STAT 222: Final write-up

Taehee Jung, SID:3032130037

Abstract

Our main task is to develop an algorithm that will create and update a portfolio of stocks from among the Standard and Poor's 500 and United States Treasury Bonds. Our goal is a sparse portfolio that achieves returns similar to those of the S&P Index, with lower volatility. Stock selection is achieved by solving an optimization problem. The original optimization routine came from Brodie et al's paper **[1]**, but we modified the objective function by including a vector of predicted stock volatilities, in order to emphasize our goal of minimizing overall portfolio volatility. In order to predict the volatility of each stock, we used Vector Autoregressive (VAR) model with added feature volume and a 24 month training window. Our selection algorithm showed very strong results during the testing period (2014-2016), achieving higher returns and lower volatility than both the S&P 500 index[1] and a naive portfolio[2].

1. Introduction

Creating a profitable but stable portfolio - one that grows steadily without high volatility - is a desirable and challenging goal for many who invest in the stock market. Therefore, our project is derived from this key question: "Can we build a portfolio of stocks and bonds to maximize profit and minimize the risk at the same time?"

To achieve this goal, we applied the methodology described in "Sparse and stable Markowitz portfolios" (Brodie et al., 2009). The traditional Markowitz framework (aka mean-variance analysis) is a simple but strong theory to minimize the risk of the portfolio for a specified expected return ρ. However, this approach requires inverting a matrix of predict returns for different assets, which are often highly correlated with one another. As a result, the matrix can be near-singular, and therefore the solution can be highly unstable: small changes in expected returns lead to large fluctuations in the optimal weights. To address this problem, Brodie et al. suggest supplementing the original Markowitz objective function by adding a regularization factor (which also encourages sparseness), namely an l_1 penalty on the portfolio weights. We incorporate this idea into our portfolio optimization process, and also add a term with volatility in the objective function, in order to minimize the variance of portfolio. More details are covered in 3.2.

Predicted return and volatilities are used as inputs on the portfolio. Brodie et al. implemented an optimized portfolio using 12-month simple moving average and surpassed the results of the naive portfolio during the test period (the naive portfolio divides money equally among all available assets). Papers from Goyal and Welch (2006) **[2]** and Campbell and Thompson (2008) **[3]** also showed that a simple moving average can be a more accurate predictor than the results from autoregressive models. Therefore, we decide to adopt 12-months simple moving average as a predicted return. On the other hand, volatility can be forecasted by autoregressive models because much empirical research shows that it is autocorrelated. Thus, we tested some time-series models with different parameters and find a best predictive model: section 3.1 explains more about this. Additionally, we try to test if external information coming from social network service (SNS) can be helpful to predict the volatility. Section 2.2 shows a way to extract sentiment features from tweets.

2. Data description

2.1 Stock prices and dividends

Stock price is used as the main information for this project. We collected daily stock prices from 'Yahoo! Finance' for the companies *existing in the S&P 500 in February, 2017*. Historical stock prices for each company were collected from *January, 1980* (initial time) or the date a given company came into the S&P 500 if it was later than the initial time. In this condition, total number of stocks will increase as time goes on from 1980 to 2016, but no stocks 'drop out' of our data, since we consider only stocks that were in the Index as of February 2017. We split the stock data into two periods: training period from 1980 to 2013 and the testing period from 2014 to 2016.

---

1. Weighted average of all companies in S&P 500
2. Portfolio having equally-distributed weights among assets (= 1/n investment strategy)

Using the bash script, csv files of (exactly) 505 companies have been collected. These files includes ticker (e.g 'AAPL' represents Apple Inc.), volume and stock prices. Note that among 5 types of stock prices (open, high, low, close and adjusted close), we only use the adjusted close price for this project. Returns in a given period are calculated based on the adjusted close price. Also, volatility is calculated as the variance of daily returns in a given month. Equation 1 shows how we calculate these quantities. Note that r_t and v_t represent monthly returns and volatility for a given stock in month t.

