USING TWITTER TO GAUGE NEWS EFFECT ON STOCK MARKET MOVES

SAM PAGLIA

1. Abstract

The rapid rate of data creation, interconnected graph structure and variety of data captured make Twitter an ideal candidate for characterizing the complicated and rapidly changing investor sentiments that move financial markets. In their 2010 work on the subject, Bollen et al[1] claimed that sentiment analysis on a broad Twitter corpus can lead to 87.6% accuracy in predicting daily changes in the Dow Jones Industrial Average.

In this project, I use a different method. Specifically, I take a news-based approach and aim to examine the specific effect of news releases via Twitter on market moves. Given that all major news outlets have twitter feeds that they run in parallel to their other distribution channels, this project aims to explore two key questions. The first is whether an aggregated body of news tweets can be used as the basis for a viable prediction strategy. The second is an analysis of what key phrases in tweets tend to be associated with directional market moves

My results indicate that, while news-feed analysis lacks significant predictive power as a binary classification tool of market directional moves, it does show promise as a multinomial classifier of market moves into discretized buckets. In this case, appropriately tuned Multinomial Naive Bayes and Multiclass SVM models show promise as Twitter-news-based predictors of market moves

2. Dataset Schema and Feature Choice

2.1. **Data Sources.** While the specific tweets I chose will be explained more fully below, I used the Twitter REST API to construct my dataset of tweets. From each tweet record, I parsed the tweet's text, authorship date, number of retweets and the id of its origin feed. I also added two additional features. One was the date of the next scheduled trading day after tweet authorship, for tweets before market close (4:00PM EST) this is just the authorship date, while for tweets after market close, it is the next trading day. The other was the number of minutes between the tweet's authorship date and the end of the next scheduled trading session.

For the market data, I downloaded the past three years of S&P 500 closing prices (P_t) from Yahoo Finance[2] and calculated the daily index returns $R_t = 1 - \frac{P_t}{P_{t-1}}$. I chose the S&P 500 as opposed to the DIJA because it is has more index components and is widely perceived to be a more broadly based measure of market movement. Use of alternative financial time series is explored in the final section. Iterating through my tweet set, I labeled each tweet (denote tweets labels l_i) with the percentage return R_t associated with the next scheduled trading close

I converted the text of each tweet into a count vector of word occurrences using a bag of words approach. For my corpus, I use a corpus of all words found across my tweet dataset. While I previously intended to use pre-existing corpuses composed of common english words, the frequent occurrence of abbreviated terms or proper nouns (which often grade out as strong market movers) made it advantageous to structure our corpus around the terms present

Additionally, I also excluded 25 extremely common english words (such as "is"), which, due to the short nature of the text being sampled, appeared disproportionally often and had a skewing effect on our sample.

2.2. **Dataset Composition.** The Twitter REST API makes only a user's 3200 most recent tweets available. In constructing my dataset, I faced a tradeoff, breadth (including feeds from many sources) versus depth (the maximum amount of samples from each feed). For this analysis, I accumulated a dataset of 115200 tweets, representing the full 3200-tweet timelines of 36 news sources, among them the Financial Times (@$FinancialTimes$), Reuters (@$Reuters$), The Wall Street Journal (@$WSJEconomy$), Forbes (@$Forbes$), and Bloomberg News (@$BloombergNews$)

These timelines are of unequal length, some span little more than 2 months, while some span almost 2 years.

3. Binary Classification

The initial problem we consider is simple binary classification problem. Given a tweet, can we predict whether the market rose or fell during its associated trading day? We transform our existing tweet labels l_i to $l_i' = 1(l_i > 0)$ and consider various classifier models.

3.1. **Naive Bayes.** We begin by fitting a Bernoulli Naive Bayes to our data. We use a laplacian smoothen parameter $\alpha = 2$ to help guard against the inherent sparsity of our dataset. The results are summarized in the table below. Note that, for this and all other models used in this paper, we employ a 70-30 train-test split of our dataset

Naive Bayes Diagnostics

Metric	Value
$P(l_i' = 1)$	41.4%
Training Error	29.6%
Test Error	41.0%
Precision	50.2%
Recall	20.8%

Date: Submitted December 13th, 2013.

