# Efficient Sentiment Analysis of Feeds for Rapid User Information Gain

**R. Kent Wills**
University of Maryland
College Park, MD 20742
rkw14@umd.edu

## ABSTRACT

Recently my effort and attention has been focused on a start-up that aims to bridge gaps in the financial world. Specifically the gap of financial literacy. Throughout this exploration, one important, recurring theme is the need for a quick assessment of aggregate analysis. Accordingly, this document will serve as a road map to generating quick, efficient, and accurate analytics of RSS and Twitter feeds. This analysis will in turn resolve the data to an overall sentiment of POSITIVE, NEGATIVE, or NEUTRAL. This paper proposes the creation of a production grade classifier for a website. To elaborate, production grade should consider not only accuracy of the assessment, but also speed. Many users may not use the feature if it takes too long to load on the site, further degrading interaction. Furthermore, the dichotomy of structured and unstructured data, RSS and Twitter feeds, should provide insight as to how one may be used to support the other. This paper shows that the production grade classifier is an attainable goal, although may never get used effectively due to the amount of irrelevant data.

### Author Keywords

Machine Learning; Natural Language Processing; Sentiment Analysis;

## INTRODUCTION

Sentiment Analysis, determining the overall polarity of text, has been widely explored in Natural Language Processing [6, 2, 12] and Machine Learning Fields [9]. It has been popularly used for tasks ranging from brand sentiment [4] to stock prediction in financial blogs [7]. The intentions of this paper are largely practical, it wishes to explore a "good enough classifier" that is also quick. It also attempts to explore using parallel corpora, one with sentence structure and the other without. Furthermore, I wish to see if the use of both corpora will result in overall better performance.

## DATA PREPARATION

Corpora generation for this paper was no small task, and as such the next few sections will not only discuss the retrieval of the data, but also the cleaning of the data.

### *Retrieval*

Twitter rate limits access to their data [13], with 150 requests per hour for unauthenticated users to 350 requests an hour for authenticated users. Furthermore, Twitter does not allow the storage of data on local servers with tweet id and text in the same corpora. This has led to Twitter forcing Stanford to take down a twitter database [10][17] of over 476 million tweets in a seven month period. Luckily, Sander's Analytics has provided an offline database [1] of twitter ID's with attached sentiments. To obtain data with a minimal amount of code, Twitterizer [14] was used. Twitterizer's TwitterStatus has the following useful properties:

- CreatedDate - Date for tweet
- Text - Tweet Text
- Geo - Location of tweet
- RetweetCount - Number of times the tweet was mentioned
- Retweeted - Whether this is an original or retweet.

It is interesting to note that the created date could be used to plot sentiment over time, Geo-Location could plot sentiment over the US, and retweet count could be used to reinforce a sentiment for a specific company.

Building an RSS corpora resulted in a slightly different approach. Notably, the data is not strictly rate limited and can be retrieved in XML format. C# has a great XML Document class [5] to handle the response. The XML response contains the following RSS data:

- Title - Title for the RSS
- Description - Short summary for the article
- Date - Date of the RSS post
- Link - Link to the article

Currently the title and description are the only tags being used for analysis. Further development could include parsing the document from the html link to provide a less sparse feature vector. Drawbacks from adding extra data could include using RSS feed data more than the twitter content for prediction. This, however, could be handled effectively by weighting the vectors [3].

