Twitter Sentiment Analysis for Airlines

Avinash Kurudi , Arvind Mandyam Annaswamy, Srinivasa Suri

April 2018

1 Introduction

The usage of Micro Blogging websites has increased over the past several years tremendously. With easy access to such blogs using cell phones, a lot of awareness among people and the ability to connect with people instantly over such blogs have led to an increase in the usage of Micro Blogging websites. With almost 250 million tweets per day, twitter is one of the most exciting social media platform to perform sentiment analysis[7]. There are several reasons why sentiment analysis is difficult in Twitter. To quote few examples: Usage of twitter varies with each country. For example, People in India use as a news portal as compared to people in USA who use it primarily to connect with others. Also, the tweets pattern vary greatly between people of different ages. A middle aged person might not use lots of LOL's as compared to a teenager. The existence of multilingual tweets and puns in the tweets make it really difficult to get the actual sentiment of the tweet.

In this project, we focused on one such site called Twitter. There's great amount of sentiment that exists in the tweets and we have picked up airlines sentiment as our primary topic. Whenever people are happy/sad about their experience with an airline, they generally tend to tweet about it in Twitter and this motivated us to choose airlines as our topic as we expect it to have good amount of sentiment information in the tweets. Our goal of the project is to gain insights into what people are tweeting about different airlines,design a model which could classify the sentiment into three major classes - positive/negative or neutral, what feature set works best and which machine learning technique is better for airlines tweets. To answer these questions, we have built a pipeline for sentiment analysis by downloading the tweets, pre-processing the tweets, extracting it's sentiment from the tweets and building machine learning predictors for the classifiers. We have compared the results between n-grams of order 1,2 and 3 to identify the best recommended approach for twitter analysis. Moreover, we have also used pre-trained Glove word vectors of 300 dimensions as our feature set to analyze if we are able to achieve any significant improvement in performance.

2 Data Preprocessing and Visualization

2.1 Dataset Description

Twitter provides API's to pull tweets from it's website. Although, there are certain limitations on the number of tweets we can pull for each day and the character limit of 140 for the tweets pulled. This still contains lots of information about the sentiment. We have used the airlines twitter handle to pull the tweets for the respective airlines. The list of airlines and the tweet counts of the data we pulled are described in the Table [1].

Sno	Airlines	Tweet Count
1	Allegiant Airlines	19744
2	Alaska Air	6600
3	Delta Airlines	100080
4	American Airlines	24983
5	Fly Frontier	1692
6	Hawaii Air	900
7	JetBlue Air	18009
8	SouthWest Airlines	68000
9	Spirit Airlines	3300
10	SunCountry Airlines	2200
11	Virgin America Airlines	1500

Table 1: Airlines and Tweets Count

2.2 Cleaning and Wrangling

Tweets represent real world data and there exists a significant amount of pre-processing before we can get the data into a right format to train the classifiers. For example , People use emoticons and words like 'lol' are not there in the dictionary. This makes cleaning twitter tweets a complex process and hence cleaning tweets becomes an important task for sentiment analysis. For this project, we cleaned the tweets in the following way: Since Twitter limits the tweet characters only upto 140 characters, we have removed the extra ... that comes up at the end of the tweets. We then convert all the tweets to lower case characters.We removed few stopwords from the tweets and certain punctuations like ',' , '.' , '-'. Most tweets generally have lot of user mentions. Since these user mentions do not affect our task of determining sentiment, we converted all user mentions to "AT_USER".

Although retaining the case sensitivity of the words is useful to get the sentiment of the tweet,for example, "WHERE ARE MY FLIGHT TICKETS" in a a tweet conveys more negative emotion than "where are my flight tickets" in a tweet, we have removed the case sensitivity from the feature space in order to optimize on the feature space for the data. Otherwise, the feature space might blow up by a couple of times and it might make the feature set more sparse and also the training of machine learning classifiers slow.

Having tweets from different language makes the sentiment analysis more complex. As our focus is on airlines data from primarily from US based airlines, we have discarded the tweets that have words in other languages than English.

We have removed the tweets which come from the verified airline handles since these tweets do not provide the sentiment regarding the airline and we have discarded the tweets that are just retweets from to avoid duplication of the same tweet.

