

Event-Driven Asset Pricing Prediction

Daniel Huang

Department of Economics, Stanford University

511 Lasuen Mall, Stanford, California 94309

Dhuang7@stanford.edu

Abstract

The ability to predict future asset pricing will directly disprove the Efficient Market Hypothesis, a central tenet to the operations of financial markets. I attempted to use newspaper headlines to predict directional movement in the Dow Jones Industrial Average (DJIA), which tracks the performance of large American companies. I used both traditional natural language processing techniques with machine learning as well as various LSTM approaches, including stacked LSTM, for this classification problem. The LSTM model outperformed all other techniques, although further work is needed to generalize this result to all asset classes.

1 Introduction

Asset pricing plays an essential role in portfolio allocation, and is a central concern for market participants, from large hedge funds to individual retail investors. For market players, asset pricing plays a crucial role in managing portfolio risk, in the form of varying the allocation of capital among different industries and asset classes. Because asset pricing is intrinsically linked to profits (“alpha generation”), the ability to predict future asset pricing is highly desirable for investors.

Despite the importance of predictive asset pricing for market players, classic economic theory dictates that it is impossible. According to the Efficient Market Hypothesis, a classic theory derived by Eugene Fama and Kenneth French, it should be impossible to consistently achieve outsized returns on a risk-adjusted basis [1]. This conclusion asserts that the present price of an asset has taken into consideration the entire realm of publicly-available information, and that asset pricing in financial markets vary due to the release of novel, material information to the public. Therefore, from a classical economics standpoint, it should not be possible for any mechanism to have predictive power in the domain of asset pricing.

In industry, however, the reality has been different. According to the mean reversion principle, the “fair value” market price of an asset can be determined mathematically, using models that incorporate all publicly-available information into the overall valuation, with the implicit assumption that the market value will eventually revert to the fair value determined by the model [2]. However, as traditional sources of information, such as company earnings reports, are central to virtually every firm’s fundamental investment strategy, the information gleaned from these reports does not give any investor a competitive edge, and we see a manifestation of the Efficient Market Hypothesis. Therefore, some firms, known as quantitative research firms, are turning to building machine learning and deep learning models to mine datasets for economically predictive insights that are unknown to the general public.

In order to explore the potential of machine learning and deep learning on asset price prediction, I will use natural language processing algorithms on news headlines, and evaluate the algorithmic efficacy in mining economically predictive insights by predicting interday asset price movement. Specifically, I will compare traditional machine learning NLP techniques (my baseline performance for evaluation) with using deep learning in NLP. I hypothesize that deep learning techniques will outperform due to scaling, because neural networks’ performance tends to increase (rather than plateau) as the amount of data increases, and news headlines are readily available.

The input to our algorithm is a set of news headlines from Yahoo Finance and Reddit News, over a period of eight years, from fiscal year 2008 to fiscal year 2016 (inclusive). We then use a baseline implementation (traditional natural language processing with machine learning algorithms such as random forest, SVC, etc) and a deep learning implementation (RNN, LSTM) to predict the direction of price movement (up or down) of the Dow Jones Industrial Average (DJIA) for a certain date, which tracks 30, large, publicly-owned companies in the United States. Therefore, this is a binary classification task.

2 Related Work

Asset price prediction has been a widely-studied topic in the intersection of financial markets and artificial intelligence. Researchers have used both machine learning and deep learning approaches on variations of the asset pricing prediction problem, which all fundamentally relate to testing the soundness of the Efficient Market Hypothesis. From a purely machine learning standpoint, Zhou et al. (2015) used sentiment analysis combined with a support vector machine approach on Chinese social media microblog posts in order to key metrics of the stock, including the opening and closing indices. Their model outperformed traditional baseline solutions that do not account for sentiment analysis that looks for emotions such as disgust, joy, sadness, and fear [6]. Another machine learning approach (with a different dataset) can be found in a well-known

paper by Lee, Surdeanu, MacCartney, and Jurafsky, who showed that information from a company's publicly-available financial reports contains predictive signaling for its stock price [3]. They used a random forest classifier for prediction, and were able to conclude that there was an impact between company-reported news and the company's stock price. Another group of researchers analyzed Twitter data in conjunction with Support Vector Machines and found a correlation between tweet sentiment and stock prices [4]. These studies have focused on using traditional machine learning techniques, whose prediction accuracy does not scale when the amount of data increases exponentially.

