HIGH SCHOOL EDITION
Journal of Student Research

Volume 10 Issue 4 (2021)

Comparing Different Data Types for Predicting the Apple Stock

Amanda Buguaidi[1] and Dina Ellsworth[#]

[1]High Technology High School, Middletown, NJ, USA
[#]Advisor

ABSTRACT

Stock markets are at the heart of market economies, and stock market prediction is a hot topic in research. There are many methods of prediction, and they often use different types of data. The two main types are technical and fundamental data. This paper explores company-specific prediction through predicting the direction of the Apple stock. Three neural networks that use different input data are compared. The Tech model uses technical data, the Comp model uses numerical fundamental data with an emphasis on company data, and the Text model uses textual news data. Several metrics, including ones useful for preventing bias when dealing with imbalanced data, were used to compare the models. The Comp and Text models showed prediction bias, and two more models, W-Comp and W-Text, were trained to attempt to mitigate that bias. The Comp model performed the best out of the original models, and W-Comp also showed good performance. W-Text performed the best out of all the models. This suggests that numerical fundamental data is useful in predicting the market, and textual data also has potential. Future research could be used to improve the performance of all the models and more thoroughly compare the types of data used.

Introduction

Financial markets are a vital part of market economies, and the state of the market can greatly affect daily life. Developing a method to predict the market would allow company managers and investors to make timely capital allocation adjustments. It might help corporate executives and policy makers anticipate stock market crashes, which have been seen to have widespread and harmful effects. Studying which factors are the most predictive of the stock market can shed insight into the complicated fluctuations of the market and the economy.

When attempting to predict the stock market, previous studies have used three approaches. The first approach, technical analysis, uses a stock's previous market data to predict future behavior. The second, fundamental analysis, uses other information such as company data or political circumstances. The third approach uses a combination of technical and fundamental data [1]. The first approach has been shown to have good performance [2]–[4]. However, there has been gathering support for fundamental analysis, particularly using news data, which has been found to have significant predictive power [1], [5]–[10], [11]–[13]. Studies have also found that the third approach combining both qualitative and quantitative data is perhaps the most effective [1], [10], [13], [14].

Past research has shown several effective practices when constructing models. Neural networks have been found to be among the most effective methods of predicting the stock market [1]–[5], [10], [13], [15]. Long short-term memory (LSTM) networks are a particular neural network architecture that has yielded good results [1], [10], [15]. When working with news data, [5] has shown that using just news titles results in better performance than using news content, because the quality of information is more important than the quantity. There is also support for using more sophisticated text representations for news data rather than simple representations such as bag-of-words, where only the frequency of words in the text is used [1], [5], [10], [11], [14].

In past studies, fundamental analysis is often performed using news or other textual data. There is little literature performing fundamental analysis using numerical data. There has also been limited research in company-specific prediction. There have been a few studies that predict specific stocks, but the companies were not the main focus in these papers [1], [2], [7], [11], [16]. It was often the case that the stocks were simply chosen for prediction without tailoring the model to the companies. Vu et al. [11] did use company-specific news in addition to the stock prices of previous days for stock prediction, but the data used was still rather limited. Company-specific stock prediction allows for the usage of a wide variety of data related to the company being predicted. This data could be combined with external data to help prevent isolation, which was noted to be important by [17]. Predicting the stocks of specific companies would also allow investors to analyze individual stocks to determine if they should invest in them.

In this paper, the direction of the stock of a specific company, Apple, is predicted. Three neural networks with LSTM architecture are constructed to test three different types of input data. The data types tested are technical data, numerical fundamental data with an emphasis on company data, and textual data in the form of news titles and descriptions. The models are compared to determine the effects of different input data on model performance. A word2vec model is used to generate word representations for the news data.

Model Design

Data Collection

Daily data was collected from 1,257 trading days between January 2, 2015 and December 30, 2019. The data consisted of historical stock data, technical indicators, company data, and news data.

Historical stock data were collected from Yahoo Finance (https://finance.yahoo.com). The daily data consisted of the Apple stock (AAPL) trading volume, its adjusted closing prices, the adjusted closing prices for the S&P 500 stock market index, and the adjusted closing prices for the Qualcomm (QCOM) stock. The percent changes in these daily values were calculated using Equation 1, where P is the percent change and x(t) is the value for the current day. The S&P 500 was chosen to provide information on general market conditions, while Qualcomm was chosen due to the company's close relation to Apple.