### *Analysis of Text*

In order to properly setup the data for analysis the data must be properly tokenized. The process of tokenizing data is described as breaking down sentences into words. With RSS data the tokenization is rather trivial, whereas tokenizing tweets can be difficult at times. Take the following tweet for example:

*One man's opinion: $AAPL in a Multi-Year Decline, Shares to Fall 50-70% http://t.co/hPSEm3jN $AAPL down 3% so far today #investing :(.*

There are many elements to a tweet that could have significance. It allows a tweet to get the maximum amount of information out in a small with the minimum amount of text. There is a potentially huge amount of information that can be garnered from such a small amount of text, at the cost of having to handle many special cases. Here are a list of common elements:

- Hashtag: The hashtag is preceded by # and it specifies the category that the tweet belongs in. This alone can provide valuable information to ensure that our tweet is in the domain that we want. The above example shows that we have retrieved a tweet that is meant to be centered in the investing category, which is the proper domain for out task.
- Dollar Sign: A newer type of hashtag is cropping up, the $ sign. This is used to specify stock symbols, such as $AAPL for the company Apple Inc.. This is equally helpful in determining if the tweet pertains to the specific company that we are interested in.
- @Replies and Mentions: This will simply tag a tweet as being part of someone else's conversation. We will be primarily unconcerned with this feature. In the future we could use this to identify whether or not the user was an originator of the comment or idea.
- Emoticon: While not a word, an emoticon or smiley can describe the entire sentiment of the tweet without saying a word. :)
- Embedded links: You may have noticed the ugly web address http://t.co/hPSEm3jN. This is a shortened hyperlink that pertains to the tweet in order to preserve the 140 char restriction. Currently the program does not make use of this feature. This could be a potentially lucrative endeavor because it may provide a link to a web page. This web page will give us more features to use in order to classify the overall sentiment for the tweet.

### *Steps to Tokenization*

The following steps have been used from Christopher Potts tokenizing techniques [8] to tokenize the twitter text.

- Basic Normalization - Isolate XML and HTML and map HTML to unicode
- Whitespace Tokenizer - Convert to lowercase and splits text on whitespace (ex. "AAPL is great" $\rightarrow$ "aapl","is","great")
- Smiley Tokenizer - Capture %96 of the emoticons used in twitter (ex. :) )
- Twitter Markup - Grab usernames and hashtags (ex. #investing)
- Shortening - Map repeating vowels of length greater to three to three or less (ex. sweeeeeet $\rightarrow$ sweeet)

## DATA STORAGE/DATA

### *Storage*

C# has great interoperability with SQL Databases using EntityFramework. All data is stored in a RSS, Twitter, and Sentiment Table for retrieval. Moreover, I can easily randomly reserve my test data in the beginning by using the SQL statement:

```
With TEST AS(
SELECT TOP 10 percent * from Twitter order by newid())
UPDATE TEST
SET Test=1
```

where, in this case, I chose to set aside %10 of my data for testing.

### *Data*

Despite the data archive claiming to have 5500 tweets, only $\sim 1,400$ were labeled positive and negative. To provide an even weighting of examples I maintained $\sim 700$ positive, negative, and neutral tweets. This led to a $\sim 2,100$ Tweet database. RSS feeds were equally as bad with more than 100 of my 300 RSS feeds being marked as irrelevant. Retrieving Data for classification was the most time consuming task for the project and will result in not being able to use this classifier in a commercial product.

## METHODS

My method is of the supervised nature, as I am trying to learn positive, negative, and neutral sentiments from labeled data. Unfortunately I was not able to utilize the Accord.Net framework due to data input issues. I instead, developed the following algorithms in C# from scratch.

### Naive Bayes

#### *Notes on Implementation*

Naive Bayes is largely known for being a simple solution that is widely used due to its performance in real world applications. In this case, while words are not independent, we can try to assume their independence and see how well the classifier performs. Much of the Naive Bayes information came from Stanford's CS124 class [11]. Naive Bayes performs better with large amounts of data. Due to a limited training set, data was bootstrapped ten times. All for-loops were parallelized as well to increase speedup in training. Finally classifiers were bagged and the mode result was taken. The following algorithm was used:

**Algorithm 1** : Naive Bayes Train

$N \leftarrow CountTwitter(Examples)$
**for all** $c \in C$ **do**
  $N_c = CountWordsInClass(T, c)$
  $Prior[c] \leftarrow N_c/N$
  $Text_c \leftarrow ConcatenateText(T, c)$
  $W[c] \leftarrow ExtractFeatures(Text_c)$
**end for**
**return** W

When classifying examples, multiplying probabilities can result in floating-point underflow, instead probabilities were summed:

$$c = \underset{c_j \in C}{argmax} \log \mathcal{P}(c_j) + \sum_{w \in Words} \log \mathcal{P}(x_w | c_j)$$

Laplace smoothing [16] was applied to the above equation to allow for nonzero probabilities of words that do not occur in the sample.