As we can see, the substantial gap between training and test error indicates that this model is likely high variance, and would benefit from additional training data. When we isolate the 20 words most indicative of a market increase (words w_i with largest coefficients θ_i), our list includes such phrases as "china", "recession", "more", "up", "stocks", "fed", "trading" and "economy". While there are some entries that are difficult to explain ("500" for example), overall this model, while not performing particularly well, does show some promise

3.2. **Weighted Naive Bayes.** We also consider one nuance of this model, the addition of observation weights $w^{(i)}$. We would like to weight our examples so as to leverage two of the primary advantages of Twitter, its network structure and the near constant flow of timestamped data. In a 2012 paper on the subject of modeling influence decay of news events[3], Kong et al posit a formula for modeling news decay over time that is based on the Ebbinghaus forgetting curve[3], which states that memory retention for arbitrary phenomena (R) is equal to $R = e^{-\frac{S}{t}}$, where S is strength of memory and t is time.

One premise of Kong's paper is that the relevance of a news item (or a tweet) should be determined by the degree to which it has been retained in memory by its readers. In that sprit, we alter Ebbinghaus' formula somewhat to yield $R = w^{(i)} = \alpha n^{(i)} ln(\frac{\beta}{t^{(i)}})$, where $n^{(i)}$ is a tweet's retweet count and $t^{(i)}$ is the time in minutes until the close of the next trading day. The reason we have altered the formula in this way is that we would like to capture the fact that many retweets is likely indicative of a tweet's perceived relevance to those that view it.

Given this weighting methodology, we attempt to select the optimal α, β for model performance. We use grid search to optimize over pairs (α, β), recording the test error for each pair. Results shown below.

Grid Search Results for Optimizing Pairs (α,β)

β/α	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
5	41.28%	41.28%	41.28%	41.27%	41.30%	41.31%	41.44%	41.60%	41.86%	42.13%
10	41.28%	41.28%	41.28%	41.33%	41.40%	41.54%	41.69%	42.01%	42.25%	42.48%
50	41.28%	41.28%	41.55%	41.97%	42.52%	43.18%	43.68%	44.04%	44.25%	44.50%
100	41.26%	41.67%	42.55%	43.15%	43.78%	44.19%	44.57%	44.76%	45.06%	45.22%
250	41.73%	42.98%	44.01%	44.49%	45.05%	45.26%	45.39%	45.50%	45.63%	45.68%
500	42.63%	44.15%	44.77%	45.15%	45.44%	45.58%	45.61%	45.68%	45.79%	45.86%
750	43.24%	44.47%	45.22%	45.65%	45.77%	45.83%	45.89%	45.89%	45.88%	45.93%
1000	43.49%	44.77%	45.52%	45.82%	46.02%	46.04%	46.06%	46.08%	46.08%	46.09%
1250	43.60%	44.99%	45.68%	45.92%	45.98%	46.03%	46.15%	46.11%	46.12%	46.12%
1500	43.94%	45.20%	45.78%	45.97%	46.05%	46.15%	46.19%	46.16%	46.25%	46.30%
2000	44.18%	45.42%	45.95%	46.16%	46.24%	46.28%	46.26%	46.24%	46.31%	46.34%
3000	44.40%	45.83%	46.17%	46.41%	46.41%	46.44%	46.48%	46.51%	46.55%	46.59%

Rather frustratingly, there is no combination of weights α, β that yields superior performance to our basic NB implementation. As such, we conclude that memory-retention based model weighting is not an effective predictive strategy and move on to other model classes

4. Perceptron

We now fit a Perceptron model to our dataset as well, results displayed below:

Perceptron Diagnostics

Metric	Value
$P(l'_i = 1)$	41.4%
Training Error	48.3%
Test Error	48.6%
Precision	41.5%
Recall	45.4%

Unlike the previous case, the similarity between training and test error indicates that this is likely a high bias model. Furthermore, the fact that the test error is substantially larger than our Naive Bayes implementation indicates that this is likely not a model class that is worth pursuing further.