Equation 1: Percent change in daily values

$$P = \frac{x(t) - x(t-1)}{x(t-1)}$$

Four of the most common technical indicators were used: moving average convergence divergence (MACD), relative strength index (RSI), on balance volume (OBV), and bollinger bands (BB). The data for these indicators were collected from Morningstar (https://www.morningstar.com), and several additional calculations were then performed in order to maximize the data's usefulness to the model. The MACD was calculated by subtracting the 26-day exponential moving average (EMA) from the 12-day EMA retrieved from Morningstar. The input used for the models was the signal line, which was the 9-day EMA, subtracted from the MACD. The time period used for the RSI was 14 days. The data for the BBs consisted of an upper band and a lower band, and these bands were two standard deviations away from a 20-day simple moving average. The inputs for the models were the closing price subtracted from the upper band value and the closing price subtracted from the lower band value.

The company data used included two common balance sheet metrics: the current ratio and the debt to equity ratio. The current ratio was calculated by taking the total current assets over the total current liabilities. The debt to equity ratio was the total liabilities divided by the total shareholder's equity. Several other balance sheet data were also used: total net sales, net income, free cash flow (FCF), and earnings before interest, taxes, depreciation, and amortization (EBITDA). The percent change found using Equation 1 was used for all four data types. FCF was calculated by first finding the capital expenditure by taking the difference between the property, plant, and equipment from this quarter and the previous quarter and adding it to the depreciation and amortization. Capital expenditure was then subtracted from net cash flows from operating activities to obtain FCF. All balance sheet data were obtained using

Apple balance sheets from Mergent Online (https://www.mergentonline.com). The balance sheets were only available quarterly, so the data was repeated throughout each quarter.

The news data was collected from ProQuest (https://www.proquest.com). The database was searched for the Apple company and specific publication titles: Wall Street Journal (Online), PR Newswire, TechCrunch, Business Wire, Washington Post Blogs, New York Times (Online), Barron's (Online), The Washington Post (Online), and Boston Globe (Online). Only the article titles and abstracts were used.

The outputs that the models were to try to predict were determined based on the percent change of the AAPL closing price. If it was above zero for day *t+1*, the output for day *t* was set as one to indicate a rise in the next day's stock price. Otherwise it was set as zero to indicate a decrease in the next day's stock price.

Data Preprocessing

The news data was filtered to remove titles and abstracts that were not directly relevant to Apple. This step assumed that general market information would have less of an effect on the Apple stock price than information specifically about Apple. Several keywords relevant to Apple were chosen: apple, iphone, ipod, mac, ios, ipad, siri, and itunes. The data was filtered to include only the titles and abstracts with at least one of these keywords. A title or abstract with a word containing one of the keywords, such as macbook, was kept.

The text was normalized in order to minimize out-of-vocabulary (OOV) words when generating word embeddings. All characters were made lowercase, and several HTML character entities were filtered out. Text in parentheses, phone numbers, emails, and urls were all removed. The hyphen (-) was replaced with a space, and all punctuation except for the apostrophe (') was removed. Stop words, which are common words that add little meaning to text, were removed. The textual data was limited to a maximum of the first five titles and first three abstracts per day. Days without any news data were left as empty strings.

The numerical and textual data were aligned with the calculated outputs to create input-output pairs. The data were then split into training and test sets, with the first 80% of the data in the training set and the remaining 20% in the test set. The training set was further split so that the last portion was a validation set that would help prevent the models from overfitting to the training data. The validation set was 20% of the entire data. The statistics of the dataset are shown in Table 1.

Table 1. Description of Dataset

	Training	Validation	Test
Time Interval	1/9/2015-1/2/2018	1/3/2018-12/31/2018	1/2/2019-12/27/2019
Days	751	250	250

All numerical data was normalized so most of the data fell in the -1 to 1 range. This was done by finding the mean, μ, and standard deviation, σ, of each field of data in the entire training set, including the validation data. Only the training set was used to prevent information leakage from the test data, which the model should know nothing about until testing. Each data point, a, in the entire dataset was then defined as $a = (a-\mu)/\sigma$.

