Sentiment Analysis of Microblogs

Vipul Pandey and C.V.Krishnakumar Iyer

***Abstract*—In this project we attempt to perform sentiment based classification of Micro-blogs using Machine Learning techniques. Sentiment Analysis of short messages posted on Micro-blogging tools can be helpful in determining the current usability and acceptance of any target product or service. It can help in raising alarms in the wake of sudden shifts in user sentiments or attitude towards the service. We present a system that can read messages relevant to a particular topic from a micro-blogging service such as Twitter, analyze the messages for the sentiments they carry and classify them as *neutral*,*positive* or *negative*. We try out different feature selection and classification algorithms in our search for the best combination. We also evaluate different strategies in order to get the best performance over the imbalanced dataset - caused by relatively infrequent changes in user sentiments - and to understand the underlying nature of this problem.**

I. Introduction

Sentiment or Opinion Mining has been an active area of research in academia because of the challenges that it poses. It is also a vital question that is sought in the industry as it gives an insight into the consumers' mind, the influences he is subject to, and his decision making process - besides being an explicit feedback about the performance of any widely used and talked about product, service, event or a phenomenon.

In our efforts, we try to detect the current attitude of the users towards an online service, the model that can be extended to other services or products without any difficulty whatsoever.

There has been a widespread interest in the area of Opinion mining and Sentiment Analysis because of the challenges it offers and its potential applicability. [2] gives an excellent survey of the recent developments and the work done in this field. [3] performs a work that is very similar to ours in that it seeks to identify market sentiments. They develop hybrid methods for extracting opinions in an automated manner from the discussions on the stock message boards. However, they report a maximum accuracy of 67.58% (Low Ambiguity, Test Set size = 290, Training Set Size = 913). [4] details different feature-selection metrics for text classification that we plan to apply in our work.[5] consider a classification problem of discriminating in between the positive and negative sentiments, and test their approaches using Naive Bayes Classifiers, Max. Entropy Classifiers and SVMs. They operate on the movie reviews dataset, and report a maximum accuracy of 82.9% with SVMs using the presence/absence of features. In addition, they also study the effects of a few variations in the parameters, just as we do.

II. Approach

We follow a simple approach of chaining two Classifiers to classify messages into the three desired categories. The first Classifier discriminates in between neutral messages and polar (negative or positive) ones. The messages deemed polar by this Classifier are then forwarded to the second Classifier in the scheme, which is trained to distinguish in between positive and negative sentiments. The overall scheme is depicted in the Fig-1.

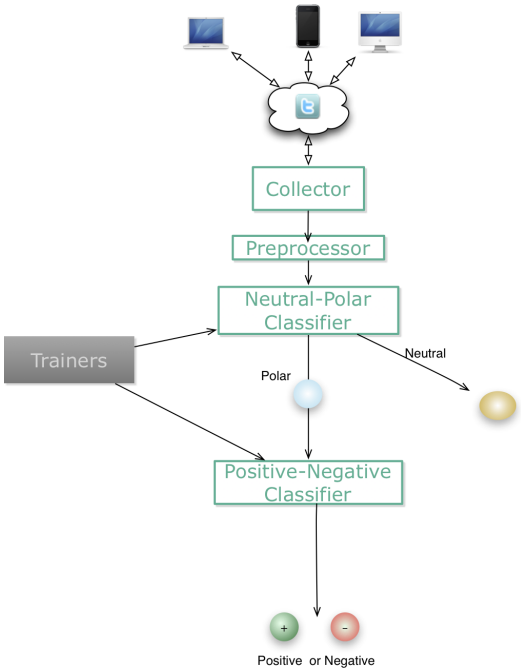

Fig. 1. Components of the System

Following operations are performed in the system to achieve the final results :

- Data Collection - in which data is collected from the source.
- Preprocessing - in which collected data is preprocessed for further action.
- Feature Selection - in which a subset of features is selected to reduce dimensionality.
- Model Selection - in which a model is trained using the training set.
- Classification - in which a trained model classifies the newly arriving messages.

We developed most of the infrastructure in-house although we use [1]Weka framework for standard machine learning algorithms for prototyping and evaluation.

