Predicting Stock Movements Using Market Correlation Networks

David Dindi, Alp Ozturk, and Keith Wyngarden
{ddindi, aozturk, kwyngard}@stanford.edu

1 Introduction

The goal for this project is to discern whether network properties of financial markets can be used to predict market dynamics. Building on previous work involving networks derived from market price correlations, we augment basic price correlation networks with additional information (revenue, sentiment, and news-flow). Our intuition is that these alternative networks will capture relationships beyond price correlations (e.g. business model exposures) that could eventually enhance downstream predictive models. The final insight we aim to provide is a prediction of future market behavior based on features that incorporate both standard trading information (price, volume, etc.) and market network characteristics (centrality, clustering coefficient, etc.).

The project methodology composes of three components: structural, analytical, and predictive. In the structural component, we filter the data to find and visualize the underlying structural motifs of the network. In the analytical component, additional metrics are computed for graphs built from the full dataset, and we do statistical testing to see whether our graph features have predicting power for stock prices. These two components use correlations of both prices and the newly introduced news/sentiment variables when building networks. They also featurize properties of our market correlation networks for sub-periods of years or quarters to see how these networks change over time. Finally, the predictive component incorporates features/metrics generated by the structural and analytical components into a recurrent neural network (RNN) to predict binary market movements (up/down) over a future period of interest.

2 Related Work

There have been several previous explorations of graphs built from stock market prices, where stocks are nodes and correlations in price movements are edge weights. Tse, Liu, and Lau (2010)[1] show that this type of graph built from US equities has power-law degree distributions under sufficiently high correlation thresholds. The authors built networks from correlations in daily closing prices, price returns, and trade volumes. All three networks had degree distributions following a power law with sufficiently high thresholds, though the power law exponent varied. The authors did not attempt to use these networks to predict future price movements, but instead used high-degree nodes to automatically create new stock indexes to track performance of the entire market. Their basic network and thresholding setup is the starting point for our structural and analytical components.

In an earlier paper, Boginski, Butenko, and Pardalos (2004)[2] explored structural differences between a similar graph structure built over daily price return correlations and the complementary graph containing edges with correlations below the threshold. The complementary graph was intended to represent independent equities which could form a diversified index fund. However, the authors found several structural properties present in the thresholded network but not the complementary graph, including a high clustering coefficient and the existence of very high-degree nodes. In addition to exhibiting scale-free behavior, the thresholded correlation network allowed automatic node clustering, while this task was much more difficult for the complementary graph. We will keep these advantages in mind by focusing on just the above-threshold part of any threshold applied to a network.

There is also some previous work on predicting future financial movements from noisy, non-stationary time series data. Tsoi et al (2001)[3] focused on predicting future foreign exchange rates based on noisy, low-volume time series data from prior exchange rates, which closely matches our task. We follow some of the authors' techniques here, including data range reduction and quantization and the use of RNNs.

Stock price prediction is a common task for new series forecasting methods. The efficient market hypothesis from the field of economics implies that time series of stock prices are unforecastable, since the market automatically incorporates all information currently known into price. Timmermann and Granger (2004)[4] explore the efficient market hypothesis with respect to potentially novel forecasting techniques, noting that new techniques may have short-term success because the knowledge they provide is not immediately incorporated into the market at scale. However, the authors note that applying a successful forecasting technique affects prices and causes the technique to self-destruct in the long term. Still, it will be interesting to determine whether a RNN with market-based features has predictive power within a controlled dataset.

3 Data Collection and Preprocessing

We aggregated daily time-series of all 2800 companies traded at the New York Stock Exchange (NYSE) between January 2010 and October 2015 from Bloomberg Market Data Services. The primary variables retrieved were closing price, high price, low price, trading volume, market cap, daily number of news stories and twitter sentiment. Additionally, we obtained descriptive information about every company; this included field such as the Global Industry Classification Standard (GICS) sector codes as well as the main country of operation.