[Equation 1] How to calculate stock returns and volatility

$$r_t = \frac{p_t - p_{t-1}}{p_t} \text{ where } p_t = \text{adjusted close price in time t}$$

$$v_t = \frac{1}{n_t}\Sigma_{i=1}^{n_t}(r_i - \hat{r})^2$$

where $r_i =$ return for the i^{th}day of month t, $n_t =$ number of training days in month t

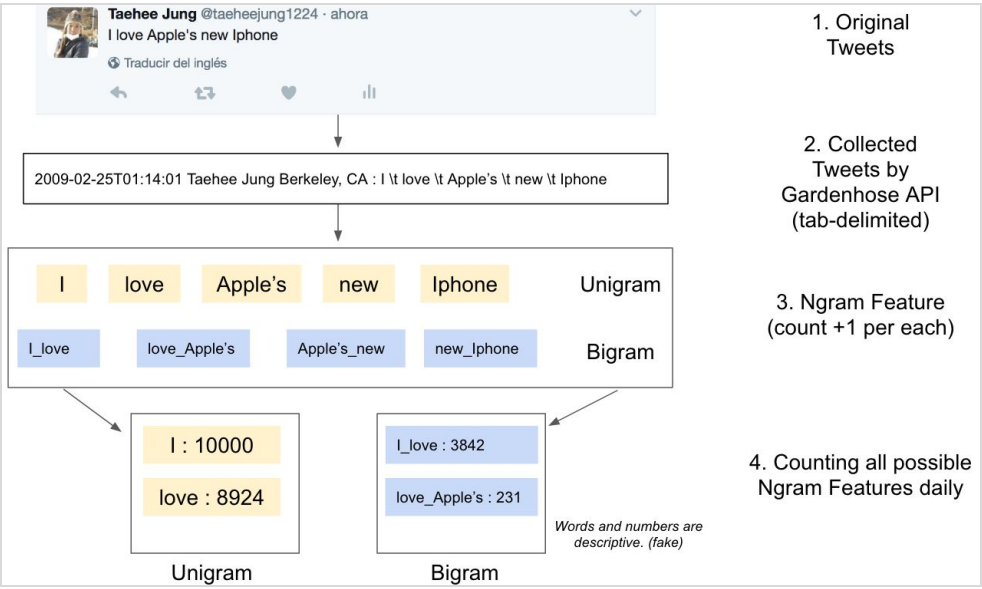
2.2 Twitter

The efficient-market hypothesis states that financial market movements depend on news, current events and product releases, and all these factors will have a significant impact on a company's stock price. Fortunately, research has been conducted to show that sentiment analysis from Twitter can be a source of new information for the future stock market **[4][5][6].** We therefore decided to extract some sentiment features from the Twitter data set which are highly correlated with each company.

First, note that we use sample tweets that are crawled with Twitter's Gardenhose API, which was originally collected by Carnegie Mellon University for an academic purpose. 10% of public tweets were recorded from September 10th, 2008 to December 31th, 2013 (approximately one million tweets per day). Each tweet provides a tweet identifier, submission date-time, and the text content of the Tweet, which is limited to 140 characters.

We extracted N-gram features**[7]**, particularly unigrams and bigrams. Here, the term 'n-gram' refers to the set of co-occurring words within one tweet. First, all unigram and bigrams words were extracted. Then, day-by-day frequencies for each word were counted. As a result, date and frequency of each word were saved in separate files. Average daily frequencies of each unigram and bigram are used to check the frequently mentioned words in Twitter. Figure 1 shows how N-gram features are extracted from twitter.