III. Data

We primarily focus on messages obtained from *Twitter* for prototyping and benchmarking but the system can be easily extended towards other sources as well. We use the Twitter Search API to obtain the 'tweets' mentioning the product that we focus on. We collected over 25000 such messages and manually classified them as *negative,*

mild negative, neutral, mild positive and positive. While manually classifying the data we realized that the data is imbalanced in that there are significantly more neutral messages than the polar ones. We also noticed that the mild-negative and mild-positive messages, though having some trace of polarity, either had a large overlap with neutral messages or were bipolar thus adding to the ambiguity. Therefore in our exercise we primarily focus only on *Neutral, Positive* and *Negative* messages by ignore the ambiguous ones while training the classifier.

IV. Preprocessing

As soon as the data is collected, it is passed through a series of preprocessors that aid in the conversion of the message strings into the feature vector, that is used by the feature selection algorithm and eventually by the classifier. The preprocessing step also performs the important task of extracting the relevant signals from the messages, while leaving out the irrelevant ones. In our feature extraction process we not only consider the common text based features as used in traditional Information Retrieval tasks but also use several domain specific features. This assumes a greater significance in our domain of micro-blogs since the text, by its nature is short and hence we *must* depend on many other features to identify the polarity in the message and the type of polarity that it exhibits.

Few of the preprocessors and the preprocessing steps that we follow are explained with examples below. This is one of the more important steps in the entire classification process since the quality of the features that we extract ultimately decides the performance of the classifiers. Let us consider a running example :

```
@cvkkumar:The iTunes Store REEAAAALLY ROCKSS!!
 :) http://bit.ly/madeup
```

A. All Caps Identification

This forms the first filter in our preprocessing step. Since we operate on the micro-blogging space, we observed that it was very common to use all capital words to represent powerful emotions, and hence are a good indicator of the polarity of the message.

```
@cvkkumar:The iTunes Store REEAAAALLY ROCKSS!!
  <>HASALLCAPS<>:) http://bit.ly/madeup
```

B. Lower Casing

Having already exploited the fact of the *all caps*, we fold the case of all the terms in the message to have a consistent casing. This is extremely important for us due to the erratic casing, often found in microblogs.

```
@cvkkumar:the itunes store reeaaally rockss!!
<>HASALLCAPS<> :) http://bit.ly/madeup
```

C. URL Extraction

Most of the microblogs often point to much more elaborate sources of information as a means of sharing content. Since URLs are unique, and it would be too expensive to crawl URLs for their content, we abstract the links to a single keyword to avoid feature explosion.

```
@cvkkumar:the itunes store  reeaaally rockss!!
<>HASALLCAPS<> :) <>URL<>
```

D. Detection of Pointers

We define *Pointer* as the means of addressing a user inside a microblog. In Twitter, pointers take the form of *@user* , where *user* is the username. Again, to avoid explosion of features, we abstract it to a constant symbol.

```
<>PTR<>:the itunes store reeaaally rockss!!
<>HASALLCAPS<> :) <>URL<>
```

E. Identification of Emoticons

Emoticons are one of the most powerful signals that we use in our system to differentiate polar from non-polar messages and positive from negative messages. We identify a range of emoticons and substitute the keyword SMILE if it is positive and FROWN otherwise.

```
<>PTR<>:the itunes store reeaaally rockss!!
<>HASALLCAPS<> <>SMILE<> <>URL<>
```

F. Identification of Punctuations

The punctuations are also a potential source of insight into the polarity of the message. For instance, exclamation marks are used to convey powerful emphasis which may correlate to polar messages, as was observed. In this step we remove the irrelevant punctuations as well so as to avoid redundant noise in our feature set.

```
<>PTR<>:the itunes store reeaaally rockss <>EX<>
 <>HASALLCAPS<> <>SMILE<> <>URL<>
```

G. Stop Word Removal

This is a standard Information Retrieval technique to remove words that are highly common (have a high IDF value) and hence do not add substantial value to the classification process. Common words like "A", "An" and "The" fall in this category. In addition, the search term that is bound to be present in every message is also added to this list.

```
<>PTR<>: reeaaally rockss <>EX<>
<>HASALLCAPS<> <>SMILE<> <>URL<>
```

H. Compression of Words

We also realized that people tend to be very informal while writing microblogs and most of the times *elongate* the words to express strong emotions. For example *SSSLLLOOOWWWW* coveys a higher degree of power than *SLOW*. However, both these modifications refer to the same fact. We employ a heuristic in which we compress such words to what we call as *shadows*. A *shadow* of a word is just the compressed form of the word with *at most* one character repeating consecutively twice. e.g. *SSSLLLOOOWWWW* gives out the shadows *sslow*, *sllow*, *sloow*,

sloww & *slow* where the last one is the most compressed or the *shortest* shadow. In our experiments we also compare the inclusion of either original, shortest or all shadows as potential features. We later see that the *shortest* shadows give the best results, an idea that is fairly intuitive.