3.1 Preprocessing for Model Input

Using raw stock-specific daily data points as features would lead to poor generalization due to the non-stationarity and noisiness of the time-series data. We thus performed transformations on the raw values for the basic variables (closing price, number of news stories, etc.) designed to combat each issue before organizing the data for model input.

3.1.1 Differencing and Dynamic Range Reduction

To handle non-stationarity of raw values, we differenced and normalized, transforming the raw daily time-series into an absolute daily percent change of the underlying value. We encoded the sign of the percent change as separate feature, e.g. "Direction of Change in Closing Price." We then reduced the dynamic range of each of the transformed time series by applying the log transformation proposed by Tsoi et al [3]:

$$\delta_1 = \frac{(x_{t-1} - x_t)}{x_{t-1}}$$
$$x'_t = \text{sign}(\delta_t) * (\log(|\delta_t|) + 1)$$

3.1.2 Quantization

To further counteract the noisiness of the ensuing time-series, we discretized each variable into a finite number of bins that correspond to percentile ranges of the data. To illustrate this point further, in a setting where 10 bins are applied, the 5th would be populated by values that fall between the 40th and 50th percentile of the specific variables. Our motivation for applying these transformations was to reduce noisy continuous data into discrete levels. Once discretized, time windows within the time series can be thought of as patterns; thus transforming what would otherwise be a regression task, into a pattern recognition problem. This allows recurrent neural networks that are designed primarily to process patterns, to achieve higher generalization. Tsoi et al [3] chose instead to quantize their resulting times series by using a self-organizing map (SOM). We choose not to follow this approach, due to the extraneous complexity in hyper-parameter optimization that this would require.

3.1.3 Selecting Data for Model Input

With all continuous variables quantized, we applied a 6:3:1 split of our dataset into training, validation and testing partitions. We performed this split by partitioning our time series for every company into a finite number of non-overlapping windows of length w. Within a given window, features taken from day 0 to day $w - 1$ served as sequence inputs to our recurrent model that aimed to predict the price direction of the given company at day w. We randomly assigned every window to either the training, testing or validation partitions in order to avoid seasonality biases. In other words, as opposed to classifying 2010 to 2012 as our training period, we randomly selected time windows between 2010-2015, during which every company in that window will serve as a training example. We did the same for the validation and test sets.

3.2 Preprocessing for Correlation Networks

The raw time-series data retrieved from Bloomberg Market Data Services was also preprocessed to enable construction of market networks for use in the structural and analytical graph analysis project segments. Specifically, the data was used to build, for each basic variable and time period, a matrix of variable change correlations between each pair of stocks.

For a given time period (a particular quarter or year), this process began by filtering out variables that had undefined values for more than 20% of trading days. Differencing was then performed to turn the series of raw values for a particular stock and variable into a series of percentage changes between trading days.

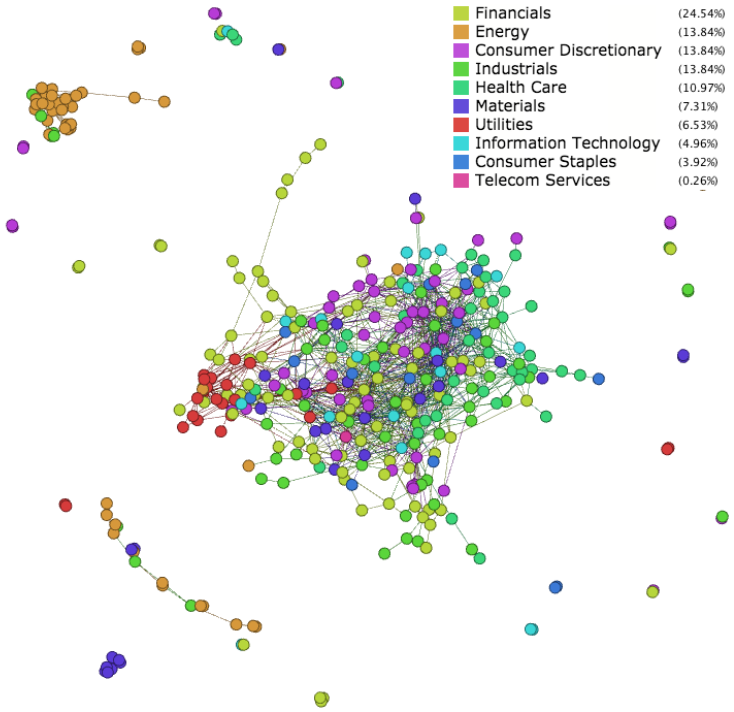

Figure 1: Network built from correlations in closing price above a 0.975 threshold, colored by industry sector.