### Decision Trees

Decision trees, from a global perspective, work very similarly to Naive Bayes, the highest frequency words are given the most importance to separate the data. Decision trees, however, seem to be more rigid in this case, with decision being yes or no, with no in−between. The decision tree used was adapted from Hal Daume's Book, CIML. The difference here is that information gain is used instead of the mode.

**Algorithm 2** : DecisionTreeTrain(Data, Remaining Features, Height) from CIML

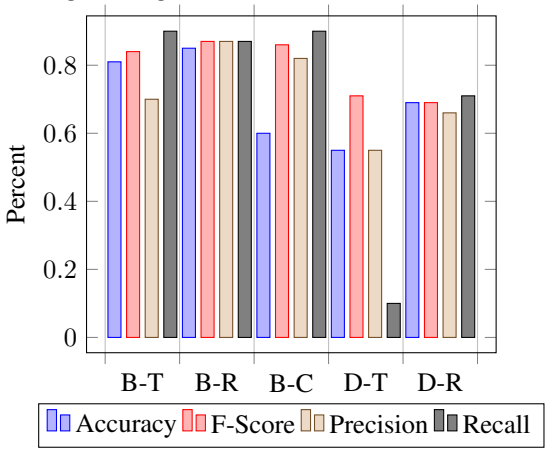
```
guess ← mode(data)
if labels in data are unambiguous then
    return Leaf(guess)
end if
if remaining features = then
    return Leaf(guess)
end if
for all f ∈ remainingfeatures do
    NO ← subset of data on which f = no
    YES ← subset of data on which f = yes
    score[f] ← InformationGain(f)
end for
f ← MAX(score[f])
NO ← subset of data on which f = no
YES ← subset of data on which f = yes
left ← DecisionTreeTrain(NO, remfeatures, height)
right ← DecisionTreeTrain(YES, remfeatures, height)
return NODE(f, left, right)
```

The classification function is straightforward and recursively iterates through the tree to find the solution.

**Algorithm 3** : DecisionTreeClassify(T, TestPoint) from CIML

```
if tree is of the form LEAF(guess) then
    return guess
else
    if f ∈ TestPoint then
        return DecisionTreeTest(left, test)
    else
        return DecisionTreeTest(right, test)
    end if
end if
```

**PERFORMANCE**




The prefix in the above chart x-axis label B abbreviates the word Bayes and D abbreviates Decision Tree. Similarly the suffix abbreviates Twitter, RSS, and Combination as T, R and C, respectively.

*Note on measurements.*
All performance results were computed based off of the F-Score, where

$$FScore = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

and

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

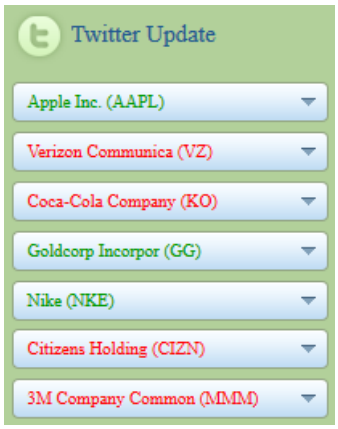
In this example, the label represents three classes, "positive, negative, and neutral" because we cannot run our traditional F-Score metric, Micro-Averaging [15] was implemented. Micro-Averaging collects decisions for all classes, computes the confusion matrix, and then is evaluated.