Our initial empirical analysis of the kinds of feature vectors obtained does seem to agree with the intuitive reasoning behind the incorporation of those features. After the feature extraction and preprocessing, the message is converted to a feature vector that can be used by the Feature Selection algorithms and Classifiers.

V. Model Selection

Our Classification mechanism consists of two Classifiers as depicted in Fig-1. The first one is the *Neutral-Polar Classifier* which is responsible for classifying the incoming message as either "Neutral" or "Polar". The *Polarity Classifier's* job is to classify polar messages as either "Positive" or "Negative". We select the most popular algorithms for feature selection and classification and carry out our evaluation of each possible combination for each Classifier. In our experiments we randomly pick one Feature Selection algorithm from the list below :

- Gain Ratio
- Information Gain
- χ^2
- Symmetrical Uncertainty

and pair it up with a randomly selected classifier to get a model. A model consists of a trained classifier and the feature space - selected by FeatureSet selection algorithm - to which any new message should be translated for classification. We use the Feature Selection algorithm to select only pre-determined number of best features and then *transcribe* the training messages to the selected feature-space. This is followed by training of the classifier with the transcribed training set using hold-out cross validation. The selected feature subset and the trained classifier instance is then forwarded to the respective Classifiers .The five classifiers we choose from - along with their possible configuration values - are listed below :

- SVM [Kernels (RBF, Linear and Polynomial(2)), C (0.5,1,1.5,2)]
- Voted Perceptron [Exponent(1-10)]
- Naive Bayes
- Bayesian Logistic Regression [Prior (Gaussian, Laplacian)]
- AdaptiveBoosting[BaseClassifier (Any of the above), iterations (6,8,10)]

Thus randomly selecting a classifier means selecting one of the five algorithms above and then randomly selecting its possible attributes. e.g. SVM (C = 0.5) with RBF Kernel is one possible choice, AdaBoosting (6 iterations) over Bayesian Logistic Regression with Laplacian prior is another and SVM (C=1.0) with Polynomial Kernel is yet another one.

In our model we treat each message as a bag-of-words and account for only presence or absence of a word in the message. Our initial experiments suggested that capturing the count of times a word appeared in the message didn't help much. It confirms the intuition that the likelihood of a word repeating in micro-blogs is generally low since the messages are themselves very short.

VI. Experiments and Observations

We tried a combination of different feature selection and classification algorithms in our search for the best models for both the Classifiers. We trained over 2500 models for Neutral-Polar (NP) Classifier and an equal number of models for Polarity-Classifier. A self guided 'evaluator' was coded which, at every iteration, randomly selects one of all possible models and trains it over the given samples. Once trained, the models were evaluated against the hold-out cross validation set and the performance statistics were captured in the database. Training time for each iteration depended upon the nature of the feature-set selection algorithm, the classifier and the number of iterations wherever boosting was applied. Training time generally varied from anywhere in between a couple of minutes to a few hours.

While manually trying to train models for classification we realized that the ambiguous messages are playing a key role in confusing the classifiers and making them perform badly. Even the human classifiers were were not so sure about which category to put those ambiguous or not-so-strong messages into. Discarding ambiguous messages from the training set gave a significant improvement in classification accuracy. Our initial manual training also suggested that the optimal feature-set size was close to 1000 features.

A. Models for Neutral-Polar (NP) Classifier

In order to generate the models for the neutral polar classification, there were two major selections to be made.

Feature Selection Algorithm Given a set of features, what is the best way to select the most discriminating features amongst them?

Classifier For the given data set, what is the best model?

A.1 Feature Selection Algorithm Selection

Feature selection is the process of choosing the most discriminative features so as to enable the classifier to *perform* better. This becomes especially relevant in the case of text classification where there are an extremely large number of features. The different feature selection algorithms that we considered along with the average F1 scores of the models that they were associated with is summarized in table below.