The titles and abstracts from the news data were combined into one list of words for each day, and using the words in the training and validation datasets, a word index where each word corresponded to a number was created. The words in all the datasets were encoded into numbers using this word index. The resulting sequences of numbers were either truncated or padded with zeros so that each sequence had a length of 500 numbers.

Numerical Models

The two columns of historical Apple stock data and five columns of technical indicators comprised the technical data. This data was used for the Tech model. The five company data columns and the percent change of S&P 500 and QCOM closing prices comprised the numerical fundamental data, which was used for the Comp model. The input for both models consisted of a delayed sequence of length five, which means the data from the past five days were used to predict each next day. The input for the numerical models were matrices $N \in \mathbb{R}^{5\times7}$.

All of the models used in this research were built using Keras with Tensorflow. Both the Tech and Comp models had the same architecture, shown in Figure 1. They consisted of an input layer specifying the input shape followed by a bidirectional LSTM with 64 units and L2 kernel regularization. Bidirectional LSTMs can learn patterns going forwards and backwards in time, which can improve learning. L2 regularization penalizes large feature weights, and this makes the model more generic and helps avoid overfitting. Following this LSTM layer was another LSTM layer, and the only difference in this latter LSTM layer was that it had 32 units. The output layer was a dense layer, which is a typical layer that performs operations on the input tensor, with one output unit and the sigmoid activation function. Sigmoid transforms its input into a value between zero and one based on the probability that the expected output is zero or one. Dropout with a rate of 0.5 was used throughout the model. This means that half of the input units to each dropout layer are set to zero while the model is training, which helps prevent overfitting.

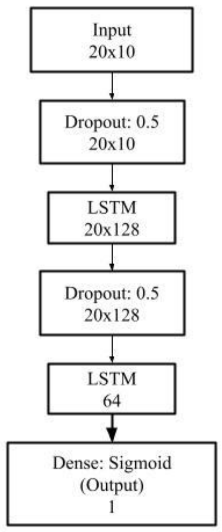


Figure 1. Numerical model architecture. The numbers underneath the layer names represent the output shapes of each layer.

Loss functions determine the error in a model's predictions compared to the expected output. The loss function chosen was a weighted binary cross-entropy. Binary cross-entropy is the most commonly used loss function for binary classification tasks such as this one. The outputs of the dataset were imbalanced, with more ones, or upward stock directions, than zeros, or downward stock directions. To counteract this and prevent the model from becoming biased towards predicting upward directions, weighted binary cross-entropy was used. A weight of 1.08 was given to the zero class, while a weight of 1.00 was given to the one class. This ratio is equivalent to the ratio of upward directions to downward directions in the training dataset.

Optimizers update the model's weights based on the output of the loss function, trying to improve the model's accuracy. The optimizer used for the two models was Adam with a learning rate of 10^{-4}.

The model was trained with the training data and tested with the validation data while model parameters were optimized. After the final design just described was reached, the models were trained on both the training and validation data. Both were trained for 30 epochs, which means the model passed through the entire dataset 30 times.

Textual Models

The input shape for the Text model was $N \in \mathbb{R}^{500}$ where 500 was the length of each text sequence. Only the news articles from day *t* were used to predict the direction of day *t+1*, as this was found to be the most effective strategy in [5].

The text model architecture is shown in Figure 2. The first layers of the model were an input layer specifying the input shape and an embedding layer. The embedding layer takes each encoded word in the sequence and transforms it into a vector known as a word embedding. This embedding contains useful information about the word. The word embeddings used by this embedding layer were from a word2vec model with pre-trained embedding vectors, trained on part of a Google News dataset. The vectors were obtained from [18]. The embedding layer was followed by a one-dimensional convolutional layer, or a temporal convolution, with 64 output filters and a convolution window of length five. The convolution helps the model recognize important information. Next was a temporal max pooling with a pooling window of four. This reduces the size of the data, which can reduce the computational cost. Following this were two bidirectional LSTM layers with 64 and 32 units and then two dense layers. The first dense layer had L2 kernel regularization and its activation function was the rectified linear unit (ReLU). ReLU is defined as $max(0, z)$. The second dense layer was the output of the model with one unit and the sigmoid activation function. Dropout with a rate of 0.5 was used throughout the model.