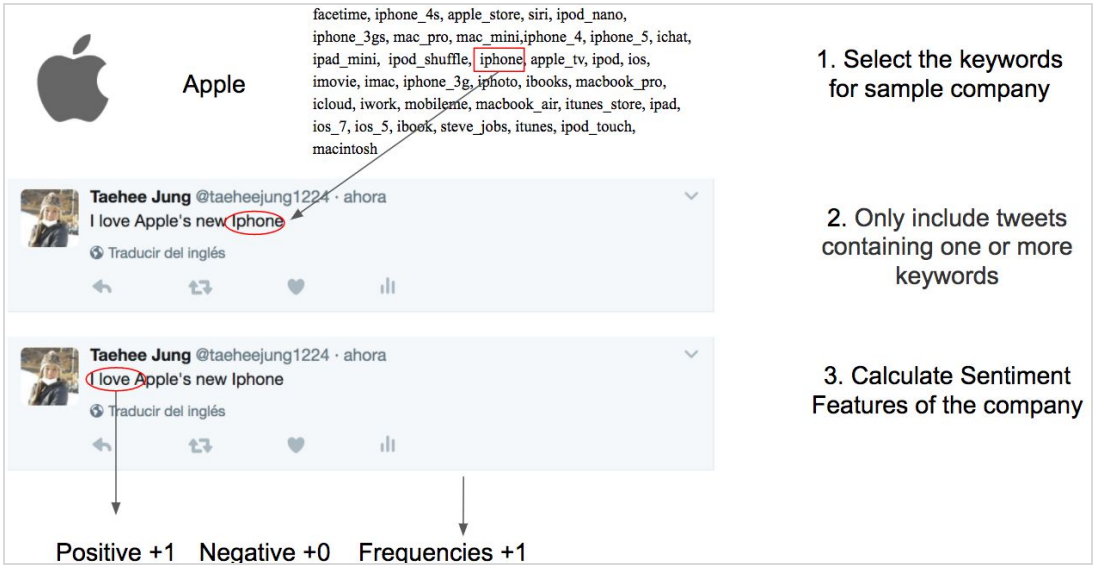
[Figure 1] How N-gram features are created (description with one example)



Sentiment features (frequencies, positive and negative) of selected companies were calculated as shown below. First, we chose highly correlated 'keywords' for each company (for instance, tweets about the iPhone refer Apple, while tweets about YouTube refer to Alphabet/Google). Then, we filtered out tweets which do contain

keywords. From these filtered tweets, daily sentiment frequencies for each company were obtained by counting positive and negative words using OpinionFinder (Wilson et al., 2005) **[8]**. Figure 2 shows how the sentiment features of each sample company are finally created.

[Figure 2] How sentiment features are calculated (description with one sample)



Finally, we want to learn whether sentiment features can improve the accuracy of the volatility prediction model. Because of time constraints, and because not all companies are actively discussed on Twitter, we limit our research to a small sample of companies with extensive Twitter data: specifically, companies whose names are mentioned with high frequency on Twitter.

3. Methods

3.1 Forecasting monthly volatility

We considered several different models to predict volatility: a simple moving average (SMA) model is used as the baseline model for the volatility prediction. It is calculated by averaging previous volatilities in time lag k (the previous k months).

The Autoregressive model (AR) expands on the SMA by adding an 'intercept' term and allowing the coefficients on the previous k monthly volatilities to vary arbitrarily. Equation. 2 shows the basic equation of the AR model for this project. Predicted value y_t represents the monthly average volatility of one company in month t. Each predictor in the model is the true volatility of this company in a previous month. Estimation window k represents a time lag - the number of previous months the model considers.

[Equation 2] AR(k)

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \ldots + \beta_k y_{t-k}$$

The Vector Autoregressive model (VAR) also allows features besides previous volatilities to be used in the model. In our project, these other features can include past transaction volume, and Twitter sentiment features. In Equation 3, c is a 2 x 1 constant vector, predicted matrix is a 2 x 1 matrix and is a function of monthly average volatility (y_t) and additional feature(f_t) in month t. A_i is a time-invariant 2 x 2 matrix. Again, estimation window k is a time lag. Note that we are only interest y_t even though this model also forecasts f_t.

[Equation 3] VAR(k)

$$\begin{bmatrix} y_t \\ f_t \end{bmatrix} = c + A_1 \begin{bmatrix} y_{t-1} \\ f_{t-1} \end{bmatrix} + A_2 \begin{bmatrix} y_{t-2} \\ f_{t-2} \end{bmatrix} + \ldots + A_k \begin{bmatrix} y_{t-k} \\ f_{t-k} \end{bmatrix}$$

3.2 Optimizing stock portfolio

Suppose that we have N assets available for purchase, and the expected return of each asset can be expressed as R. Also, assume that we set up a specific goal for the total returns which can be written as ρ (ie. ρ = 0.2 means 20% gain in a given period). In Markowitz portfolio, the objective function can be constructed with 2 constraints; (1) sum of portfolio weights should be 1, meaning that we will invest one unit of capital and (2) expectation of total returns for the portfolio is equal to ρ. Equation 4 describes this.