Feature Selection Algo	Avg F1
Gain Ratio	0.1994
Information Gain	0.4581
Symmetric Uncertainty	0.4696
χ^2	0.5031

The averages in the above table were calculated over 2500 different models, each of which was evaluated using the hold-out cross validation technique.

A.2 Classifier Selection

We started with individual classifiers and later applied the *boosting* based ensemble methods to deal with imbalance in the data set as suggested by [6], [7]. Imbalance in training set also makes the accuracy measure of a model insignificant hence we focus on the recall, false-positive-rate and F1 measure of each model for comparison. After filtering the ambiguous messages our training-set was reduced to around 2500 polar messages and over 10000 neutral messages.

The observed performance of the classifiers without any *boosting* is given in the following table.

Classifier	Avg F1	Avg Acc.	Max Acc.
BLR(Gauss)	0.432	76.02	80.09
BLR(Laplace)	0.454	78.39	83.00
Naive Bayes	0.4985	78.66	81.55
SVM(Linear)	0.4700	80.03	83.42
SVM(Poly)	0.4536	79.71	81.97
SVM(RBF)	0.0864	78.05	79.4
Voted Perceptron	0.4300	78.64	83.66

It was very astonishing to see SVM with the RBF kernel perform so badly. One reason of this low accuracy could be over-fitting of the model to the training data. Hence we remove the SVM(RBF) from further consideration here.

When boosting is applied along with oversampling of minority samples and undersampling majority samples , we get the following results for the classifiers.

Classifier	Avg F1	Avg Acc.	Max.Acc.
BLR(Gauss)	0.515	80.09	85.51
BLR(Laplace)	0.510	79.98	87.04
Naive Bayes	0.563	75.63	84.18
SVM(Linear)	0.557	80.2	85.44
SVM(Poly)	0.499	79.22	83.37
SVM(RBF)	-	-	-
Voted Perceptron	0.508	78.70	83.62

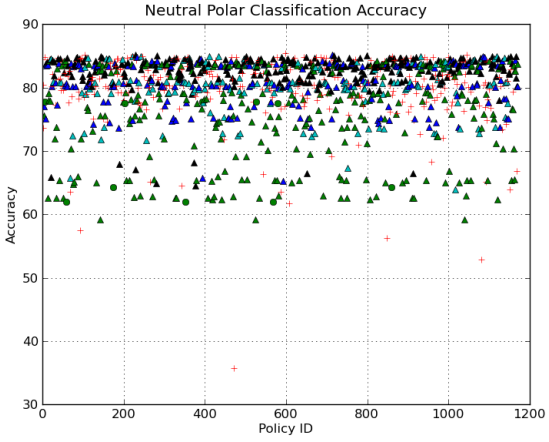

Fig. 2. Accuracy of all models for NP Classifier

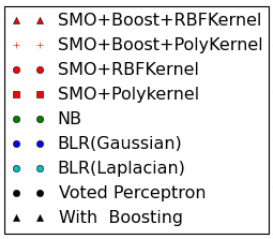

Fig. 3. Legend for Fig-4,2,7,8,9

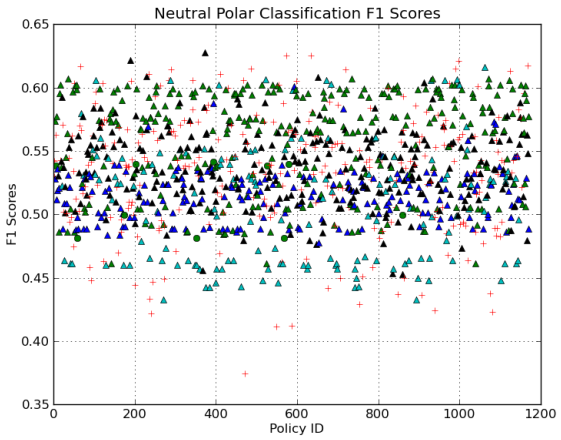

Fig. 4. All Models' F1 Scores - NP Classifier

The Fig-3 represents the legend for Figures 4,2,7,8,9. The Figures 4 and 2 represents the F1 and accuracy scores for all of the NP classifiers that we trained. It is interesting to see how the triangles and '+'s - representing boosted-classifiers - pervade the top portion of the plots. Though maximum accuracies are high, the F1 scores are relatively low since either the recall is low or the false-positive rate is considerably high.