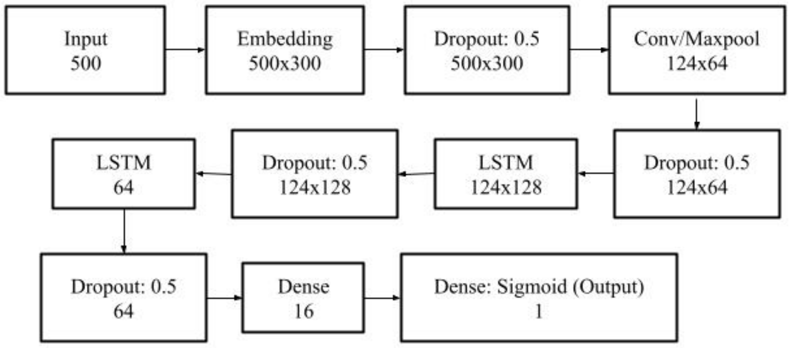


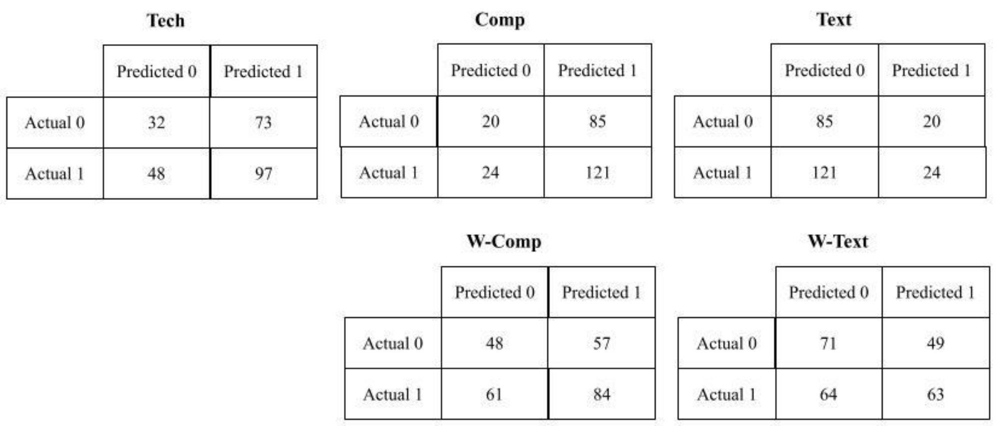
Figure 2. Textual model architecture. The numbers underneath the layer names represent the output shapes of each layer.

The loss function used was the same weighted binary cross-entropy used in the numerical model, with a weight of 1.08 given to the zero class and a weight of 1.00 given to the one class. The optimizer was again Adam, but with a learning rate of 5×10^{-5}.
After optimizing model parameters, the model was trained on both the training and validation data for 30 epochs.

Results

All models were evaluated on the test dataset. Figure 3 shows the confusion matrices for the models. Through comparing the model prediction to the actual values, the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) can be seen in the matrices. In this context, a positive refers to an output of one, or an upward direction in the Apple stock.

The weights used for the loss function were not optimal for the Comp and Text models as seen by the heavy prediction bias when testing the models on the validation data. Through testing different values, more optimal weights were found, and two more models, W-Comp and W-Text, were tested with them. W-Comp's loss function had a weight of 1.15 for the zero class and 1.00 for the one class. W-Text had a weight of 1.03 for the zero class and 1.00 for the one class.



Tech

	Predicted 0	Predicted 1
Actual 0	32	73
Actual 1	48	97

Comp

	Predicted 0	Predicted 1
Actual 0	20	85
Actual 1	24	121

Text

	Predicted 0	Predicted 1
Actual 0	85	20
Actual 1	121	24

W-Comp

	Predicted 0	Predicted 1
Actual 0	48	57
Actual 1	61	84

W-Text

	Predicted 0	Predicted 1
Actual 0	71	49
Actual 1	64	63

Figure 3. Model Confusion Matrices

The accuracies of the models during the training and test periods are shown in Table 2, where accuracy is defined in Equation 2. Because the dataset was skewed, the accuracies were compared to a benchmark accuracy of 51.80% for the training set and 58.30% for the test set. These are the accuracies that a trivial majority classifier that always guesses an upward direction would achieve.