Variable	$x_{\min}$	α	Log-Likelihood
Market cap	1	2.44	-3.1
News heat	1	2.44	-4.4
Number of news stories	1	2.21	-43.5
Closing price	1	2.44	-8.1
High price	1	2.44	-5.6
Low price	1	1.49	-181.1
Twitter sentiment	1	2.17	-47.4
Volume	1	1.50	-135.3

Table 1: Power law fits for the degree distributions of networks built from the full dataset, thresholded at a 0.9 correlation coefficient.

Then for each remaining variable and for each pair of stocks, we calculated the Pearson correlation between their respective series of daily percentage changes.

Note that stocks were added and removed from the market between the relevant years, 2010-2015. To ensure correlations were not calculated in the case where one or both stocks had extensive missing data, we required at least 30 coincidental trading days for the two stocks in the quarterly series and at least 60 coincidental trading days in the annual series.

4 Methodology: Market Correlation Networks

As previously stated, we split our methodology into three components: structural, analytical, and predictive.

4.1 Structural Component

In the structural component, we use correlation thresholds to restrict our market network edges to stocks that have highly correlated movements in one of our variables. The non-singular connected components of one such thresholded network over the full 2010-2015 period are shown in Figure 1. Notably, there is a dominant connected component with companies from a variety of sectors. This cluster is dominated by the financial sector (28.3% of equities in the component), as also found by Tse et al [1], but there are visible subclusters from the Consumer Discretionary (14.5%), Industrials (14.2%), Health Care (13.1%), and Utilities (7.3%) sectors. The next largest connected components are much more homogeneous and represent the Energy and Materials sectors.

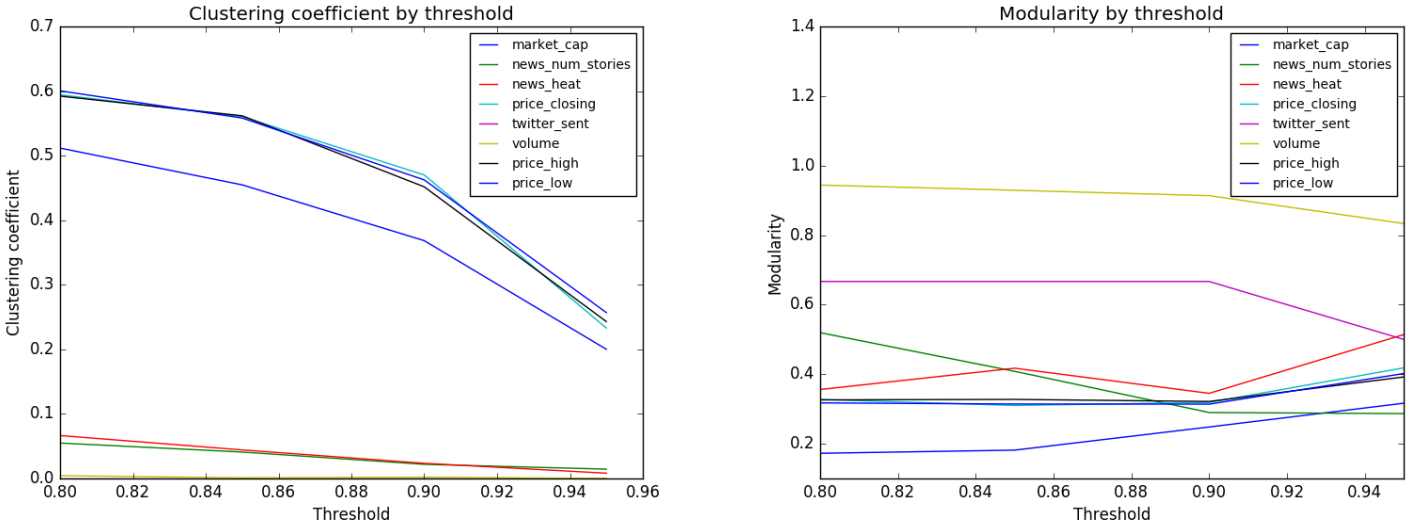

Figure 2: Clustering coefficients and modularity scores for networks built from all variables and thresholds.