5. SVM

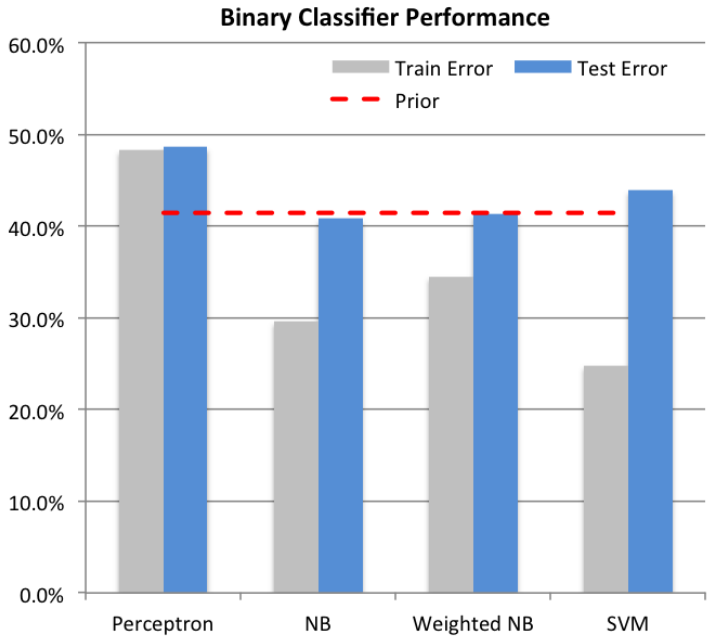
The final classifier we fit is a Linear Kernel L_2-regularized SVM. The sparse featuresets, few irrelevant features and linear separability of most classification problems makes SVM's an ideal candidate for text classification[5]. Diagnostics shown below:

SVM Diagnostics

Metric	Value
$P(l'_i = 1)$	41.4%
Training Error	24.8%
Test Error	43.7%
Precision	46.9%
Recall	52.6%

While the test error is higher than a Naive Bayesian implementation, the greater precision and especially recall indicates that our SVM implementation performed well in identifying positive market moves with greater accuracy. Furthermore, the large gap between training and test error indicates that the model could improve considerably with continued expansion of our dataset

5.1. **Binary Classification Wrapup.**



As the summary chart above indicates, while Naive Bayes and SVM implementations do show some promise, there does not appear to be a significant enough uptick in performance above a naive "Always predict 0" strategy to consider text-based classification a viable alternative to sentiment based methods, especially if Bollen et al's results are indeed correct. Admittedly, more better data and more expansive timelines could partially remedy this issue, but for now we move on to our next topic, multinomial classification

6. Multinomial Classification

One area in which news-based approaches show much more promise is the area of multinomial classification. We begin by outlining the discretization technique which we use to cluster our observations

6.1. **Observation Discretization, Error Metric and Labeling Methodology.** Looking at the descriptive statistics of the three year daily return history (Jan 1, 2010 - December 10, 2013) daily return history (R_t) of the S&P 500, we compute the following interquartile boundaries $q_{0.25} = -0.57\%, q_{0.5} = -0.007\%, q_{0.75} = 0.41\%$. We would like any partitioning to partition our dataset somewhat equally, as well as use a symmetric partitioning bound to avoid overfitting. We use these interquartile boundaries as the basis for the following 4-class partition: $P_1 = 1[R_t \leq 0.5\%], P_2 = 1[0.5\% < R_t \leq 0\%], P_3 = 1[0\% < R_t \leq 0.5\%], P_4 = 1[0.5\% < R_t]$.

Of more interest is how we label tweets depending on inclusion in these respective classes. In a 2008 paper, Fok et al[6]. posit that the mean absolute error, defined as $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, is a suitable error metric for stock price prediction. Furthermore, since we are dealing with discrete and not continuous output, MAE is a much better choice than other common error metrics such as RMSE. Hence, MAE will be our error metric of choice for the remainder of the paper

Having said this, we choose the following labeling schema $l_1 = 0, l_2 = 1, l_3 = 3, l_4 = 4$, where a tweet attains label l_i if its associated market move is contained in partition P_i. The reason we have set $l_3 - l_2 = 2$ as opposed to opting for a simple $l_i = i - 1$ labeling is that we would like our error metric to reflect the fact that directional errors are more severe than directionally correct magnitude errors. Increasing $l_3 - l_2$ will allow MAE to appropriately penalize models that commit directional errors more frequently. Now that we have defined our multinomial estimation framework, we move on to our actual implementations. We implement two model classes, Naive Bayes and Linear Kernel SVM, but use an unweighted and weighted variant of each