2.3 Building the Sentiment

One of the harder problems in sentiment analysis is getting the label. Everyday , we have new tweets and new trends that happen in the twitter world. The only golden truth can be a true label by a human which is very expensive. We have almost two hundred thousand tweets and it's not possible to manually label the tweets. One of the popular ways is to manually label few tweets, build a classifier on the bootstrap data and classify few more tweets with this classifier. Build a new classifier on this labeled data and keep on filling the labels in a cascading manner till we end up with labelling all the tweets. This method is called bootstrap labelling. However,we didn't follow this approach as we didn't have enough human resources to label tweets (to get right bootstrap labels, it's good to label atleast 1000 tweets etc). Our initial approach was to take bootstrap samples of 100 tweets picked randomly but, we realized that might have a lot of bias based on the 100 tweets that we get and we may not get a true representation of the entire dataset.

We have relied on two NLP methods to get the labels for the tweets. The methods are defined as follows:

- Using Sentiment Lexicon - One of the ways to manually label the tweets is to use the sentiment lexicon. We used the MQPA sentiment lexicon [8] to manually label the tweets. The lexicon consists words and whether they convey a positive or negative effect. We calculated a score for each word in the tweet based on this lexicon. If the final score obtained was positive, we labeled it as a positive sentiment, if the final score was negative we labeled it as a negative sentiment and if the final score was close to 0 we labeled it as a neutral sentiment.

- Using textblob polarity - One of the other ways to label the tweets was using the textblob library of python [6]. Given an input text, the library produces a polarity score for the text. Using this value, we get a label for each tweet in our dataset.

Following this approach serves us two purposes. It gives us insights into which method is working better. This also eliminates the necessity to label the data manually which takes a lot of time as discussed earlier.

One of the drawbacks of using these approaches is that there is no guarantee that these labels are the 'gold standard'. However, since we cannot hand label thousands of tweets, for the purpose of this project we have used this approach.

2.4 Tweets Visualization

Our motivation for this research work was to equip company managers with visualizations and help them interpret public sentiments for effective decision making for their brands.

Our research scope is limited to airlines operating in the United States and thus we have identified three major areas within the scope of this research where targeted visualizations may be effective for brand managers. The center theme of all the visualizations was decided to be sentiment analysis.

Geographic Location: It is useful for brand managers to know how sentiments of people vary across different cities. Our dataset as seen in the reports section provides info about public sentiments varying across all major US cities. For the sake of data visualization, we have picked top 5 cities where most number tweeters for Hawaiian Airlines are located and plotted the sentiment analysis for the same. Location was obtained using the users location attribute from the tweet.

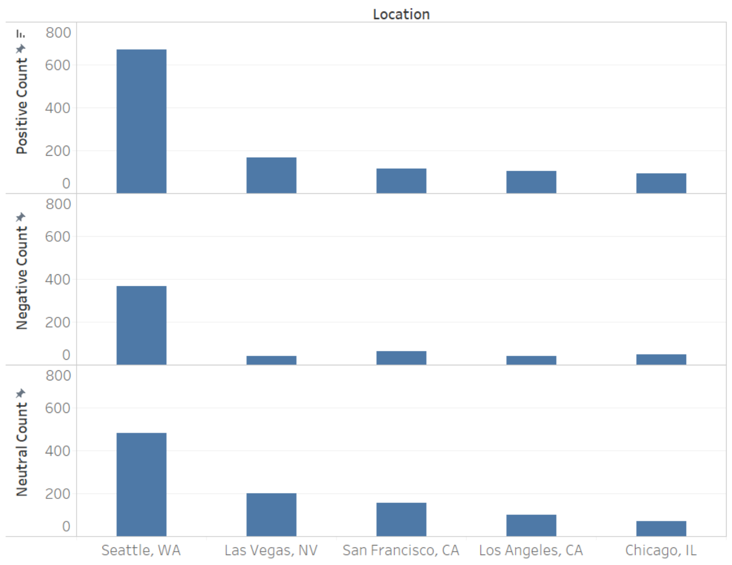

Figure 1: Positive, Negative and Neutral tweet counts based on location for Hawaiian Airlines

Time: Analysis of change in sentiment over time is useful in many business settings. Visualization involving time can enable users to identify sudden change in sentiment trends which can lead to pinpointing events that may have led to change in trend. Our research can yield both long term and short-term patterns in sentiments with respect to change in time. In the below graph we show how sentiments changed for one of the airlines over a span of 10 days.