More recently, there has been interest in applying deep learning to this problem. Researchers Vargas, de Lima, and Evsukoff have used CNNs and RNNs to predict future trends of stock prices, and have shown some improvement in stock market forecasting [5]. In particular, they have focused on using CNNs to catch semantics from texts and RNNs to catching context information and modeling complex temporal characteristics. However, they have not explored the variations in RNN architecture that can lead to improvements in prediction. Similarly, Akita et al. have used a LSTM model to model temporal effects of past events on opening prices by converting newspaper articles into their distributed representations via Paragraph Vector. This approach preserves the fine-grain price and textual details, and has been tested on the Tokyo Stock Exchange with moderate results.

3 Dataset and Features

To obtain the dataset, I started with a pre-formatted dataset from Aaron Jiahao Sun (University of Oxford) [8]. The dataset spans the years 2000 to 2016, and headlines are given by day. This range should capture some interesting market-moving headlines, such as the 2001 9/11 attacks and 2008 financial crisis.

Text features: The text features consist of the news headlines published on a certain day. By the Efficient Market Hypothesis, we will assume that the stock price movement is affected by new information from the previous day, so we will use the previous day's headlines to predict the interday stock price movement between the previous day and the current day. In order to augment this dataset further, I pulled headlines from Reuters and BBC news (financial news section). Thus, I ended with a dataset of the format below (2845 days, 50 headlines each, so **147250 features** in total).

Price labels: We focus on the price of the Dow Jones Industrial Average (DJIA), a stock market index reflecting the performance of the 30 large, publicly-traded companies in the United States [6]. If the price increases from the previous day, then the label will be a '1'; otherwise it will be a '0'. Since we are not focusing on predicting the exact asset price and instead only intend to predict the direction of the change, we pursue a classification problem rather than a regression problem. For our purposes, we care more about the direction of the asset price movement rather than the magnitude, because knowing the direction is an actionable insight. For instance, if the asset price is predicted to increase, then the action would be to purchase the asset.

Dataset format:

Date	Label: DJIA price movement	Features: news headlines			
2008-06-08	1	Headline 1	Headline 2	...	Headline 50
2008-06-09	0	Headline 1	Headline 2	...	Headline 50
2008-06-10	0	Headline 1	Headline 2	...	Headline 50
...

One example:

2013-11-09	1	German support for small business has kept its economy thriving as the rest of Europe languishes in recession	Once touted as an economic miracle, faltering India cannot provide jobs for millions of university graduates.	...
------------	---	---	---	-----

I stripped the textual information of any non-alphanumeric characters (e.g. punctuation marks) during preprocessing. I used the CountVectorizer from the sklearn library to convert the headlines into a matrix of token counts (explained in more detail later). The time series is discretized by day.

4 Methods

My baseline implementation will be using traditional NLP along with machine learning classification algorithms. Then, I will do a deep learning implementation with the same dataset and preprocessing methods, and compare the two results.

Baseline Implementation

Train/test split: The train/test split is conducted randomly, and we use a 95/5 split. Since this is time-series data, we must be aware of look-ahead bias, or using future days' data to predict today's price movement [15]. Thus, we split the dataset sequentially based on time, with the train set encompassing the earliest 95% of the dataset, and the test set encompassing the remaining 5%. Therefore, the train set comprises all days on and before 2015/01/01, and the test set comprises all days afterwards.

Text preprocessing: First, we remove punctuation marks and convert all letters to lowercase. Then, we join all headlines in a single day together, so that we are left with the label column and the combined headlines column. We also correct for the imbalance problem by ensuring that a roughly equal amount of positive and negative examples exist in each set.