6.2. **Unweighted Multinomial Naive Bayes.** Once again using $\alpha = 2$ are our laplacian smoothing parameter, we fit an unweighted multinomial naive bayesian model to our data. Diagnostics shown below:

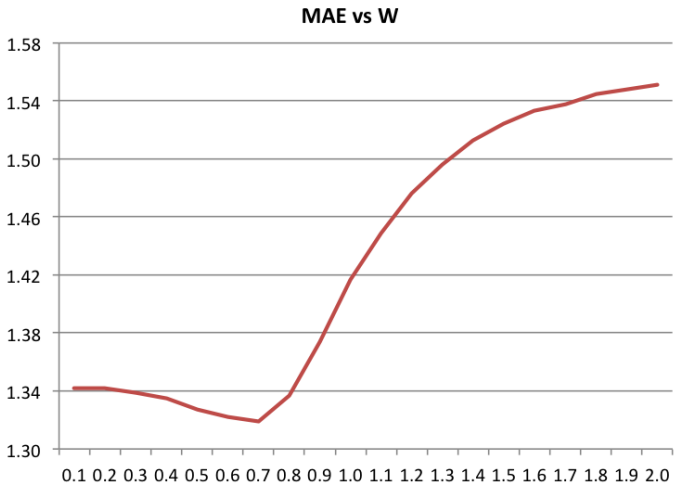
Multinomial Naïve Bayes Diagnostics

Test Error	64.2%
MAE	1.403

Fit Matrix	**lfit = 0**	**lfit = 1**	**lfit = 3**	**lfit = 4**
l = 0	13.7%	10.8%	2.7%	1.2%
l = 1	9.8%	16.2%	3.2%	1.2%
l = 3	8.1%	10.2%	4.0%	0.8%
l = 4	6.8%	7.2%	1.7%	1.8%

This model appears to perform well, with accuracy 5.8% better than any individual class prior probability. All told, our unweighted model makes a "directional error" (predicting that $y_i < 0$ when the opposite is true or vice versa) only 40.6% of the time, more accurate than a naive strategy. However, there are ways it can be improved as we show in the following section

6.3. **Weighted Multinomial Naive Bayes.** For a weighted model, we do not use a memory-retention weighting model as we did in our classification task (as it similarly unhelpful). Instead, we use a value-based weighting scheme as follows. If $l_i = 0, 4, w_i = W$, where W is a fixed parameter, otherwise $l_i = 1$. Essentially, W controls the degree to which we consider "extreme" observations more or less heavily than non-extreme observations. We perform a linear search over W to find the optimal value over our test set, the results of which can be seen below



As we can see, a value of $W = 0.7$ (meaning that extreme observations have 70% the weight of non extreme observations) minimizes our MAE at 1.319. Setting $W = 0.7$, we fit a weighted Multinomial Naive Bayes model. Diagnostics below:

Weighted Multinomial NB Diagostics

Test Error	65.8%			
MAE	1.319			
Fit Matrix	**lfit = 0**	**lfit = 1**	**lfit = 3**	**lfit = 4**
l = 0	5.2%	20.5%	0.7%	2.1%
l = 1	2.8%	24.7%	0.9%	1.7%
l = 3	2.3%	17.5%	1.6%	1.5%
l = 4	2.1%	11.1%	2.0%	2.7%

As we can see, despite a higher test error, the MAE of this model is significantly lower than in the unweighted case. Furthermore, the probability of directional error is now only 39%, less than in the unweighted case. Digging in to the specific key words isolated by our model, we see that our model isolates words such as "earnings", "debt", "down" and "recession" as indicators of a strong negative market move ($l_i = 0$), and words such as "growth", "up" and "sales" as indicators of strong positive move ($l_i = 4$). Finally, a few phrases, such as "fed", "trading" and "economy" are key predictors of both types of extreme market moves

