Final Report: The Outliers

Varun Gupta, Hari Parameswaran, Kevin Procopio

December 13, 2016

1 Initial Data Analysis:

First, we analyzed the tweets for additional features in addition to the word frequencies that could be used to make the classification better. A variety of features were considered including but not limited to tweet length, number of '#'s in the the tweet, number of '@'s in the the tweets, number of non-alphabetical characters in the tweets, and length comparison between happy and sad tweets. We found that the happy tweets we in average longer and has more '#'s in general. Using a plot of word counts for each feature we were also able to determine that approximately 4155 words in the vocabulary did not occur in the training set.

2 Simple Methods

2.1 Multinomial Naive Bayes

One of the first methods we used to classify the tweets was the built-in MATLAB Naive Bayes function. The multinomial version of Naive Bayes was used the multinomial version is well-known to work for data that can be turned into counts (like the word counts we used in this data set). Using a 10-fold cross-validation of the data we received an accuracy of 81.2%, and a test on the training set, we received an accuracy of 81.04 %. This was quite a good accuracy (it beat both of the baselines), and so we elected to use it in our aggregate models as well.

2.2 K-Nearest Neighbors

We also attempted to use the built-in MATLAB k-nearest neighbors model. Because k-nearest neighbors works best in a low number of dimensions (to avoid the curse of dimensionality), we combined this with PCA (i.e. the training data was PCA'd, and the same transformation that the training data produced was used to transform the test data). Using 10-fold cross-validation, we adjusted the two parameters (number of nearest neighbors and number of PCA dimensions)

in an attempt to optimize the accuracy of the model. Based on a grid search with 10-fold cross validation on a random partitioning of the data, we found that the optimal parameters for this method were 5 nearest neighbors and approximately 10 dimensions. This resulted in a (hold-out set, where we held out 500 data points) accuracy of approximately 72.00%. While this was reasonable, our method did not take into consideration the distances of the neighbors when voting on points in the test set. Therefore, choice of an appropriate kernel may have boosted our accuracy further (though we did not have time to implement this).

2.3 Support Vector Machine

We also trained a support vector machine on the data using the liblinear library provided in one of the CIS 520 homework assignments. Using 10-fold cross-validation, we received an average accuracy of approximately 78.40%. This worked reasonably well, but we decided that a further improvement could be made if the points close to the hyperplane margin were reclassified according to a different method (as the model was least confident about those points). This method is described below under "model composition".

2.4 Clustering

We also attempted a cluster based classification model. We tried to cluster the data using a gaussian mixture model, assign happy and sad to each cluster, and then assign test data to clusters (and hence retrieve a classification. The cluster method was trained in two different ways.
The first method involved using the entropies of the words as show in Figure 1. The words with entropy below 0.5 and above 0 were chosen for the GMM model because they were the words that tended to appear only in either happy or sad clusters. The datasets that had 0 entropy were not chosen because they had a one-to-one correspondence with sad and happy and hence provided no information. This criteria reduced the number of words to just 238.

The other method of choosing features for clustering was PCA. In this case, raher than words being used we clustered the reduced representation of each data point based on the principal components. 750 principal components were used as that explained more than 90% of the data.
Various models were constructed with different number of clusters. However it was observed that only 1 cluster for each sentiment produced the least in-training error. The most interesting observation was that more than 90% of the data that was clustered into the happy cluster was actually labeled happy. We concluded that there was a certain subspace of the feature space that could not represent sad sentiments and hence we by predicted happy for every dataset that belonged to this cluster with a comparatively higher probability than if it belonged to the sad cluster. The sad cluster did not show similar traits as the

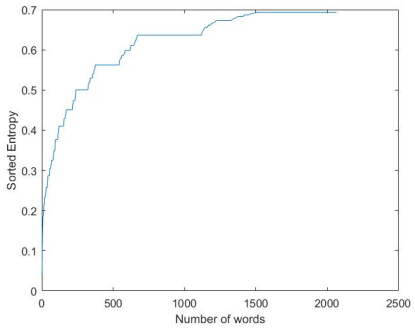

Figure 1: Sorted Entropy vs number of words

happy cluster with almost equal proportions of sad and happy datasets. Therefore, we built a multinomial Naive Bayes model using the entire training set to predict the sentiments of the dataset that belonged only to the sad cluster. The cross validation accuracies came to be around 82% but the accuracy on the test set came to be only 78%. This could be because of the generalization assumed when we chose only 1 cluster and also probably because of the selectivity of the words for the clustering. Clustering was also tried using word presence instead of word count but it did not improve the result.
In our entropy based feature selection, the criteria could be improved further. All features that occur more than once and had entropy of 0 could also be taken into consideration because they comprise of words that occur more than once in the entire training set and also imply happiness or sadness with certainty.