Based on these observations, we can clearly see that though, in both the cases (with and without boosting) the accuracy is high whereas the F1 score is pretty low. This happens because of the data skew that is present in our dataset.

A.3 Actions To Counter Skew and Observations

There are a set of observations that we gathered from these runs of the classifiers.

Of-the-shelf Classifiers : Classifiers with no boosting, oversampling or undersampling gave lower recall with relatively lower false positive rate. While in some cases it may work but generally not being able to identify the minority class with a good success rate is not acceptable - which is true in our case as well.

Boosting: On an average, classifiers with Boosting had a bit of improvement over the classifiers without boosting. This is understandable since boosting is supposed to enhance the accuracy by making a relatively stronger classifier out of a committee of weaker classifiers.

Oversampling with Boosting : Oversampling is a tech-

nique that we attempted to use against the data skew. It involves the duplication of the already existing minority class instances so that the sizes of the classes become comparable. We saw some improvement with this over the scenario where we had just boosting.

Undersampling with Boosting : Undersampling is a technique in which some samples of the majority class are discarded so that the sizes of the classes become comparable. We find that even undersampling gives only a slight improvement over boosting.

Both Undersampling and Oversampling with Boosting : This gives better results than any of the methods given above, and has yet been our most successful tool to counter the skew in the data and address the recall issue. With this we got over 85% recall in some cases though with the cost of a higher false-positive-rate.

SMOTE : SMOTE is proposed by Chawla et al in [9]. It tries to counter the imbalance in the skewed dataset by synthetically generating the samples for minority class. We briefly tried SMOTE but didn't see much improvement.

We can see from the ROC plots in Fig-5 that for the models trained *without* boosting, oversampling or undersampling and with 1000 features the maximum recall attained is close to 65%. The points on the lower left corner of the plot correspond to either feature selection based on Gain Ratio and/or SVM classifier with RBF kernel. Repeating this test for all the possible models with 500 and 1500 features did not give much improvements.

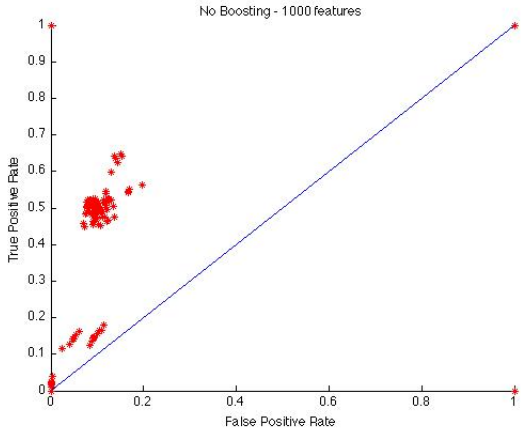

Fig. 5. Classifier Performance - No Boosting, Oversampling or Undersampling

We then applied the boosting techniques to get a better recall for the minority class. We saw that oversampling the polar messages and undersampling neutral ones improved the overall recall of all the possible models but at the same time also increased the false positive rate. As show in Fig-6 oversampling the polar messages to become equivalent to the under-sampled neutral message gave us the highest recall of over 85% with the false-negative rate touching about 30%, which we consider to be very high. NaiveBayes and SMO gave the best recalls. VotedPerceptron and BayesianLogisticRegression always gave less than 65% recall while also giving less than 20% false-positive rate

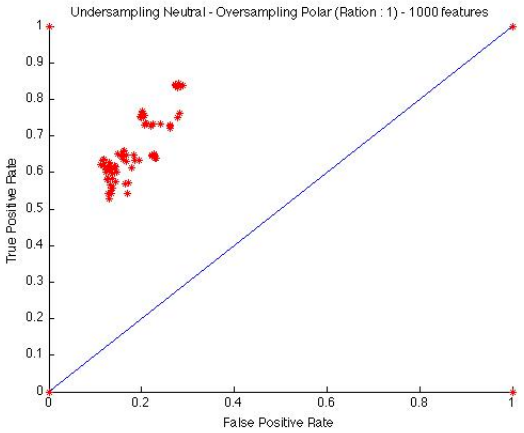

Fig. 6. Classifier Performance - Boosting, Oversampling

B. Models for Polarity Classification

Polarity classification was not plagued with the class imbalance problem in our case. Although we have very small training set - of 2500 samples with almost equal number of positive and negative messages - to train the classifiers with. The resulting models, their accuracies and the F1 Scores can be found in Fig-7 and 8