[Equation 4] Markowitz portfolio : Objective function

To minimize $w^T \Sigma w$ subject to $R^T w = \rho$

where $w =$ vector of portfolio weights

R = a vector of expected returns

Σ is the covariance matrix for the returns on the assets in the portfolio

$w^T \Sigma w$ = a variance of the portfolio

In practice, columns in R are highly correlated with each other, so this yields an instability in optimization, as described above. Brodie et al., therefore, added an l_1 penalty to the Markowitz objective function. This penalty stabilizes the optimization problem, and encourages a sparse portfolio (with many 0s in the portfolio weight vector). Further, this penalty can be interpreted as transaction costs: investors are discouraged from taking numerous 'short positions' because it increases transaction costs.

Using Brodie et al.'s optimization procedure as a starting point, we constructed our own objective function. In our exact problem, we seek to minimize volatility while achieving a specified return. We modified this exact problem by moving some of the constraints into the objective function, in order to find an approximate solution. The advantage of this approximate approach is that it can be solved easily with the R package GLMNET. To update the portfolio, we solve a similar optimization problem to find the optimal 'update vector,' which represents changes to our existing allocations, such as buying or selling a stock. Just as this procedure yields a sparse portfolio, it also results in sparse updates: in a typical month, we make either zero transactions or only a few transactions. Figure 3 describes how we optimize our portfolio.

[Figure 3] Our portfolio description

What we want: $\underset{\mathbf{w}}{\text{argmin}} \|\hat{\mathbf{v}}^\intercal \mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1$

subject to the following constraints:

1. $\mathbf{1}_N^\intercal \mathbf{w} = 1$ (weights sum to 1)
2. $\hat{\boldsymbol{\mu}}^\intercal \mathbf{w} = \rho$ (achieves target predicted return)
3. $\mathbf{w}_i \geq 0, \forall i \in \{1, ..., N\}$ (no short positions)

What does R's package glmnet do? $\underset{\mathbf{w}}{\text{argmin}} \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1$

How to solve our problem (approximately) using glmnet?

Let X be the $3 \times N$ matrix: $X = [\hat{\mathbf{v}}, \hat{\boldsymbol{\mu}}, \mathbf{1}_N]^\intercal$

Let Y be the vector of length 3: $Y = [0, \rho, 1]^\intercal$

Our observation weights are $[1, \epsilon, \tau]^\intercal$, where the second element controls the importance of $\hat{\boldsymbol{\mu}}^\intercal \mathbf{w} = \rho$ and the third element controls of the importance of $\mathbf{w}_i \geq 0, \forall i \in \{1, ..., N\}$.

Our new, approximate optimization problem is:

$$\underset{\mathbf{w}}{\text{argmin}} \underset{\textbf{[1]}}{1\|\hat{\mathbf{v}}^\intercal \mathbf{w} - 0\|_2^2} + \underset{\textbf{[2]}}{\epsilon\|\hat{\boldsymbol{\mu}}^\intercal \mathbf{w} - \rho\|_2^2} + \underset{\textbf{[3]}}{\tau\|\mathbf{1}_N^\intercal \mathbf{w} - 1\|_2^2} + \underset{\textbf{[4]}}{\lambda\|\mathbf{w}\|_1}$$

Additional constraint: $w_i \in [0, 0.1]$ meaning that no individual stocks can make up more than 10% of our initial portfolio (this improves volatility). Also note that bonds are included in the portfolio but not in the optimization problem. They are just tacked on at the end.

Term [1] tries to minimize the weighted volatility of the portfolio.

Term [2] tries to achieve the target level of return.

Term [3] tries to make the sum of weights equal to 1.

Term [4] tries to make all weights to 0.

As a result, four terms play a different role in our objective function. ϵ, τ and λ constrain the weight of each term in a function. These parameters can be also interpreted as weights of each term proportional to the first term [1] .