A key structural property of thresholded market correlation networks found by previous authors is a power law degree distribution for appropriate correlation thresholds [1][2]. As in Tse et al, we found that correlation thresholds of 0.85 and 0.9 resulted in degree distributions that were well-fit by power laws. The power law fits for each variable are given in Table 1. Except for low price and volume, which had worse power law fits than the other variables, the power law exponents are between 2.17 and 2.44, typical for empirical data. Several variables had very similar degree distributions at high thresholds and thus very similar power law fits. That the degree distribution follows a power law suggests that a few equities are highly correlated (in terms of changes in price, volume, etc.) with the rest of the network, while the majority of stocks are not very correlated with most other stocks. Intuitively, and as found by previous authors, financial-sector equities, and especially funds holding a variety of stocks, usually dominate the tail of the degree distributions. For example, the equities with the 10 highest degrees in a closing price correlation graph over the 2010-2015 period with a threshold of 0.85 are UDR, EQR, ESS, AVG, CPT, EOI, ETG, ETO, ETY, and EVT. With the exception of AVG (security software), these equities are all either investment funds or real estate firms.

We also explored clustering properties of thresholded graphs for the various variables and thresholding levels. Figure 2 shows two clustering measures, clustering coefficient and modularity, for all variables over thresholds between 0.8 and 0.95. Interestingly, Twitter sentiment, trade volume, and news-related variables lead to better modularity scores than strictly price-related variables, so properties of graphs based on these variables may help our eventual model (section 4.3) distinguish between network communities more easily than properties based on price-related variables' graphs. However, they also had much lower overall clustering coefficients than graphs built from price-related variables, which tend to have a large, highly connected component (as we saw in Figure 1).

With these graph-wide structural properties in mind, our goal is to generate stock-specific (and thus node-specific) graph-based features that can be used as inputs to our prediction model (section 4.3). In particular, we would like to give the model the preprocessed variable values (section 3.1), some notions of how central or influential stocks are in our market correlation networks, and the knowledge of which stocks are connected (have an edge remaining after thresholding) in these networks. With this information, we hope the model can capture latent market structure and predict a stock's future price movements based on the recent movements of its neighboring (highly correlated) stocks and the market as a whole. Therefore, we computed both a collection of graph-based features and a list of neighboring nodes for each node (stock).

The graph-features chosen are intended to convey numerical measures of a stock's (node's) centrality, connectivity, or membership in larger structures. After exploring the degree distributions of these networks, we included a node's degree as well as the number of neighbors at 2, 3, and 4 hops as features. For centrality, we computed PageRank, betweenness, and closeness (with less extreme thresholding; see the next section). To capture membership in the dense market core that typically appears in these correlation networks (as shown earlier in Figure 1), we added a feature for the size of a node's weakly connected component. To attempt identification of stocks that bridge market sectors, we added an indicator feature for whether nodes were articulation points. Finally, to measure local clustering, we calculated the number of triads in which a stock's node was a member.

Rank	Equity	PageRank	Sector
1	UTF	0.0003976	Financials
2	BDJ	0.00039759	Financials
3	EOS	0.00039758	Financials
4	FEO	0.00039757	Financials
5	NIE	0.00039757	Financials
6	CII	0.00039746	Financials
7	MGU	0.00039745	Financials
8	AVK	0.0003974	Financials
9	NFJ	0.0003974	Financials
10	FFS	0.0003974	Financials

Rank	Equity	Betweenness	Sector
1	UTF	50.125	Financials
2	PMC	48.685	Health Care
3	FLC	48.388	Financials
4	DPM	48.286	Energy
5	GEL	48.121	Energy
6	CII	48.000	Financials
7	BDJ	47.672	Financials
8	MIL	47.592	Industrials
9	KHI	47.485	Financials
10	WMB	47.355	Energy

Table 2: Equities with top PageRank and betweenness centrality for a 0-thresholded network built from correlations in percent changes in closing price over the entire 2010-2015 period.