In summary, weighted multinomial Naive Bayes can be a viable predictor for discretized market moves. While the partial overlap in predictive keywords between classes, as well as the fact that our current model has a notable bias toward labels l_0, l_1 (predicting market declines) indicate that there is room for improvement, the promise is notable. We now move on to multi class SVM's

6.4. **Unweighted SVM.** We now fit an one-vs-all multiclass Linear Kernel SVM to our data. Results shown below

One-v-Many SVM Diagnostics

Test Error	65.3%			
MAE	1.492			
Fit Matrix	**lfit = 0**	**lfit = 1**	**lfit = 3**	**lfit = 4**
l = 0	10.5%	7.1%	5.7%	5.2%
l = 1	7.2%	10.9%	6.4%	5.3%
l = 3	5.6%	6.4%	7.5%	3.9%
l = 4	4.6%	4.4%	3.6%	5.6%

While the overall performance is worse than that of our Naive Bayes, it appears that our confusion matrix is more uniform that in our Naive Bayes implementation, indicating that this model likely has fewer inherent prediction biases. Furthermore, this model has very low training error 26.7%, indicating that its performance may well improve with additional data

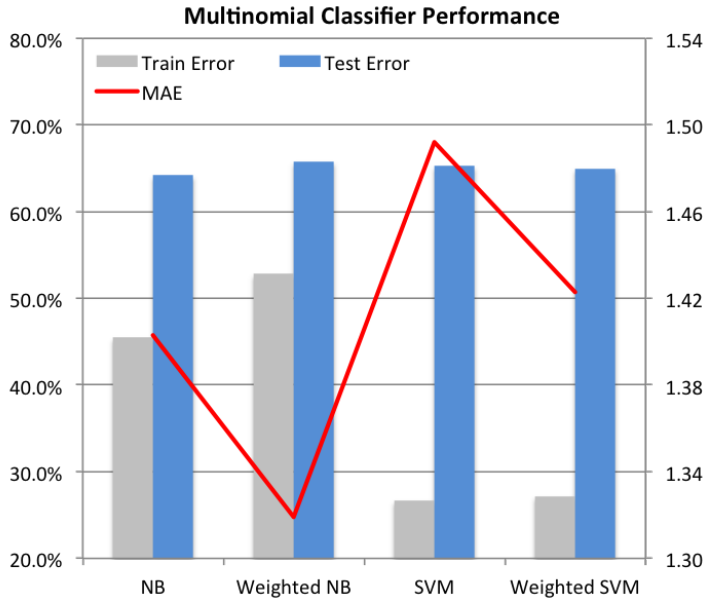
6.5. **Weighted SVM.** If we apply a similar value-based weighting method to our SVM implementation, and optimize our weighting parameters W, we find that our optimal $W = 0.5$. Using this value of W, I fit a weighted SVM. Diagnostics below:

One-v-Many Weighted SVM Diagnostics

Test Error	64.7%			
MAE	1.423			
Fit Matrix	**lfit = 0**	**lfit = 1**	**lfit = 3**	**lfit = 4**
l = 0	10.0%	9.5%	6.0%	3.1%
l = 1	6.6%	13.8%	6.4%	3.1%
l = 3	5.2%	8.3%	7.6%	2.3%
l = 4	4.3%	6.0%	3.9%	3.9%

As we can see the addition of weights does substantively improve our model's performance, but still leaves it below our Naive Bayes benchmark

6.6. **Multinomial Classification Wrap.**



As we can see from the chart above, given our current dataset, Naive Bayes models outperform Linear SVM's for our prescribed multinomial classification task, both with and without weights. However, the more uniform nature of the SVM-derived fit matrices and the large gap between SVM training and test error indicates that our SVM models may still be plagued by variance issues, and that with an even larger dataset, SVM models could outperform all other implementations in predicting discretized market moves.

Regardless, it is clear that tuned multinomial models are much more effective at predicting directional market moves than simple classification strategies. Furthermore, the introduction of value-based weights can further improve model accuracy, reducing both MAE and the probability of directional error.