4. Results

4.1 Sentiment analysis for sample companies

4.1.1 Company selection

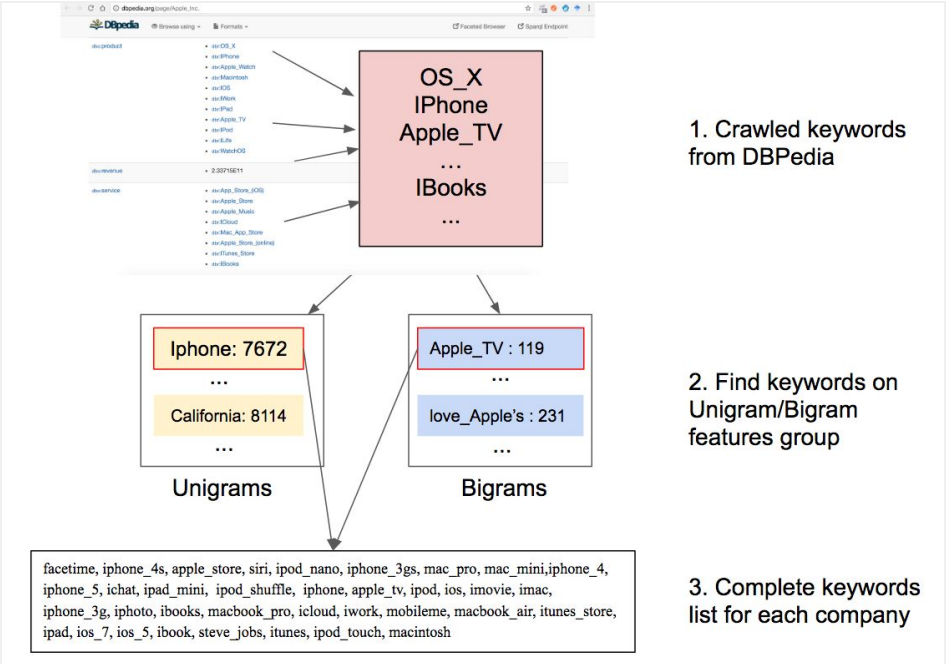
We first calculated the total frequencies of S&P 500 companies' name from 2009 to 2013 based on frequencies of unigram/bigram features. Appendix A contains the result. Unsurprisingly, only 26 companies were actively mentioned on Twitter. Among them, we chose 13 companies which have more than 50,000 mentions of their company names. In this process, Google, Facebook, HP, Microsoft, Yahoo, Apple Inc. eBay, AT&T, Nike, Intel, Paypal, Oracle, Amazon were selected.

4.1.2 Keyword selection and expansion

For a better forecasting performance, we need highly correlated keywords for each company. For example, Google may have high correlation with such keywords: "Android" a product of Google or "Sundar Pichai" CEO of the company. Collecting such keywords manually is very difficult, so we propose an automatic keyword selection process.

Our keyword selection method uses an external knowledge base called DBPedia **[9]**. It has structured or unstructured crowd-sourced information about people, places, and organizations, including companies. For example, the company PayPal has a page on DBPedia (http://dbpedia.org/page/PayPal) that includes a general description of the company, its founders, location, products, and so on. We crawled the company's DBPedia page and extracted the n-gram words associated with each company. Then, if these words exist in our unigram/bigram features, we chose them as our keywords for each company. However, the resulting list of keywords still had some 'noise' so we manually filtered out few of them. In this process, 8 companies for which we found no meaningful keywords were also dropped. Figure 4 shows how we choose keywords for each company and appendix B shows the selected companies and keywords chosen by our method. In final, Google, Apple, Facebook, Amazon and Microsoft were selected as sample companies for predicting volatilities with sentiment features.

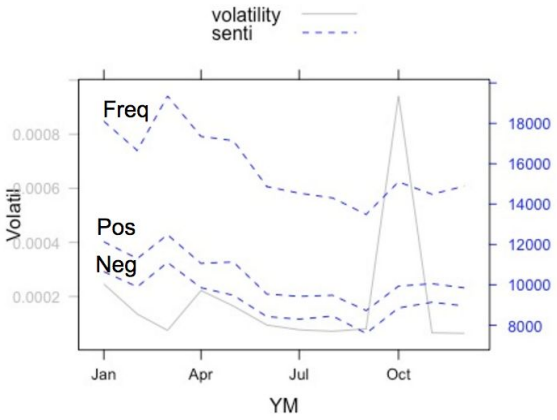
[Figure 4] How to choose keywords for each company (Example of APPLE inc.)



