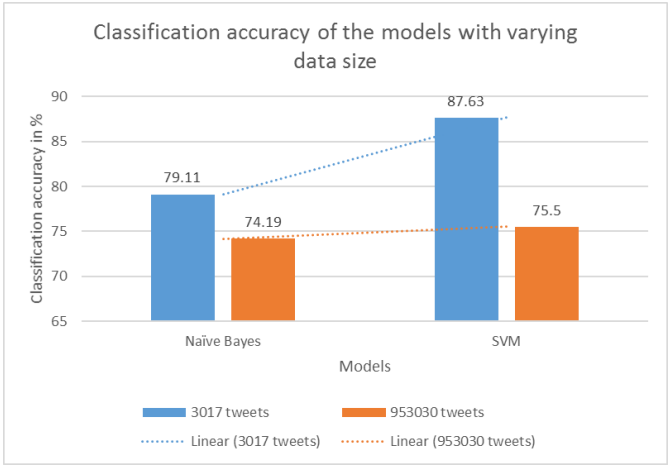

Fig.7.0: Comparison of classification accuracies

The classifiers are compared and contrasted based on the varying size of the data set. As Fig 7.0 illustrates, there is a huge difference in the accuracy prediction by Naïve Bayes classifier and SVM classifier in case of a small data set. This is not the case with large data set, where in there is no observable difference between the accuracies of the classifiers.

```
Domain Classification statistics
--------------------------------

Total Number of tweets classified   :  95410

Naive Bayes Classifier -
    Number of sports tweets         :  49840
    Number of politics tweets       :  45570

SVM Classifier -
    Number of sports tweets         :  32189
    Number of politics tweets       :  63221

Sentiment classification of sports tweets by
NaiveBayes classifier :

Total Number of tweets classified   :  49840

Sentiment Classification statistics
-----------------------------------
    Total number of Positive tweets :  26176
    Total number of Negative tweets :  23664

Sentiment classification of politics tweets by
 NaiveBayes classifier :

Total Number of tweets classified   :  45570

Sentiment Classification statistics
-----------------------------------
    Total number of Positive tweets :  18780
    Total number of Negative tweets :  26790

Sentiment classification of sports tweets by
SVM classifier :

Total Number of tweets classified   :  32189

Sentiment Classification statistics
-----------------------------------
    Total number of Positive tweets :  18376
    Total number of Negative tweets :  13813

Sentiment classification of politics tweets by
SVM classifier :

Total Number of tweets classified   :  63221

Sentiment Classification statistics
-----------------------------------
    Total number of Positive tweets :  28367
    Total number of Negative tweets :  34854
```

Fig.8.0: Statistical Analysis

We end our study with a complete statistical analysis of our model. We begin with Domain Classification, which takes place in Layer 1 of our classifiers followed by the Sentiment Analysis in Layer 2. Our model classifies the data into Sports and Politics. Then it further evaluates the tone of the tweets and segregates them into positive and negative sentiment.

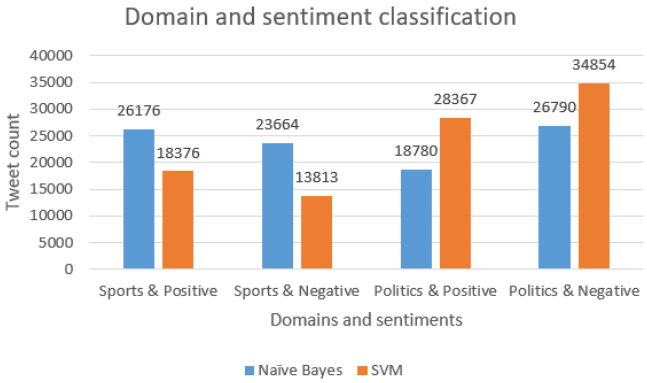

Fig.9.0: Statistical Analysis Representation

The bar graph representation in Fig.9.0 paints a clear picture about our observations of Domain Classification and Sentiment Analysis through Naïve Bayes classifier and Support Vector Machine classifier. We can clearly note that the trends observed are similar for both our classifiers, that is, the relative sentiment in each domain is same for each of the classifiers, thus we can conclude that both the classifiers have performed reasonably in this task of classification.

VI. CONCLUSION

We started this project with the goal of understanding the intricacies of classifiers, filters, data steaming platforms and other concepts learnt in the course. We believe we have achieved it and it is clearly reflected in the outcome of the project. For the domain classification and sentiment classification we observe that both the type of classifiers used have performed reasonably well, though Naïve Bayes falls behind while classifying the data when there is less training set, it performs on par with SVM when we have very large data. From the result obtained we could infer that the outcome of the domain classification was not accurate as the words used as the filter by the bloom filter was very limited and ambiguous and because of the limited training data set we had for domain classification. Thus, we can conclude that the sentiment classification gave us satisfactory results owing to the aforementioned constraints.

VII. ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to Professor Andrew Schwartz for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in understanding the concepts involved in Big Data Analytics with ease. Besides we would also like to thank the teaching assistant Youngseo Sen for his continuous support throughout the course. This course has been an amazing learning experience for each one of us.

VIII. REFERENCES

http://help.sentiment140.com/for-students

http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/

https://spark.apache.org/docs/1.6.0/api/python/pyspark.mllib.html

http://docs.tweepy.org/en/v3.5.0/

http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf

http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/viewFile/2857/3251

http://www.kdnuggets.com/2016/08/gentle-introduction-bloom-filter.html