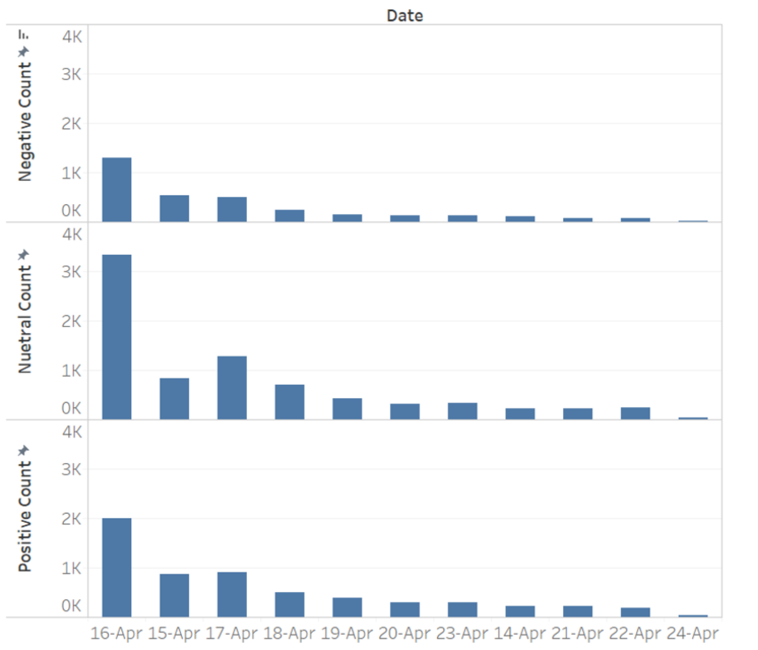

Figure 2: Positive, Negative and Neutral tweet counts based on different dates for Alaskan Airlines

Word Cloud: Our analysis can also identify the most common words used in the tweets related to airline companies and use sentiment analysis to identify the most positive and negative trending words for the company. Below is one such example depicting most used words to express negative sentiments of American Airlines.

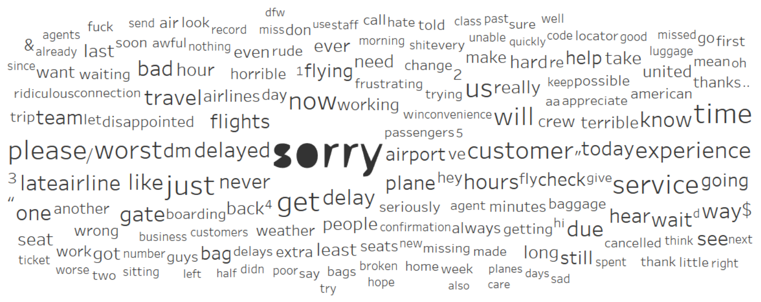

Figure 3: Word cloud representing negative sentiment for American Airlines

3 Implementation

3.1 Feature Engineering

A significant time of any Machine Learning Engineer is spent on feature engineering. Good feature engineering metrics can help the classifiers to improve the accuracy and reducing training time.

In our project, we first build the entire bag of words model with ngrams of words varying from 1 gram to 3 grams as features and then reduce it with PCA to discard the features with low variance.

Another way of feature engineering we used is to incorporate the Glove vector corpus to build the feature set. GloVe [1] is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus to produce 300 dimensional word vectors.

For each word in the tweet, we get it's 300 feature vector using the pre-trained glove word vectors and average out all the vectors obtained for each of the words in the tweet. So, each tweet is represented as a 300 dimensional vector which can be passed into any machine learning algorithm. Using this approach gives us the advantages of having better representation of the tweets for the machine learning algorithm. These pre-trained vectors encode much more information about the word using large word-word co-occurence matrices.

3.2 Description of Machine Learning Techniques

After pre-processing the tweets and labelling the sentiment, our next goal is to build a classifier to predict the sentiment of the tweets. Building a predictor for sentiment serves two purposes. One, having a good classifier is one way with which we can predict the sentiment for new tweets from twitter. Another goal for building a classifier is to enhance the sentiment lexicon to be more suitable for twitter dataset or more specifically for the airlines dataset. The sentiment corpus is very generic and is meant for the English language in general. This classifier will have more insights about the relevant dataset as it is trained on the dataset and thereby, we can remove the dependence from the nlp corpus set for estimating the sentiment of the new tweets. Some of the machine learning techniques we tried are:

- Logistic Regression: Logistic Regression is one of the well known classifiers and we have used GridSearchCV with C values of C=0.01,0.1,10 to fine tune the parameters. Because of limited computation power , we have used max iterations of 100 instead of running the Gradient Descent till convergence.