Vectorizing text: Before applying machine learning models, it is first necessary to transform the text features into numerical feature vectors. We use the bags of words representation, described below [18]. First, we build a dictionary of all words appearing in the training set by assigning a fixed integer id to each unique word. Then, for each example i , we count the number of occurrences of each word w and store it in $X[i, j]$, where j is the index of w in the dictionary. We use CountVectorizer in sklearn, which is an implementation of this process.

We also vary the counts for different N-grams of words, which means N adjacent words are stored in the dictionary, instead of one single word. We vary the range from 1 to 5 inclusive, and pick a small range because we do not expect long-distance dependencies in headline language.

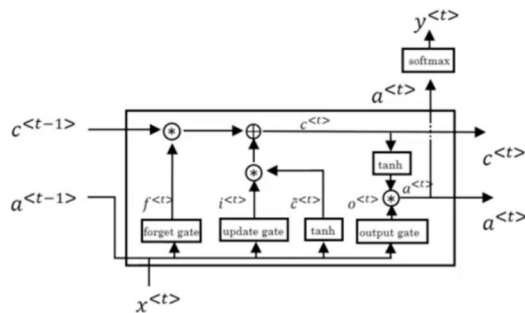
Model selection: To implement the baseline models, I used the scikit-learn library in Python. Specifically, the models I tried were Logistic Regression, Random Forest Classifier, Support Vector Classifier, Linear Support Vector Classifier, and K-Neighbors Classifier. These models have been used by Sathvik Raju on stock index data and Josephine Ho on Twitter data [16] [17]. As a future extension to this project, I would like to implement grid search to tune model parameters in order to get a better estimate of the performance of these baseline models.

Deep Learning Implementation

For the deep learning implementation, I tried an LSTM approach with gridsearch as well as a stacked LSTM.

Examining one unit, we use $a^{<t-1>}$ (activation/output cell value for the previous time step) and $x^{<t>}$ to compute all the gates (forget gate, update gate, tanh, and output gate). In combination with the previous memory cell value $c^{<t-1>}$, you can obtain the current memory cell value $c^{<t>}$.

Figure 1: Unit used in LSTM, described below.



The reason why the LSTM is good at memorizing values is because you can link up many units. In the diagram (figure 2), you can see how $c^{<0>}$ is able to be passed to $c^{<3>}$, avoiding any mutations for many timestamps. At every step, we consider a candidate $\tilde{c}^{<t>}$ to replace the current $c^{<t>}$. For this, we have a tanh input gate layer to decide what values to update:

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

Then, we use the update gate (sigmoid function outputting between 0 and 1) to create an update to the state. The forget gate controls which information to throw away or forget at this step. Lastly, we decide what to output by

filtering out the irrelevant information from the cell state. This information is summarized in figure 1.

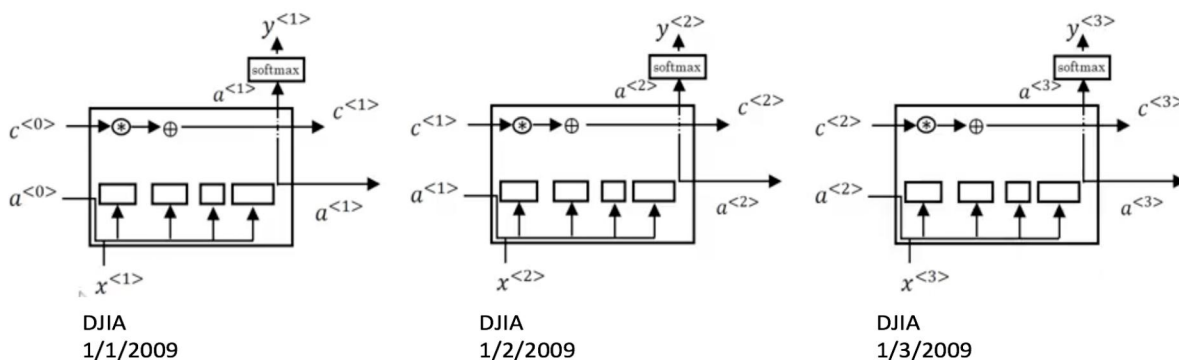


Figure 2: Linked together units in LSTM

(Image credits: Andrew Ng/Coursera: "Sequence Models" course, Week 1 lecture)