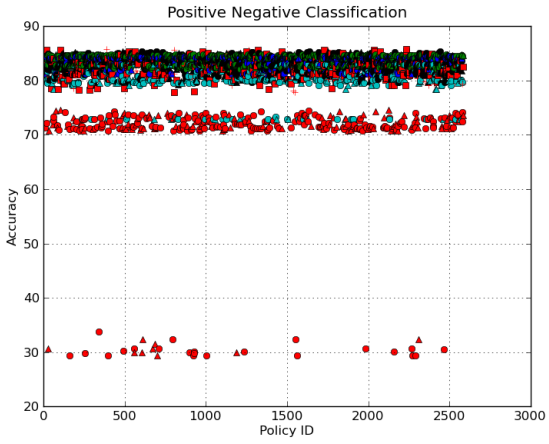

Fig. 7. Accuracy-All Models-Polarity Classifier

We again saw that SVM with RBF kernel is performing poorly and gives high bias towards any one class in each run. As can be see in Fig-10 RBFs either give a very high FalsePositive Rate or a very poor recall.

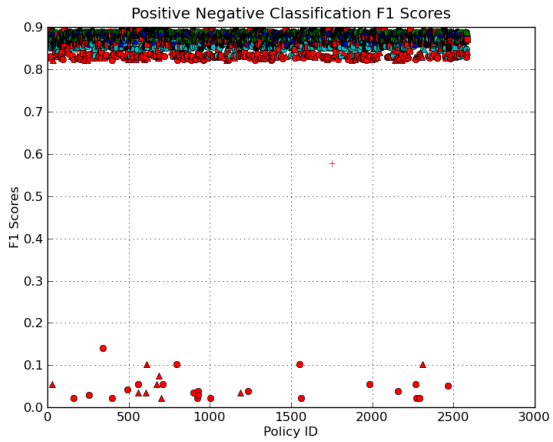

Fig. 8. All F1 Scores-Polarity Classifier

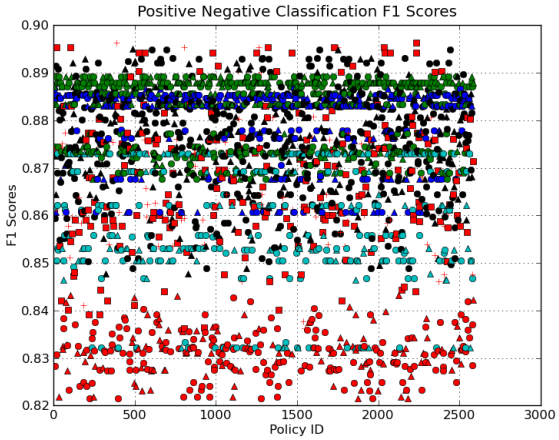

Fig. 9. Top F1 Scores-Polarity Classifier

Classifier	Avg.F1	Avg.Acc.	Max.Acc.
BLR(Gauss)	0.879	83.52	84.496
BLR(Laplace)	0.857	80.098	82.94
Naive Bayes	0.882	83.91	84.80
SVM(Linear)	0.880	83.29	85.736
SVM(Poly)	0.8571	80.80	82.946
SVM(RBF)	0.757	68.22	74.573
Voted Perceptron	0.875	82.96	85.43

SVM with Polynomial Kernel gives a reasonably good performance for Polarity classifier but not as good as the one with linear kernel. We also observed that Naive Bayes gives only marginally better performance than SVM. Voted Perceptron and Bayesian Logistic Regression also seem to perform comparably. Also, the best performance is achieved by considering either the *shortest shadows* of the words or all *shadows*. Also, SVM and Voted Perceptron give better performance with both InformationGain and χ^2 based feature selection as opposed to Naive Bayes that seem to favor χ^2 over Information Gain. Boosting based ensembling methods seem to give no significant performance gain with any classifier in this case.