- Naive Bayes: Naive Bayes is a widely used classifier for spam classification or for bag of words model. We have used Naive Bayes with no assumption of prior weights. Hence ,the parameters learnt using M.L.E are just the counts of the categorical values.

- Linear SVM: SVM's are generally known to be slow while training on the dataset as training with kernels are of order O (n^2 * m) where n is the the number of training examples and m is the number of features in the feature space. Hence, we have used LibLinear SVM which trains faster on larger datasets as it is optimised for linear SVM. Moreover, LibLinear is known to be very efficient for large scale sparse datasets[5].

- Decision Trees: We have used Decision Tree with Gini index impurity with a max depth of 3 or 5 or 7 using GridSearchCV. As our feature space is around 300 at maximum, we have restricted the maximum tree depth to 7 to avoid overfitting of the training data.

- AdaBoost: AdaBoost is one of the most promising out off the shelf classifiers. With each new classifier in the AdaBoost, it tries to learn the instances that it has missclassified in the last classifier. Although in theory, AdaBoost is expected to overfit, it has proved to be very valuable in the practical world.This motivated us to use adaboost with decision trees and 100 estimators.

- Random Forests: We have used Random Forests with 100 estimators for the bag of words model. Random forests are known to be very fast because it selects a subset of features from the feature set and trains very fast.

3.3 Model Evaluation

We have trained the data on Logistic Regression, Naive Bayes, SVM, Decision Trees, AdaBoost and Random Forests.We split the data in 75:25% for the training and the test set. We then train the classifiers on the training set with GridSearchCV and with a cross validation of 5. The parameters that we tuned for the dataset for different classifiers are defined in the table[2]. We report the score in terms of balanced accuracy as this is a multiclass problem with three categorical varaibles of negative, neutral and positive sentiments.

Sno	Classifier	Parameters
1	Logistic Regression	C=[0.01,0.1,10]
2	Naive Bayes	Default Parameters
3	SVM	C=[0.01,0.1,10]
4	Decision Trees	max_depth=[3,5,7,]
5	AdaBoost	n_estimators = 100
6	Random Forests	n_estimators = 100

Table 2: Classifiers and parameters

4 Analysis

We trained the classifier mentioned in Section 3.2 over MQPA Sentiment labellings, Textblob Sentiment labelling and over different feature sets like Glove word vectors, unigram/bigram/trigram feature set. Because of our limited computation power, training on the full blown feature set was consuming too much of time and thus we have projected the 5000 feature set data to lower dimensions of 30,100 and 300 features using PCA for unigram/bigram/trigram feature set with textblob polarity and MQPA corpus polarity. As we used PCA, we were able to retain the important features in the feature set and making the classifiers train faster.

Table 3,4,5 show the results of the classifiers on polarity computed through MQPA Corpus set. Clearly, it can be seen that unigram model outperforms all the other ngram models like bigram or trigram in all these 3 tables mentioned. This makes sense as unigram model is more dense as compared to bigram/trigram models which are very sparse. We have performed PCA on the 5000 word feature set and we observed that as we incresaed the number of feature in the PCA, the accuracy of the best classifier has improved. The best Clasifier AdaBoost has an accuracy of **61.7%** with 30 features. It increased to **65.6%** with 100 features and to **68.5%** with 300 features reduced by using PCA. Clearly, having more features has increased the accuracy of the model.

In comparision with polarity determined by MQPA Corpus set, Tables 6,7 and 8 show the accuracy of the models by training them on the textblob polarity labelling. An exhaustive comparision is made between all the classifiers, uni/bi/tri gram models after applying PCA with 30,100 and feature set. Our results show that using textblob polarity has results in better classifier prediction and accuracy. The best performing classifier SVM has an accuracy of **76.0%** test accuracy with Logistic Regression and AdABoost trailing by only a slight margin in accuracy. Moreover, It can be seen that the models performed significantly better with unigram models than bigram/trigrams models in this case too. Clearly, as we increased the number of n-grams, the accuracy started desreasing and in the case of trigrams, the accuracies of some of the classifiers is almost 0.5 which is just as good as random guessing.