2.5 Bigrams

Since 4155 of the top 10000 words did not occur in the training set, and since bigrams generally give better performance than unigrams, we built a new vocab (containing roughly 15800 entries) and frequency matrix that included bigrams. This was produced by interating through each tweet and extracting the relevant tokens.
Unfortunately, using our composition model (see below) without AFINN, the bigrams gave an accuracy of only 45-50% during 10-fold cross validation and an accuracy of just 66% on the test set. We believe that this method was not accurate because there were 4155 words in the "top words" list that did not appear in the training set (and hence only appeared in the test set). These were not included in the bigrams, probably leading to errors.

2.6 AFINN

AFINN is a list of English words rated for valence with an integer between minus five (negative) and plus five (positive). The words have been manually

labeled by Finn Årup Nielsen in 2009-2011 (see [1]). We used these labels to compute a total happy/sad value for each tweet based on the AFINN values for the words present. One major draw back this method was that only about 900 words in the vocabulary were present in the list, and more that 2450 tweets didn't have any words categorizable by AFINN. Therefore, we elected to use this methods to break ties for uncertain values after tweets had already been classified by simpler method, such as SVM and multinomial Naive Bayes. We found that tweets with a high AFINN score ¿5 or ¡-5 had high correlation with the Y values. Thus when the individual methods gives a different predictions, we used the AFINN scores for the tweets if present as a final decider. Using AFINN as the final decider, we achieved our highest test accuracy of 81.80%.

Ref. Figure on page 7: We can see from the above graphs that 94.56% of the happy tweets have a non-negative AFINN score. We also notice that 68.15% of the sad tweets have a non-negative AFINN value. This can be attributed to the fact that there can be negations in the tweets. For instance "I am not happy" will attribute a positive and a negative AFINN score. Hence our final model included a —5— cutoff when making a decision, thus taking only those tweets which gives us maximum confidence.

3 Aggregate Methods

After testing the simpler models, we attempted to combine the simple models in various ways to produce ensemble methods with greater predictive power.

3.1 Regression

One naive attempt to aggregate the models was to simply train each individual model on a training set, use the models to make predictions on a held out set of data, and then perform a regression using the simple models' predictions against the actual labels for the held-out data to generate weightings for each of the simple models. Additionally, the method could be regularized with LASSO to zero out any simple models that are highly correlated with other models in our set. This method was performed using a k-nn, neural net, random decision forest, multinomial Naive Bayes, and support vector machine model. The support vector machine and multinomial Naive Bayes seemed to generally receive larger weights than the other models. The neural net (trained using MATLAB's neural net library) and k-nn consistently received the smallest weights. Using the weighted model, we received an accuracy of approximately 80.30 % (for regular regression) and approximately 79.8 % (for regression regularized with the built-in MATLAB LASSO function) on the test set. While these results were not bad, they were similar or worse than the individual multinomial Naive Bayes model. This was most likely because the assumptions of linear regression were violated by the models (for instance, linear regression assumes that the indicators are not correlated, and its possible that different models make mistakes in the same direction). LASSO did not seem to improve the model significantly (it seemed to

decrease the training set accuracy, but this was only slightly and could perhaps be due to random variations in the training sets, so it's more likely that LASSO simply provided no benefit to the aggregation weights). The high weightings on MNB and SVM were considered when designing later aggregate models, though.

3.2 Randomized Weighted Majority Algorithm

Another approach to aggregate the simpler models was the randomized weighted majority algorithm. This takes n "experts" and has them vote on each incoming example. If a weighted majority of the experts vote that the tweet is "happy", the aggregate model predicts "happy" with probability equal to the sum of the weights of individual models voting happy divided by the total weight of all models. If the aggregate model is incorrect, all of the individual models that gave incorrect assessments have their weights penalized. This algorithm was performed using a k-nn, neural net, random decision forest, multinomial Naive Bayes, and support vector machine model. Each model was individually trained on a random 80% of the data. The remaining 20% was then used to perform the weighting. This was done 10 times and the results averaged. Then, the weightings we used in conjunction with the individual models on the test set. Using a penalty wherein the weights were divided by 1.005 each time a prediction was incorrect, the total accuracy of the aggregate model was approximately 80.80 %. While this was a good accuracy (it beat both baselines), it was lower than the MNB by itself (which had accuracy around 81 %). We suspected that this was because there were data points that confused multiple models in the same direction. Therefore, we took a closer look at the data points that confused each model (which ultimately led to our model composition method, below).