Stacked LSTM implementation:

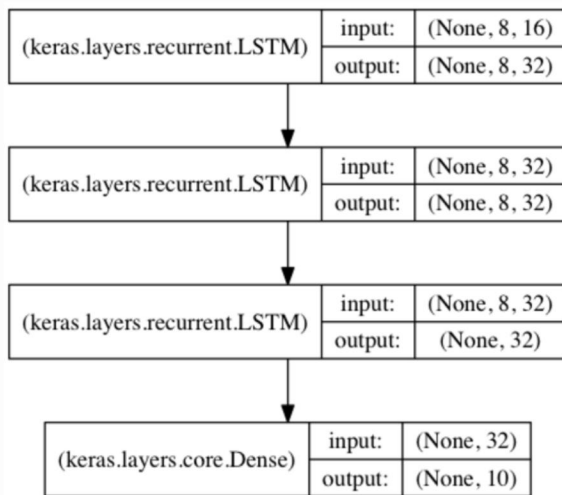


Figure 3: stacked LSTMs [19]

This model is similar to the implementation of stacking within random forest algorithms, except we stack LSTM layers on top of each other. This architecture allows the model to learn high-level temporal representations. Even though it is not theoretically clear why this deeper architecture may produce better prediction results, it has been observed empirically that deep RNNs work better than shallow RNNs on some tasks, such as machine-translation performance [9]. Since layered RNN architectures may report better results, I will also try stacking LSTMs.

The structure is displayed in figure 3. The first LSTM returns the full output sequence and feeds it to the second LSTM. The second LSTM likewise also returns its full output sequence. However, the last LSTM only returns the last step in its output sequence. Therefore, the input sequence is converted into a single vector, and the temporal dimension is dropped from the output.

5 Results and Discussion

To be able to compare the results from the different implementations, I use the same dataset, which means using the same asset (DJIA stock market index) over the same time range. Therefore, I can compare the prediction accuracy of the various models.

To evaluate the performance of the models, I will report the F1-score and the accuracy score.

The F1-score is the harmonic mean of precision and recall, with a higher value indicating a better performing model.

$$F1 = 2 \times \frac{Precision \cdot Recall}{Precision + Recall}$$

The accuracy score indicates the fraction of correct predictions out of all predictions made by the model.

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \quad (TP = \text{True Positive}, TN = \text{True Negative}, FN = \text{False Negative}, FP = \text{False Positive})$$

Baseline Implementation

The following results are achieved using the default parameter settings in sklearn.

Model	F1-Score (0, 1)	Accuracy Score
Logistic Regression	0.17, 0.82	0.45
Random Forest Clf	0.49, 0.52	0.52
SVC	0.00, 0.67	0.50
Linear SVC	0.15, 0.85	0.54
KNN	0.18, 0.70	0.55

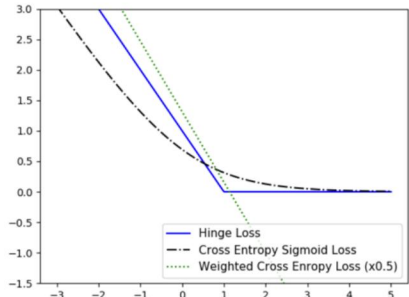
* Varying the n_grams does not produce any meaningful change in accuracy score, so I have omitted them from the table and only reported with n_grams (1, 1).

Without correcting for the imbalance issue (ensuring that there is the same amount of positive and negative observations), we would notice that the F1 scores and accuracy scores for models Logistic Regression, RF, and Linear SVC are high, north of 0.8. After correction, we see that all of the accuracy scores are within the bounds [0.45, 0.55]. This suggests that the prediction accuracy is near a random coin flip, which confirms the Efficient Market Hypothesis. Notably, we see that the KNN model performs the best, with an accuracy of 0.55.