Boosting is one of the significant classifier that outperformed other classifiers most of the times and thus turned out to be a better predictor for our sentiment analysis. From tables, 3,4,5,6,7,8 it can be seen that boosting has the highest test accuracy or the second highest test accuracy. Naive bayes tends to perform poorly on the PCA feature set for all the different feature set that we have tried out here. Logistic Regression is the next best classifier after Adaboost as we can see from table 3,4,5,6,7,8. One of the results that we can see is that random forests doesn't perform very well as comapred to logistic regression or adaboost, one of the main reason why this might be happening is because random forests select few of the features amongst all the feeatures for each of its random forest, as our feature set is already a reduced feature set based on PCA, selecting few of the features from this set results in poor selection of features and hence it it can be observed that random forests's accuracy is not better than adaboost or logistic regression or adaboost.

As for our limited computation power, we have performed one round of building Logistic Regression, Naive Bayes, SVM and Random forests over the entire 5000 feature set along with emojis in the feature set and we observed that the classifiers prediction rate has increased significantly. Table [9] shows the accuracies of different machine learning classifiers using the full blown feature set including emojis also as a feature. The Accuracy of logistic regression has increased to **90.3%** with the full blown feature set and the accuracy of svm has increased to **90.6%** from **76.0%**. However, training the classifiers with a large feature set took a lot of time, almost 4-5 hours to train the logistic regression with gridsearchCV. Table [10] shows the accuracies of the different models using the averge glove word vectors as the feature set. The best model is adaboost with accuracy of **73.5%**. This average score may be due to the fact that the many of the words in our corpus may not be available in the glove corpus and hence this might not give the best representation of the word. Hence, we recommend falling back to the classifiers trained on 300 PCA feature set if the missclassification cost isn't too high. Missclassification cost generally comes from the business perspective and the cost of mispredicting depends on how serious we consider our misprediction is. For example, missclassifying a positive tweet as negative is generally okay.But, missclassifying a negative sentiment tweet as positive is not okay as any company generally wants to provide a good and superior customer service.

ngram	logreg	nb	svm	dt	adaboost	rf
unigram	.563	.505	.560	.544	.617	.567
bigrams	.482	.460	.483	.538	.557	.539
trigrams	.448	.441	.449	.474	.478	.472

Table 3: 30 features with MPQA Corpus on Test Data

ngram	logreg	nb	svm	dt	adaboost	rf
1gram	0.639	.537	.634	.558	.656	.588
2gram	0.526	.503	.523	.529	.555	.531
3gram	0.468	.459	.468	.478	.481	.475

Table 4: 100 features with MPQA Corpus on Test Data

ngram	logreg	nb	svm	dt	adaboost	rf
1gram	0.694	.549	.691	.546	.685	.593
2gram	0.557	.522	.556	.539	.567	.537
3gram	0.474	.463	.474	.470	.477	.470

Table 5: 300 features with MPQA Corpus on Test Data

ngram	logreg	nb	svm	dt	adaboost	rf
1gram	.605	.519	.598	.593	.656	.602
2grams	.535	.505	.531	.564	.595	.560
3grams	.498	.491	.498	.522	.528	.521

Table 6: 30 features with TextBlob Sentiment on Test Data

ngram	logreg	nb	svm	dt	adaboost	rf
1gram	0.681	.560	.678	.617	.710	.653
2gram	0.575	.535	.571	.565	.604	.574
3gram	0.513	.496	.512	.511	.528	.519

Table 7: 100 features with TextBlob Sentiment on Test Data

ngram	logreg	nb	svm	dt	adaboost	rf
1gram	0.759	.587	.760	.631	.759	.673
2gram	0.603	.551	.602	.565	.609	.582
3gram	0.522	.508	.523	.511	.526	.520

Table 8: 300 features with TextBlob Sentiment on Test Data

ngram	logreg	nb	svm	dt	adaboost	rf
1gram	**0.903**	**.839**	.906	-	-	.505
2gram	0.624	0.623	0.621	-	-	0.492
3gram	0.53	0.53	0.527	-	-	0.486

Table 9: Scores with emojis included as feature on the test data on 5000 feature set

ngram	logreg	nb	svm	dt	adaboost	rf
1gram	0.608	.604	.725	.598	.735	.616

Table 10: Scores with wordvec of 300 features

4.1 Error Analysis

Since we cannot theoretically evaluate our classifier across thousands of tweets, we have analyzed a few random tweets from our data set.