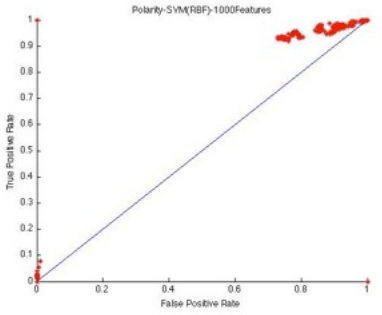

Fig. 10. Classifier Performance - SVM-RBF & LogisticRegression-LAPLACIAN

VII. Future Work

Our future efforts will be focused on finding a model for Neutral-Polar Classifier that gives a better F1 score. As of this writing, models with high *Recall* also give a high *False Positive Rate* whereas the ones with a low *False Positive Rate* have a lower *Recall*. Boosting based ensembling methods helped in uplifting the F1 score but we believe that we can get even better performance by applying smarter heuristics and other techniques. For instance, we believe that applying Natural Language Processing techniques *might* help in identifying the context and nature of the phrases thus assisting in finding similarities. Although our initial guess is that it might still not give a *very* high performance gain because microblogs are very short and not very well structured making them further difficult to be analyzed with language parsing techniques.

We also believe that considering 2-Grams or 3-Grams might give us slightly better performance. The intuition is that the word limit on the microblogs increases the probability of the messages that are closer in intent to be closer in syntax.

We briefly tried SMOTE to synthetically generate the samples for minority-class oversampling but without much success. We believe that we can give SMOTE another try after understanding its nuances. Coming up with our own method of synthetically generating minority class data can also prove to be useful.

Other areas that we can look for improvements are the *PreProcessing* of messages and *Feature Selection* that can help better discriminate the messages.

We also saw that SVMs gave a very good precision but were plagued with lower recall. We can consider introducing bias in SVMs towards the minority class by using different error cost functions for the two classes thus penalizing heavily for any minority class misclassification. Further, using SMOTE with this technique can give better performance as suggested in [8].

Another possible solution is to use SVM with a Kernel that exploits the semantic relations in between the words.

VIII. Conclusion

In this work, we analyzed the problem of *Sentiment Classification of Micro-blogs*, that is significantly different from

the other usual sentiment classification problem on structured and detailed messages. We used a two-classifier approach where the first classifier was a neutral-polar classifier and the second was a positive-negative(polarity) classifier. We tried different models, different parameters and different feature selection algorithms to address the problem. The biggest unsolved challenge for the neutral polar classifier is the imbalance in the dataset that gives a high accuracy but low recall. In an attempt to counter this effect we tried different re-balancing, sampling and boosting algorithms, the most successful of which is the combination of the boosting, oversampling and under-sampling taken together. We have been able to find a range of models from the ones giving a low false-positive rate but a lower recall to the ones with higher recall but higher false-positive rate. Some applications may benefit from the first extreme of the models and some may from the other but the best model is yet to be discovered that gives an optimal F1 score. In the case of Polarity Classifier may combinations seem to perform well with Naive Bayes and SVM being marginally better than the others. This cascade of a balanced neutral polar classifier, coupled with an efficient positive negative classifier marks our approach to this problem and leaves a wide scope for further research.

References

[1] Weka : Data Mining Software in Java. The University of Waikato *http://www.cs.waikato.ac.nz/ml/weka/*

[2] Bo Pang and Lillian Lee. *Opinion Mining and Sentiment Analysis*

[3] S. R. Das and M. Y. Chen *Yahoo! for Amazon: Sentiment extraction from small talk on the Web*, Management Science, vol. 53, pp. 13751388, 2007

[4] G. Forman *An extensive empirical study of feature selection metrics for text classification*, Journal of Machine Learning Research, vol.3, pp. 12891305, 2003.

[5] Bo Pang, Lillian Lee and Shivkumar Vaithyanathan *Thumbs up? Sentiment classication using machine learning techniques*, Proceedings of EMNLP,2002

[6] Mahesh V. Joshi, Ramesh C. Agarwal and Vipin Kumar *Predicting Rare Classes: Can Boosting Make Any Weak Learner Strong?*

[7] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse and Amri Napolitano *Building Useful Models from Imbalanced Data with Sampling and Boosting*

[8] Rehan Akbani, Stephen Kwek, Nathalie Japkowicz *Applying Support Vector Machines to Imbalanced Datasets*

[9] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer *SMOTE: Synthetic Minority Over-sampling Technique*