Deep Learning Implementation

For the deep learning implementation, I carried out a GridSearch to determine the best (hyper)parameters for my model:

Hyperparameter	Possible Values	Interpretation
MAX_TOP_WORDS	5000, 10000, 20000	Tokenizer's max number of words to keep (based on word frequency). Higher value produces bigger vocabulary.

LSTM_ACTIVATION	tanh, sigmoid	We usually use tanh because it converges faster than sigmoid (so is less computationally expensive for gradient computation), and can handle the vanishing gradient problem better than sigmoid [10].
DENSE_ACTIVATION	sigmoid, softmax, none (linear)	See above
LOSS	binary_crossentropy, categorical_hinge	 <p>Cross entropy measures the divergence between two probability distributions, whereas the hinge loss is used for maximum-margin classification [11]. In the figure to the left, we see that hinge loss decreases much slower than cross-entropy loss [12].</p>
OPTIMIZER	sgd, adam, adadelta	SGD is a stochastic approximation of gradient descent optimization. There are several variations, including Adam optimization, which includes adaptive gradient algorithm (good for sparse gradients) and RMSprop [13]. Adadelta is another extension that does not require learning rate tuning and is robust to noisy gradient information [14].
...SGD Parameters:		
.....LEARNING_RATE	0.01, 0.05, 0.1, 0.2	The learning rate controls how much the neural network weights are adjusted, with respect to the loss gradient. Too high of a learning rate causes oscillations, while too low of a learning rate makes for slow convergence and possibly getting stuck at a saddle point.
.....MOMENTUM	0.0, 0.1, 0.5, 0.9	Momentum accelerates gradient descent for faster convergence.
.....DECAY	0.0, 0.00001, 0.0001	Weight decay introduces regularization into the neural network's loss
NB_EPOCH	1, 2, 5, 10, 20	In general, the models improve with more epochs of training, until performance plateaus

In the end, the model with the highest prediction accuracy of **0.58** had the following hyperparameters:

MAX_TOP_WORDS = 5000, LSTM_ACTIVATION = tanh, DENSE_ACTIVATION = sigmoid, OPTIMIZER = adam, NB_EPOCH = 5

The stacked LSTM with default parameters (below) had the highest prediction accuracy of **0.56**.

MAX_TOP_WORDS = 10000, LSTM_ACTIVATION = tanh, DENSE_ACTIVATION = sigmoid, OPTIMIZER = sgd, NB_EPOCH = 5

Example of incorrect prediction:

Around December 16, 2015, the Federal Reserve announced that it was raising interest rates. This usually causes a decline in the stock market, and indeed the DJIA moved downwards due to the announcement. However, the neural network predicted an increase in rates. Why?

Example headline "Fed **Raises** **Key** Interest Rate **For First Time** Since 2006" (positive words highlighted in **green**)

The effect on the stock market that the previous times that the Federal Reserve raised rates had were not captured in the dataset, so the NN did not learn the relationship between the Federal Reserve raising rates and the direction of the DJIA.

6 Conclusion/Future Work

In this project, we tried to predict the future price movement of the DJIA using news headlines. We compare traditional models (logistic regression, random forest classifier, svc, linear svc, and knn) as well as deep learning models (LSTM, stacked LSTM) with gridsearch. The best-performing baseline model (traditional machine learning model) was knn with 0.55 prediction accuracy, while the best performing deep learning model (LSTM with tuned hyperparameters) had prediction accuracy of 0.58. A performance accuracy of 0.50 would suggest violation of the Efficient Market Hypothesis, but more work must be done to determine if this deviation of 0.08 in prediction accuracy by the LSTM is statistically significant. This could be done via hypothesis testing, using a larger sample of more diverse news headlines, and using more temporally historical headlines to predict a day's asset price movement. Another direction of future work would be to extend this analysis to other asset classes. This would be interesting since some assets (e.g. short-term bonds, foreign exchange securities) are likely more sensitive to breaking news headlines than other assets (e.g. long-term bonds, blue chip stocks). A further step would be to convert this from a classification problem into a regression problem (estimating the asset price instead of predicting the direction). Therefore, it remains to be seen whether deep learning methods can disprove the Efficient Market Hypothesis.