Graph-Based Feature	Correlated?	Causal Variables at p=0.01
Degree at hops 1, 2, 3, 4	no	price_closing, price_high, price_low
Number of triad memberships	no	-
Articulation point indicator	no	price_closing, price_low
WCC size	no	price_closing, price_high, price_low
Closeness	no	price_closing, price_high, price_low, volume
PageRank	no	-

Table 3: Correlation and causation testing results between changes in the various graph features and changes in stock price.

4.2 Analytical Component

For the analytical component, we first analyzed the centrality measures of the full graph (considering every positive correlation as an edge, but still discarding missing and negative correlations, following findings in Boginski et al [2]) in order to investigate the companies with the highest centrality. We then performed statistical analysis on various graph features to see if we could detect any correlation or causation with raw stock price changes, to determine if any features had predictive value.

4.2.1 Centrality in Full Graph

Table 2 shows the top 10 equity nodes by PageRank and betweenness for a network based on closing price and all positive correlations. Both centrality measures are dominated by financial companies, which is consistent with previous work by Tse et al [1]. Financial companies likely dominate as their fortunes are linked directly to the performance of many other companies (their investments). Additionally, the performance of financial companies is heavily linked to the performance of the market at large. If stocks in general are rising, then so will the prices of financial companies. Thus their price is positively correlated with a large number of varied stocks. This interconnectivity results in high PageRank and betweenness ratings.

Energy companies are also heavily represented in the betweenness table. Looking at figure 1, we see that energy companies are not very connected with the main connected component of nodes, and instead are very interconnected between themselves in clusters of their own. Thus, some energy companies end up being large fish in a small pond and end up with high betweenness.

4.2.2 Correlation and Causality Between Graph Features and Price Movements

To analytically determine whether the graph-based features previously detailed had possible value for our prediction task, we performed statistical analysis on the time series of graph features with regards to stock movements. Specifically, we took the quarterly graph-based features for each variable used by the model (closing price, high price, low price, and volume) and applied differencing to find quarterly percentage changes. We constructed a similar series of quarterly percentage changes by differencing raw stock prices on the first and last trading days of each quarter. We concatenated these series across all stocks to gather all pairs of graph and price percentage changes. We then computed the correlation coefficient of each variable's series. In addition, we applied the Granger causality test (lagged F-tests) to these series for lags of 1 and 2 quarters, making sure to adjust the concatenated format so that graph features and prices of different stocks were never compared. The results of statistical testing are summarized in Table 3.

Feature Class	Example Feature	Example Value
Intrinsic Features (A)	Quantized Percent Change in Closing Price	Level 1 Positive
Graph Features (B)	Closeness Score of Stock Over Prior Period	2.5
Network Locality Features (C)	Mean Closing Price Levels of Neighboring Stocks	25% Level 2

Table 4: Summary of feature classes.

$$
\begin{aligned}
i^{(t)} &= \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)}) && \text{(Input gate)} \\
f^{(t)} &= \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)}) && \text{(Forget gate)} \\
o^{(t)} &= \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)}) && \text{(Output/Exposure gate)} \\
\tilde{c}^{(t)} &= \tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)}) && \text{(New memory cell)} \\
c^{(t)} &= f^{(t)} \circ \tilde{c}^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{(Final memory cell)} \\
h^{(t)} &= o^{(t)} \circ \tanh(c^{(t)})
\end{aligned}
$$

Figure 3: Long short-term memory (LSTM) equations.