4.1.3 Sentiment features vs monthly average volatility

Before we move to the forecasting process, we simply plot the graph for the monthly average volatility versus sentiment features for each selected company. Figure 5 describes how actual monthly volatility and the corresponding sentiment features for Google are aligning. In the graph, the grey line represents monthly volatility on the left y axis, meanwhile each of 3 dotted blue lines means total, positive and negative tweets about Google (right y axis). As a result, patterns from different sentiment features are very similar. Also, it looks like there is a correlation between volatility and sentiment features.

[Figure 5] Volatility and sentiment features (Google, 2013)



4.2 Forecasting monthly volatility

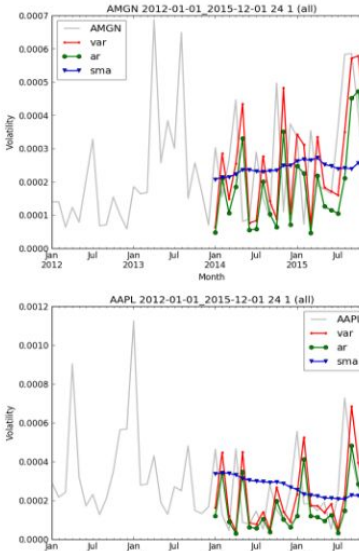
4.2.1 Experiment

To find a best model, we tested 48 different combinations of method, training window size and steps. One parameter, training window, means the length of the period for training. Another parameter, steps, represents the total number of months predicted at one time. For example, a model with training window = 6 and steps = 2 uses the past 6 months to predict volatility for the upcoming 2 months. In total, we predicted the monthly volatility of each company throughout the entire testing period and got the average root mean squared error (RMSE). Table 2 shows the summary of our experiment. Based on the result, we chose a model with lowest average RMSE, which is VAR with 24 months training window and 1 month of step size.

[Table 1] RMSEs of tested models

Training window(=t)	Steps	RMSE		
		AR	VAR	SMA
6	1	0.000447	0.000410	0.000372
6	3	0.001927	0.005971	0.000435
6	5	0.051782	1.159149	0.000473
12	1	0.000399	0.000375	0.000393
12	3	0.001159	0.000929	0.000461
12	5	0.018620	0.018547	0.000501
18	1	0.000383	0.000360	0.000415
18	3	0.001029	0.000504	0.000482
18	5	0.016111	0.000948	0.000523
24	1	0.000378	**0.000358**	0.000436
24	3	0.000989	0.000479	0.000503
24	5	0.015571	0.000664	0.000545

[Figure 6] Actual vs Predicted Volatility



4.2.2 Comparing the best predictive model versus VAR models with sentiment features

In the previous section, we already chose our best model for volatility forecasting. Now we wanted to learn whether we could gain more accuracy by including the sentiment features we created. We fitted models for 5 sample companies based on data from 2009 to 2013 and compared the RMSE of a VAR with sentiment features with the RMSE of the best prediction model without sentiment features. Table 4 represents a summary for this test. Unfortunately, we did not find evidence that a VAR with sentiment features consistently out-performed a VAR without sentiment features.

[Table 2] RMSE Comparison of VAR models with different features (For selected companies)

Company	Training window	Steps	RMSE			
			+ volume (No Twitter)	+ tweet freq (Twitter)	+ positive tweets (Twitter)	+ negative tweets (Twitter)
AAPL	24	1	0.000195	0.000242	0.000235	0.000295
AMZN	24	1	0.000074	0.000198	0.000194	0.000239
MSFT	24	1	0.000149	0.000181	0.000179	0.000199
FB	18	1	0.000084	0.000109	0.000108	0.000495
GOOGL	24	1	0.000285	0.000270	0.000266	0.000289