Github: <https://github.com/danielhuang74/CS230-Project>

7 Contributions and Acknowledgements

Daniel Huang completed the project.

A special thank you for the continuous support from Teaching Assistant Sagar Honnunar. In particular, he suggested many ways to tackle problems that I encountered, including expanding the dataset to avoid overfitting, defining test metrics, suggesting using walk-forward cross validation, helping me overcome the class imbalance issue, and helping to guide me on writing the final report.

References

- [1] Fama, E. F., & French, K. R. (2004). The capital asset pricing model: Theory and evidence. *Journal of economic perspectives*, 18(3), 25-46.
- [2] Poterba, J. M., & Summers, L. H. (1988). Mean reversion in stock prices: Evidence and implications. *Journal of financial economics*, 22(1), 27-59.
- [3] Lee, H., Surdeanu, M., MacCartney, B., & Jurafsky, D. (2014, May). On the Importance of Text Analysis for Stock Price Prediction. In *LREC* (pp. 1170-1175).
- [4] Kordonis, J., Symeonidis, S., & Arampatzis, A. (2016, November). Stock price forecasting via sentiment analysis on Twitter. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics* (p. 36). ACM.
- [5] Vargas, M. R., De Lima, B. S., & Evsukoff, A. G. (2017, June). Deep learning for stock market prediction from financial news articles. In *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)* (pp. 60-65). IEEE.
- [6] Zhou, Z., Zhao, J., & Xu, K. (2016, November). Can online emotions predict the stock market in china?. In *International Conference on Web Information Systems Engineering* (pp. 328-342). Springer, Cham.
- [7] Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016, June). Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)* (pp. 1-6). IEEE.
- [8] Sun, A. (2016, August 25). Daily News for Stock Market Prediction. Retrieved from https://www.kaggle.com/aaron7sun/stocknews/version/1#Combined_News_DJIA.csv
- [9] Goldberg, Yoav. "A Primer on Neural Network Models for Natural Language Processing." *J. Artif. Intell. Res.(JAIR)* 57 (2016): 345-420. https://scholar.google.com/scholar?cluster=3704132192758179278&hl=en&as_sdt=0,5 ;
<http://u.cs.biu.ac.il/~yogo/nnlp.pdf>
- [10] Liu, Angli. (2017, Oct 13). Loss Functions in Neural Networks. Retrieved from <https://www.quora.com/In-an-LSTM-unit-what-is-the-reason-behind-the-use-of-a-tanh-activation>
- [11] Hao, Zhu. (2017, June 06). Loss Functions in Neural Networks. Retrieved from https://isaacchanghau.github.io/post/loss_functions/
- [12] Lulema, A. (2018, January 18). Classification Loss Functions (Part II). Retrieved from <https://alexisalulema.com/2017/12/15/classification-loss-functions-part-ii/>
- [13] Brownlee, J. (2018, December 12). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Retrieved from <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>
- [14] Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.
- [15] Corporate Finance Institute. Look-Ahead Bias - Definition and Practical Example. Retrieved from <https://corporatefinanceinstitute.com/resources/knowledge/finance/look-ahead-bias/>
- [16] Raju, S. (2017, June 26). SATHVIKRAJU/Stock-Market-Prediction-using-Natural-Language-Processing. Retrieved from <https://github.com/SATHVIKRAJU/Stock-Market-Prediction-using-Natural-Language-Processing>
- [17] Inemho, J. (2018, January 31). Josephinemho/Analyzing_Unstructured_Data_for_Finance. Retrieved from https://github.com/josephinemho/Analyzing_Unstructured_Data_for_Finance
- [18] Sklearn. (n.d.). Working With Text Data. Retrieved from https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
- [19] Keras. (n.d.). Getting started with the Keras Sequential model. Retrieved from <https://keras.io/getting-started/sequential-model-guide/>