3.3 Model Composition

Our most successful model was constructed which by combining the SVM, multinomial Naive Bayes, and AFINN word list. The idea behind the new model is that SVM is very accurate for features that lie sufficiently far from the hyperplane that separates the space into happy and sad subspaces. Any tweet that lay within a normalized margin of 1 from the hyperplane could not be guaranteed to have been predicted right (to test this empirically, we tested the accuracy of SVM on points only sufficiently far outside of the hyperplane and received an accuracy of approximately 85%). Therefore, the prediction from the Naive Bayes model was used to reclassify those tweets. For this classification, only the approximately 5845 words that occur in the training set were used to make the predictions. The accuracies observed from this cross validation were averaged over all the cross validation sets. For the individual SVM and MNB on all the data, we received accuracies of approximately 78.40% and 81.04 %, respectively. However, by using the procedure described above, we received a combined accuracy of 81.38 %. By using the AFINN word list in addition to this model we were able to achieve our highest accuracy of 81.80 %.

4 Citations

[1] Finn Årup Nielsen. "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs", Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages 718 in CEUR Workshop Proceedings : 93-98. 2011 May.

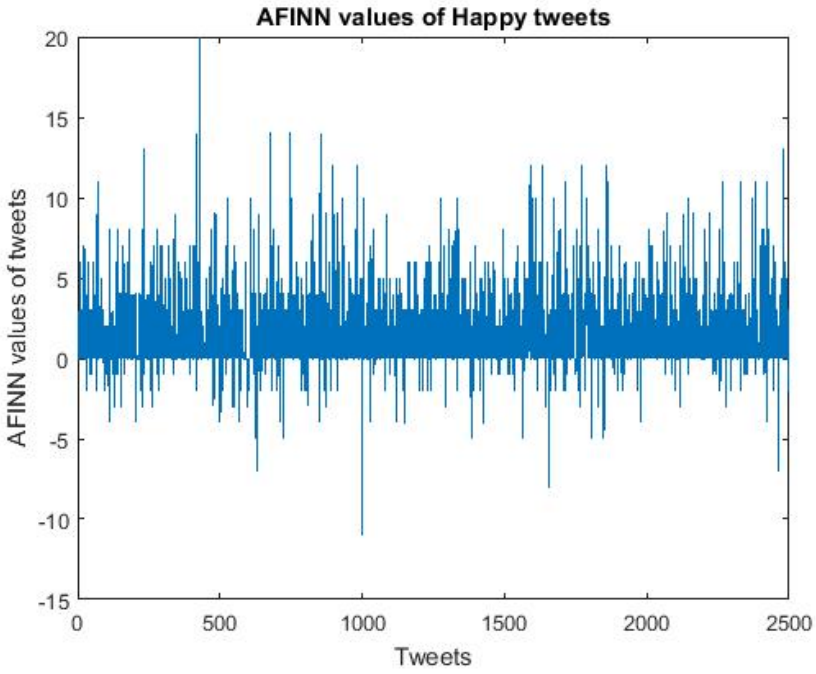

Figure 2: AFINN values for happy tweets in train data

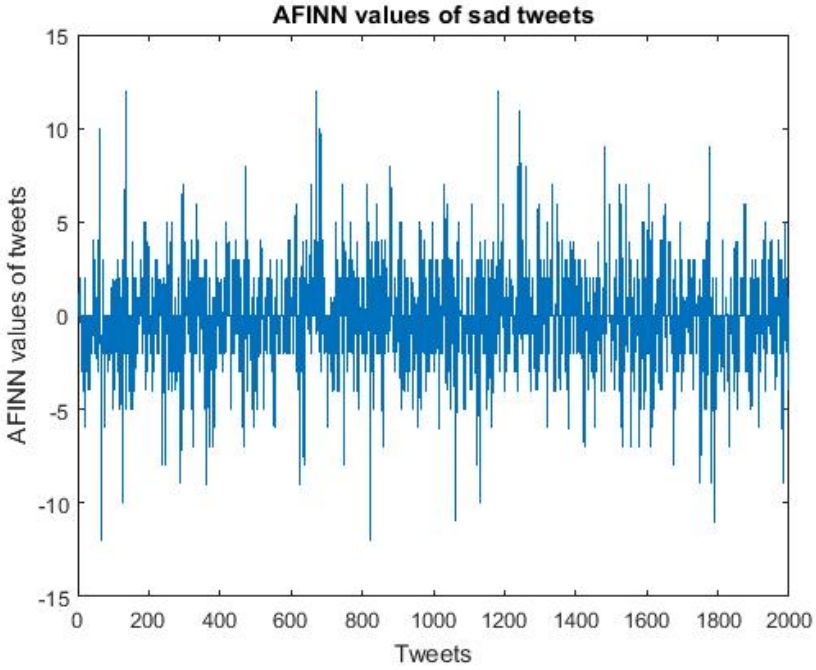

Figure 3: AFINN values for sad tweets in train data