4.3 Optimizing Stock portfolio

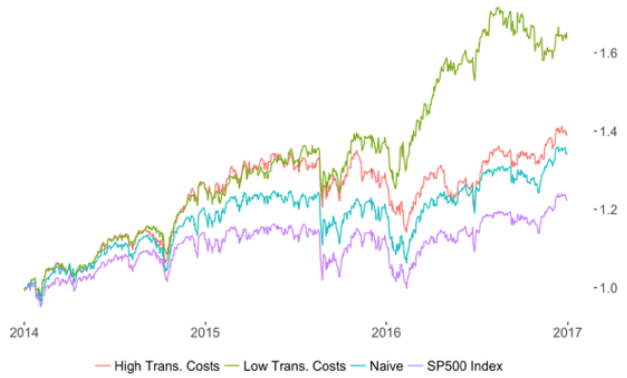
We trained a portfolio based on the dataset from 1980 to 2013. Comparing 3,000 different predictors, we got the optimal parameters; 5(τ), 1(ϵ), and 0.016(ρ). We did not tune λ, since it represents transaction costs, which are not something an investor can typically control. We created two different portfolios based on different assumptions about transaction costs: With the high transaction costs portfolio, we used $\lambda = 0.2$ for initializing and updating process. On the other hand, for the low transaction cost portfolio, we initialized with λ=0.2, but then changed it to 5e-05 for all updates. We observe that small λ leads to increasing number of active positions in the updating process.

We compare our portfolios with both S&P 500 index and naive portfolio strategy. Table 5 shows the result. As we can see, our strategies beat both of them with higher returns and smaller volatility. We can also see the trade-off between returns and volatility by comparing our two portfolios.

[Table 3] Optimal portfolios vs actual S&P 500 index vs naive portfolio

Portfolio	Annual Returns	Volatility	Relative Volatility	# Active Positions	
				Start	End
High Transaction Costs	11.6%	5.9E-05	82%	19	19
Low Transaction Costs	17.9%	6.4E-05	89%	19	29
SP 500 Index	6.9%	7.1E-05	100%		
Naive Portfolio	10.2%	7.5E-05	105%		

[Figure 7] Optimal portfolios vs actual S&P 500 index vs naive portfolio



5. Conclusion

We constructed a sparse and stable portfolio using predicted volatility and returns. We showed that our portfolio algorithm can give stable and relatively high returns to an investor during the testing period. The following are suggestions for future research to improve the portfolio algorithm:

First, we can explore more features for the volatility prediction. One idea is incorporating previous volatility of stock X to forecast the volatility of stock Y. External features such as P/E ratio can be considered. Second, we only considered treasury bonds with 10-year maturity dates in our portfolio. Therefore, treasuries with different maturities can be included in a future portfolio. Further, we initialized the portfolio with 20% of capital in bonds, but did not buy or sell any bonds thereafter. In the future, we could incorporate annual re-balancing to maintain the initial 20% allocation. Since bonds provide stability, we expect this would help the portfolio maintain low volatility over time.

6. References

[1] Brodie, Joshua, et al. "Sparse and stable Markowitz portfolios." *Proceedings of the National Academy of Sciences* 106.30 (2009): 12267-12272.
[2] Welch, Ivo, and Amit Goyal. "A comprehensive look at the empirical performance of equity premium prediction." *Review of Financial Studies* 21.4 (2008): 1455-1508.
[3] Campbell, John Y., and Samuel B. Thompson. "Predicting excess stock returns out of sample: Can anything beat the historical average?." *Review of Financial Studies* 21.4 (2008): 1509-1531.
[4] Bollen, Johan, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." *Journal of computational science* 2.1 (2011): 1-8.
[5] Pagolu, Venkata Sasank, et al. "Sentiment Analysis of Twitter Data for Predicting Stock Market Movements." *arXiv preprint arXiv:1610.09225* (2016).
[6] Mittal, Anshul, and Arpit Goel. "Stock prediction using twitter sentiment analysis." *Standford University, CS229 15* (2012).
[7] https://en.wikipedia.org/wiki/N-gram
[8] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/
[9] https://en.wikipedia.org/wiki/DBpedia

Appendix A

[Table] Top 26 frequencies of S&P 500 companies name on top 10,000 frequent unigram/bigram features