We found no strong correlations between the quarterly change in graph features and price changes; the magnitude of the largest correlation coefficient was just 0.177. However, for significance level p=0.01 and after applying the Bonferroni correction to account for testing multiple feature/variable pairs, the lagged F-tests found causal relationships between some of the graph features (primarily degree, closeness, and size of a node's weakly connected component) and variables (primarily price-related variables) for lags of 1 quarter, 2 quarters, or both. It is worth noting that quarterly data is coarse and that causation without correlation may merit deeper investigation. However, our tests provide some quantitative basis for including a subset of the graph features as model inputs.

4.3 Predictive Framework

4.3.1 Featurization

We modeled the daily directional change of the 2800 NASDAQ-listed stocks between 2010 and 2015. Our dataset is composed of 86,427, 39,285, and 13,095 training, validation and testing examples respectively. We employ the three classes of features show in Table 4. Class A features are derived from stock-specific daily data points (e.g. closing price) preprocessed in the fashion outlined in section 3.1. Class B features are derived from graph metrics of networks computed over the quarter prior to our trade execution date. We only incorporated metrics that exhibit causality into our Class B features (Table 4). Class C features are the average quantized Class A feature of all stocks that exhibited a Pearson correlation of 90% to the closing price of the stock of interest, over the prior quarter. We incorporate Class C features to examine whether a company's network locality provide any additional predictive information.

We develop four Recurrent Neural Network (RNN) models that incorporate different combinations of our features classes (Table 3). RNNs are well-positioned for prediction over financial time series due to their capacity to internalize and process arbitrary sequences of inputs. However, RNNs tend to forget long-term information. Given weeks of data, a vanilla RNN would base its prediction mostly on the last few days.

4.3.2 LSTM

To overcome this constraint, we employ the long short-term memory (LSTM) technique.[5] Under LSTM; gates are applied at each activation unit to preserve long-range memory across our input sequence. The five mathematical steps preceding the final hidden unit calculation that make up the LSTM unit are shown in Figure 3.

First, the current input and previous hidden state are linearly combined and non-linearly transformed to create the New memory cell. Two gates, the Input gate and Forget gate, are then calculated; the first governs the importance of the current input and the second indicates how much of the previous state must be remembered or forgotten. The Final memory cell is generated by filtering the New memory cell from the current state through the Input gate and the New memory cell from the previous state through the Forget gate. Lastly, the final memory cell is passed through an Output/Exposure Gate to determine which information to preserve in that unit's hidden state.

4.3.3 Tuning Hyperparameters

With regards to our input features, we optimized for the sequence length and the number of discretization levels of Class A features. The optimal sequence length was determined by evaluating the precision of a naïve model that bases its prediction on whether it has seen more ups or downs over the sequence window. For instance, if in the last 5 days this model had seen more positive trading days than negative ones, it would predict that the next day would be a positive day. We found that predictions that employed a longer window performed better than those that did not. We settled on the optimal window size of 25 days that achieved a precision score of 36%. The fact that this time frame corresponds exactly to one month (25 trading days) indicates that predictive patterns in stock prices occur over a monthly time scale.

The optimal bin size was determined by evaluating the performance of a shallow 200-d hidden unit LSTM when supplied with differently discretized Class A features. We found there to be no statistically significant difference in performance based on discretization if four or more bins are used. This result indicates the magnitude of stock price changes over a past window is of little consequence in comparison to the directions of change. We proceeded with 10 bins, due to its natural interpretation as decile percentiles.

With regards to model hyper parameters, we employed a greedy approach to optimization: Starting with our default 200-d hidden unit shallow LSTM that utilizes only Class A features, we successively optimized for depth, hidden unit dimension, learning rate, momentum, regularization, mini batches, and training epochs. Our optimal parameters were as follows: depth 2, hidden dimension size 300, learning rate 1e-4, Nestrov momentum 0.9, regularization 1e-5, mini batches 100, and training epochs 3. We intuit that the optimal depth of 2 allows the model to capture patterns that occur at different timescales [6].