For example our model predicts the following tweet as negative sentiment which is correct *Utterly disgusted w @AmericanAir - my sis booked a trip w her 2mo old son not knowing he didn't need a ticket amp; not... https://t.co/LRVc9dx8Qk*

Another example which shows that our model predicted the below tweet as positive whereas the real label would have been negative
@AmericanAir They weren't bothered. Half the passengers on the plane complained about the same thing. The first hal... https://t.co/7hjZxjO8Vu

This maybe due to a variety of reasons. Our training feature set n-grams may not have seen the ones occuring in the tweet or the true label which had been assigned to it may itself be wrong.

5 Recommendations and future work

In this project, we have presented sentiment analysis with respect to two different sentiment extractor techniques and compared their performance with 1gram,2gram and 3 grams feature vectors. Extending this, one interesting study would be to include the 1-gram,2-gram and 3 gram together as a feature vector and build a model on all the features together instead of training models on each ngram dataset separately. This will give the classifier model the comprehensive power of using all the different ngrams to train it and might potentially lead to better results if engineered properly. Because of limitations of computational power and limits of google cloud credits, one thing we couldn't explore much was deep learning. Research [3] has shown that using CNN and LSTM can give much better predictive power in sentiment analysis.This is one area which can be explored for the sentiments as we can gather lots of data from Twitter, if we have a paid subscription from twitter.

One of the challenges we faced during our sentiment analysis is how to go about benchmarking our classifier results with gold standard techniques from the twitter sentiment analysis world. For example , [2] results show that having tree kernels and POS tags in the sentiment analysis outperforms unigram models. Although we were very excited to try to these techniques in our project to compare the results with our results, few of the challenges we faced were not having the dataset that they used and not having access to the code base that they used etc. Few of the existing literature [4] have experimented with ngrams of characters instead of words and it turns out to be a better predictor than n-gram model using words. One good area of study would be to see if using ngram characters improve the accuracy of the models for bigram/ trigram in our case etc

6 Conclusion

Sentiment analysis on micro-blogging sites such as twitter is a complex task involving labelling the data with the true labels, cleaning the data, identifying sentiment from data which may include linguistic structures like puns, sarcasm. This poses a slew of challenging tasks. In this paper, We have explored the pipeline of data analytics of gathering the data, cleaning the data, visualising the data,building the features for the data and generating predictive models for the data etc. We have several key observations from the analysis. They are as follows:

- Our analysis shows that sentiment prediction works best for 1gram words more than bigram/trigram words. The classifier performance trained on 1gram models outperform the models that are trained on bi/trigram models significantly.
- We observed that sentiment analysis performed using the labelling from textblob is more accurate and performs better than the sentiment labelling done by MQPA Corpus set. Thus, we recommend using textblob polarity to label the sentiment of the tweets over using MQPA Corpus to label the tweets.
- Sentiment analysis including emojis tend to outperform the sentiment analysis without emojis significantly and have proven to be powerful features for the machine learning algorithms since they inherently encode a lot of human sentiment. We thereby recommend using emojis in the sentiment analysis as they are powerful predictors of the sentiment.

References

[1] Glove word vectors. https://nlp.stanford.edu/projects/glove/.

[2] Sentiment analysis of twitter data. http://www.cs.columbia.edu/ julia/papers/Agarwaletal11.pdf.

[3] Sentiment-analysis-using-cnn. http://nlp.arizona.edu/SemEval-2017/pdf/SemEval094.pdf.

[4] Sentiment analysis using n-grams. https://arxiv.org/pdf/1505.02973.pdf.

[5] Svm liblinear. https://www.csie.ntu.edu.tw/ cjlin/liblinear/.

[6] Textblob polarity score. https://pypi.org/project/textblob/.

[7] Tweets count. www.geekosystem.com/twitter- 250- million- tweets- per- d.

[8] Janyce Wiebe Theresa Wilson and Paul Hoffmann. Mpqa subjectivity lexicon. http://mpqa.cs.pitt.edu/lexicons/subj$_lexicon/$, 2005.