7. Prediction of Other Financial Time Series

Now that we have completed our full analysis, turn our attention to one small aside. While we have seen that a news-based Twitter predictive approach can be effective in predicting movement of the equity market, we were curious how this performance would generalize to other market time series. The reason for this curiosity is thus: The obvious shortfall of a news-based approach to market prediction is that it ignores much of the sentimental signs affecting the human

beings driving the markets. Thus, logically speaking, news-based strategies will perform more poorly in markets that are dominated by sentiment versus rationality versus those that are not. To this end, we analyze three new data series, the yield on the 10-year treasury note, the price of gold (as tracked by the ETF "GLD") and the overall value of the US Dollar (as tracked by the ETF "UUP"). It would stand to reason that that, given that treasuries and gold are both "safe haven" assets that are typically purchased when investors are worried about the state of the economy, that these two assets would be excessively sentimental and difficult to predict, while the US Dollar, which is typically the domain of more seasoned investors, would be more rational and easier to forecast

Our methodology in each new market is the same we used in the equity market. We give tweets a label $l_i \in \{0, 1, 3, 4\}$ based on whether the tweets associate market movement falls into one of four partitions defined by boundaries $(-T, 0, T)$, where T roughly corresponds to the 75th percentile of the data series daily return set. We then cross validate an optimal weighting parameter W, and fit a weighted Naive Bayes to the data. The results for each market series, along with the optimal W and chosen T, are shown below

Other Indicator Performance

Index	Test Err	MAE	W	T
S&P 500	65.8%	1.32	0.5	0.5%
10-Yr T-Note	62.1%	1.342	1	1.5%
Gold	66.2%	1.605	1.1	0.6%
US Dollar	64.9%	1.382	1.1	0.3%

Looking at the above table, it appears that our results are partially borne out in that Gold does appear to be difficult to predict, while equities, treasuries and the dollar all lie in similar range. Furthermore, the contrast in optimal W between the S&P 500 and other assets is interesting to note. In summary, while it is not a hard and fast relationship, it does appear that news-based approaches are more effective in markets that are less influenced by sentiment

8. Summary

My results show that, while not particularly compelling in binary classification problem, news-based approaches can indeed be useful in multinomial classification of market moves, both in the equity market and otherwise. While, for our sample size, appropriately tuned Naive Bayes models show the most predictive power, with a $\epsilon_{test} = 65.8\%$ and $MAE = 1.319$, there is some evidence to suggest that, with a larger dataset and perhaps more fine tuning of observation weights, that a linear kernel one-vs-many multi class SVM could surpass Naive Bayes models as a accurate and informative predictor of discretized market moves

9. Works Cited

In addition to the papers and resources cited below, I would like to thank Andrew Ng and the entire CS 229 teaching staff for their invaluable help and instruction. Furthermore, special mention is needed for the Python sklearn module, which I used for my off-the-shelf model fitting. Documentation for the module can be found here (http://scikit-learn.org/stable/documentation.html)

1) Bollen, J.; Mao, H.; and Zeng, X.-J. 2010. Twitter mood predicts the stock market Journal of Computational Science 2(1):18.
2) "http://finance.yahoo.com/q/hp?s=%5EGSPC+Historical+Prices"
3) Kong et al, Ranking News Events by Influence Decay and Information Fusion for Media and Users (2012) CIKM 12, pg 1849-1853
4) H. Ebbinghaus. "Memory: A Contribution to Experimental Psychology". Teachers College, ColumbiaUniversity, 1913.
5) Manevitz, L. M. and Yousef, M. 2002. One-class SVMS for document classication. J. Mach. Learn. Res. 2, 139154.
6) Dr. Wilton.W.T. Fok, IAENG, Vincent.W.L. Tam, Hon Ng, Computational neural network for global stock indexes prediction , Proceedings of the World Congress on Engineering, 2008, Vol II WCE 2008, July 2 - 4, 2008, London, U.K.

Other
7) C. Ornes and J. SklanskKita et al, Application of Bayesian Network to stock price prediction (2012) DOI: 10.5430/air.v1n2p171
8) Ke Tao, Fabian Abel, Claudia Hauff, and Geert-Jan Houben. What makes a tweet relevant for a topic? Making Sense of Microposts (# MSM2012), page 9, 2012