5 Results

Metric	LSTM-A/B	LSTM-A	LSTM-A/C	LSTM-A/B/C	Random	Contrarian	Optimistic
F1-Score Tot.	**55.95**	53.91	48.89	51.67	50.28	70.26	62.87
F1-Score Pos.	**51.87**	49.42	49.30	51.09	46.57	0.00	45.85
F1-Score Neg.	**58.20**	55.62	42.38	48.36	52.20	54.15	0.00
Precision Tot.	**56.00**	53.96	48.54	51.40	50.28	54.15	45.85
Precision Pos.	**51.92**	49.74	45.24	47.54	46.14	0.00	45.85
Precision Neg.	**59.78**	57.91	53.14	56.39	54.45	54.15	0.00
Recall Tot.	**55.90**	53.86	49.24	51.94	50.28	100.00	100.00
Recall Pos.	54.35	52.97	57.86	**58.43**	50.46	-	100.00
Recall Neg.	**57.44**	54.74	40.62	45.44	50.14	100.00	-

Table 5: Development set results for the LSTM model with various combination of feature classes, as well as the baseline random, contrarian, and optimistic models. Metrics are given both for all stocks (total) and broken down by positive and negative movements.

6 Discussion

6.1 Graph Features

Table 5 presents the development results of the four LSTM variants trained using different combination of feature classes. We observed that LSTM-A/B (a network that uses Class A and Class B features) all other LSTM variants in overall precision, recall and F1 scores. We observe that the graph features chosen (see Table 5) due to their passing of the Granger causality test, provided a 2.94% increase in precision over a model that predicts price-movements based on stock intrinsic features alone. Graph features add new information about the market dynamics that is not captured by stock intrinsic features. Overall we observe that the LSTM-A/B outperforms a randomly predicting model in precision by 5.72%. We observe as well that 54.15% of trading instance in our validation set were negative. On a directional basis, we note that LSTM-A/B features provide a 5.63%, and 6.07% increase in precision over a contrarian strategy (short only) and an optimistic strategy (buy and hold) respectively.

6.2 Network Locality Features

Our results further reveal that Class C features (average intrinsic features of a company's graphical neighbors) are detrimental to performance. For LSTM-A/C we observe a precision score that is 1.74% lower than of