Rank	Company	Frequencies	Rank	Company	Frequencies
1	**Google**	1049339	14	**Staples**	37077
2	**Facebook**	883326	15	**Whole Foods**	32623
3	**HP**	534074	16	**Citi**	21523
4	**Microsoft**	343384	17	**Hershey**	19698
5	**Yahoo**	310254	18	**Time Warner**	19186
6	**Apple**	297567	19	**Walt Disney**	18550
7	**eBay**	284309	20	**Goldman Sachs**	16151
8	**AT&T**	160972	21	**Wells Fargo**	15966
9	**Nike**	155339	22	**Dr Pepper**	12755
10	**Intel**	88224	23	**American Airlines**	12000
11	**PayPal**	68615	24	**Urban Outfitters**	11303
12	**Oracle**	57574	25	**Chevron**	10581
13	**Amazon**	51314	26	**Mastercard**	10467

Appendix B

[Table] Final keywords after keyword expansion and filtering

Company	DBPedia URL	Expanded keyword list	Final keyword after filtering
Google	http://live.dbpedia.org/page/Google	google_voice, google, chrome_os, picnik, google_calendar, google_maps, goggles, gmail, redirect, the_google, google_news, google_buzz, adwords, adsense, internet, =, math, @google, business, company, google_reader, convert, knol, google_chrome, california, linkedin, nexus_one, about, google+, url, youtube, picasa, admob, igoogle, agent, google_analytics, orkut, anchor, software	google_voice, google, chrome_os, picnik, google_calendar, google_maps, gmail, google_buzz, adwords, google_reader, knol, google_chrome, nexus_one, google+, youtube, picasa, igoogle, google_analytics
Apple	http://dbpedia.org/resource/Apple_Inc.	facetime, apple, iphone_4s, apple_store, siri, ipod_nano, nasdaq, decrease, cups, redirect, iphone_3gs, mac_pro, mac_mini, tap_tap, iphone_4, iphone_5, ichat, ipad_mini, chairman, ipod_shuffle, company, iphone, apple's, apple_tv, business, ceo, quote, ipod, ios, imovie, imac, iphone_3g, iphoto, ibooks, california, macbook_pro, icloud, iwork, mobileme, macbook_air, itunes_store, ipad, about, ios_7, ios_5, ibook, url, steve_jobs, itunes, airport, agent, iad, ipod_touch, convert, macintosh	facetime, iphone_4s, apple_store, siri, ipod_nano, iphone_3gs, mac_pro, mac_mini,iphone_4, iphone_5, ichat, ipad_mini, ipod_shuffle, iphone, apple_tv, ipod, ios, imovie, imac, iphone_3g, iphoto, ibooks, macbook_pro, icloud, iwork, mobileme, macbook_air, itunes_store, ipad, ios_7, ios_5, ibook, steve_jobs, itunes, ipod_touch, macintosh
Facebook	http://dbpedia.org/resource/Facebook	instagram, agent, nasdaq, usd, microblogging, chairman, internet, whatsapp, mark_zuckerberg, business, company, update, social_network, fb, social_networking, facebook, photo_sharing, about, url, clear, steady, friendfeed, nearby	instagram, whatsapp, mark_zuckerberg, fb, facebook
Amazon	http://live.dbpedia.org/page/Amazon.com	agent, nasdaq, decrease, seattle, since_then, ebooks, woot, quote, zappos, business, amazon_kindle, company, internet, amazon, amazon.com, convert, us$, flag, e-commerce, kindle_fire, small, anchor	zappos, amazon_kindle, amazon, amazon.com, kindle_fire
Microsoft	http://live.dbpedia.org/resource/Microsoft	comic_sans, refs, linkedin, nasdaq, decrease, skype, internet_explorer, usd, windows_mobile, windows_vista, kinect, msn, chairman, microsoft_office, yammer, new_mexico, business, windows_8, windows_7, microsoft's, company, update, windows_xp, agent, zune, msft, xbox_360, bill_gates, center, xbox_live, xbox, ms, microsoft, software	skype, internet_explorer, windows_mobile, windows_vista, kinect, msn, microsoft_office, yammer, windows_8, windows_7, microsoft's, windows_xp, zune, msft, xbox_360, bill_gates, xbox_live, xbox, ms, microsoft