Training was completed on Twitter and RSS feeds separately, then trained together. The hope was that the RSS feeds would reinforce the overall sentiment for the companies because it reduces the sparsity when dealing with short twitter feeds. Unfortunately it appears that the addition of the structural text did not provide any gains to prediction. It is also important to note that the Decision tree performs much worse than Naive Bayes and was not tested for improvement with RSS and Twitter feeds combined.

Because F-Score metrics alone are tough to visualize, a word diagram was generated to give the user a sense of which words were pulled out and used in the majority of positive prediction tasks:


## CONCLUSION

While it does seem that RSS feeds do reinforce the Twitter feeds positively, limitations keep this from being a production grade classifier. One limitation is the cap on the number of Twitter feeds that can be downloaded an hour, a large scale website would easily reach this cap. Many of the feeds pulled from both twitter and RSS suffered from having too many irrelevant examples. These irrelevant examples would have to be filtered out of the process to get the best performance. While the production grade classifier might not be practical for my process, I have provided an example of the working "split-second" classifier used on the current website.




## REFERENCES


1. Analytics, S. http://www.sananalytics.com/lab/twitter-sentiment/, 2012.

2. Choi, Y., Kim, Y., and Myaeng, S.-H. Domain-specific sentiment analysis using contextual feature generation. In *Proceedings of the 1st International CIKM Workshop on Topic-sentiment Analysis for Mass Opinion*, TSA '09, ACM (New York, NY, USA, 2009), 37–44.

3. Gao, C., Sang, N., and Tang, Q. On selection and combination of weak learners in adaboost. *Pattern Recogn. Lett. 31*, 9 (July 2010), 991–1001.

4. Ghiassi, M., Skinner, J., and Zimbra, D. Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network. *Expert Syst. Appl. 40*, 16 (Nov. 2013), 6266–6282.

5. Microsoft. http://msdn.microsoft.com/en-us/library/system.xml.xmldocument.aspx, 2012.

6. Nasukawa, T., and Yi, J. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2Nd International Conference on Knowledge Capture*, K-CAP '03, ACM (New York, NY, USA, 2003), 70–77.

7. O'Hare, N., Davy, M., Bermingham, A., Ferguson, P., Sheridan, P., Gurrin, C., and Smeaton, A. F. Topic-dependent sentiment analysis of financial blogs. In *Proceedings of the 1st International CIKM Workshop on Topic-sentiment Analysis for Mass Opinion*, TSA '09, ACM (New York, NY, USA, 2009), 9–16.

8. Potts, C. http://sentiment.christopherpotts.net/tokenizing.html, 2012.

9. Shamshurin, I. Data representation in machine learning-based sentiment analysis of customer reviews. In *Proceedings of the 4th International Conference on Pattern Recognition and Machine Intelligence*, PReMI'11, Springer-Verlag (Berlin, Heidelberg, 2011), 254–260.

10. Stanford. http://snap.stanford.edu/data/twitter7.html, 2012.

11. Stanford. http://www.stanford.edu/class/cs124/, 2012.

12. Thet, T. T., Na, J.-C., Khoo, C. S., and Shakthikumar, S. Sentiment analysis of movie reviews on discussion boards using a linguistic approach. In *Proceedings of the 1st International CIKM Workshop on Topic-sentiment Analysis for Mass Opinion*, TSA '09, ACM (New York, NY, USA, 2009), 81–84.

13. Twitter. https://dev.twitter.com/docs/rate-limiting, 2012.

14. Twitterizer. http://www.twitterizer.net/, 2012.

15. Wikipedia. http://datamin.ubbcluj.ro/wiki/index.php/evaluation_methods_in_text 2012.

16. Wikipedia. http://en.wikipedia.org/wiki/additive_smoothing, 2012.

17. Yang, J., and Leskovec, J. Patterns of temporal variation in online media. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, ACM (New York, NY, USA, 2011), 177–186.