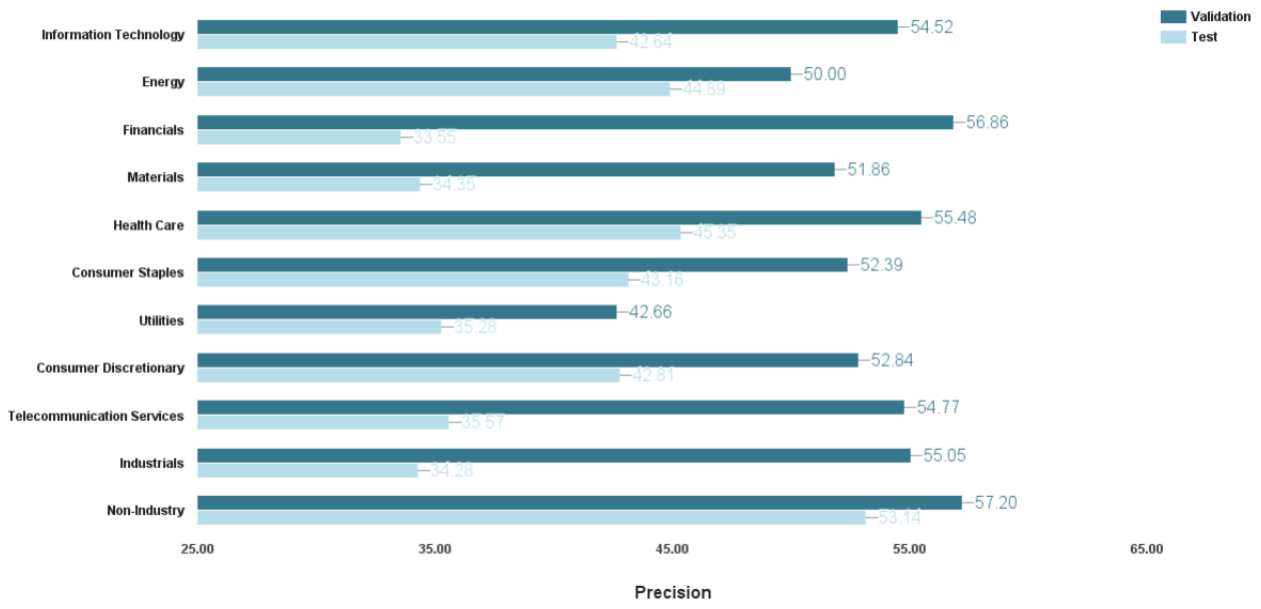

Figure 4: Development (validation) and test set precision results, broken down by sector.

the random strategy, and 7.46% lower than that of LSTM-A/B. By definition, Class C features are highly correlated to Class A features. We recognize that this high correlation may have led to over-fitting due to the extraneous degrees of freedom that are supplied to the network. The fact that adding Class C features to LSTM-A/B, i.e. LSTM-A/B/C, mitigated the loss in generalization, provides further evidence of the utility of graph features.

6.3 Performance By Sector

On a per-sector basis (see Figure 4) we observe that non-industry stocks (e.g. index stocks such as Exchange Traded Funds) and Financial Stocks are most amenable to pattern based prediction in our validation set. We observe, however, that precision drops across all sectors in our test-set. This drop in performance is a consequence of the fact that our model was trained, validated and tested in three different market environments. 50% of the trading instances were positive in our training set; 45% were positive in our validation set; and 58% we positive in our testing set. We recognize that our models are sensitive to the market environments in which they are trained. We attempted to counteract this bias by randomly assigning time windows to partitions. A five year period, however, was not sufficient to eliminate the bias completely. Nonetheless we observe high precision (53.14%) relative to a a random strategy ($\sim$ 50%) for Non-industry type stocks. We posit that these stocks are more amenable to pattern based prediction, because they track market indices and are consequently less susceptible to company-specific idiosyncrasies (i.e. catalytic events) that might offset pattern based predictions.

7 Future Work

Future iterations on this work should first try to improve model generalization error and reduce overfitting. Using training, development, and testing data of roughly the same general market trend (similar percent of stocks with upwards vs. downwards movements over a given period of time) would allow for more accurate measures of model performance. In addition, adding more training data, reducing the size of the feature space through feature selection or similar techniques, and tuning the model hyperparameters would help tackle overfitting.

Additional computing power could be used to work with network-derived data at much more granular periods of time, such as weekly or intraday data, as opposed to the quarterly splits used in this paper.

Another avenue for further improvement involves the compilation of more centrality/connectedness features, including those not specific to the stock but rather to the whole network (such as graph diameter). Additionally, many centrality features were disqualified by being inapplicable to a non-connected graph but could be adopted for use. Considering that the graph edges often have vastly different correlation values, the adoption of centrality measures incorporating weighted edges may also be beneficial.

8 Team Contributions

David: obtaining and preprocessing data, model setup, model optimization
Keith: structural graph properties, computing graph features for model
Alp: analytical graph properties, statistical feature testing

References

[1] Tse, C. K., Liu, J., and Lau, F. C.M., 2010. A network perspective of the stock market. Journal of Empirical Finance 17 (2010), p. 659-667.

[2] Boginski, V., Butenko, S., and Pardalos, P. M., 2004. Statistical analysis of financial networks. Computational Statistics & Data Analysis 48 (2005), p. 431-443.

[3] Giles, C., Lawrence, S., and Tsoi, A., 2001. Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference. Machine Learning, Volume 44, Number $\frac{1}{2}$, July/August, pp. 161-183.

[4] Timmermann, A., and Granger, C. W.J., 2004. Efficient market hypothesis and forecasting. International Journal of Forecasting, Volume 20, Issue 1, January-March 2004, p. 15-27.

[5] Hochreiter, S., and Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computation 9, p. 1735-1780.

[6] Hermans, M., and Schrauwen, B., 2013. Training and Analyzing Deep Recurrent Neural Networks. NIPS 2013. http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2013_5166.pdf