Equation 2: Model accuracy

$$accuracy = \frac{correct\ predictions}{total\ predictions} = \frac{TP + TN}{TP + FP + TN + FN}$$

Table 2. Comparison of Model Performance

Model	Training Accuracy (%)	Test Accuracy (%)	Precision (%)	Inverse Precision (%)	Recall (%)	Inverse Recall (%)	F1 (%)	Inverse F1 (%)	MCC
Tech	51.65	51.60	57.06	40.00	66.90	30.48	61.59	34.59	-0.0278
Comp	52.45	56.40	58.74	45.45	83.45	19.05	68.95	26.85	0.0323
Text	52.65	43.60	54.55	41.26	16.55	80.95	25.40	54.66	-0.0323
W-Comp	51.45	52.80	59.57	44.04	57.93	45.71	58.74	44.86	0.0363
W-Text	52.45	55.20	56.25	52.59	49.61	59.17	42.72	55.69	0.0881

The precision, recall, and F1 scores for the positive and negative classes for the models are also shown in Table 2. They were calculated for the test dataset. These metrics are defined for the positive class in Equations 3–5. Inverse precision, recall, and F1 have the positive and negative labels exchanged so that the metrics provide information about the negative rather than the positive class. For the positive class, precision is the proportion of predicted upward directions that were correct, while recall is the proportion of upward directions that were identified. The F1 score is a way of combining the precision and recall through finding the harmonic mean of the two.

Equation 3: Model precision

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

Equation 4: Model recall

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

Equation 5: Model F1 score

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

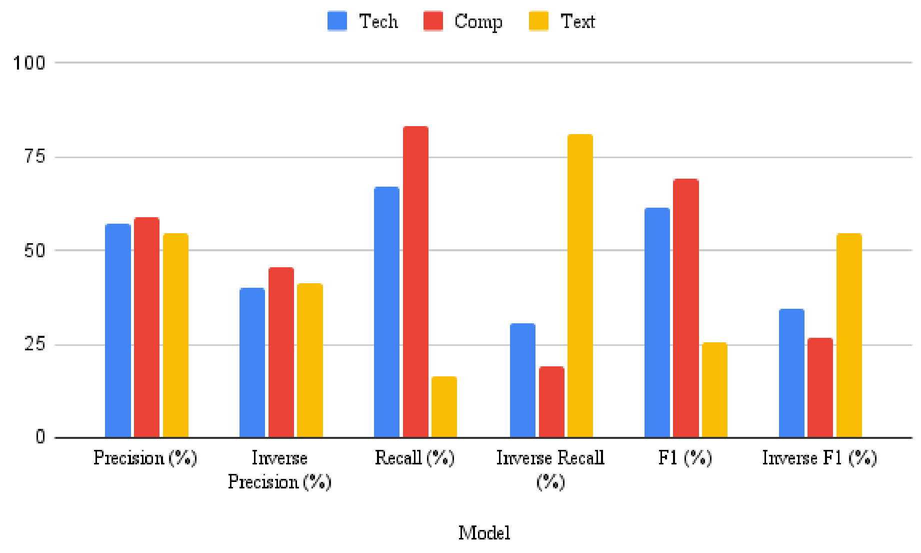


Figure 4. Comparison of model precision, recall, and F1 scores for positive and negative classes. Metrics are shown for only the three original models. Inverse precision, recall, and F1 score are metrics for the negative class.

Matthew's correlation coefficient (MCC) is another metric that is useful for imbalanced classes. It takes into account both the model's performance in all four categories of the confusion matrix and the sizes of the two classes. It produces a result from -1.0 to 1.0, where 1.0 is a perfect classifier. MCC is defined in Equation 6. The MCCs for the models are shown in Table 2.

Equation 6: Model MCC

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Discussion

The imbalanced data posed a significant challenge for the models and resulted in noticeable prediction bias. The Tech model was biased towards the upwards direction as can be seen by the confusion matrix, where 170 upward directions were predicted compared to 80 downward directions. This disparity was much less when testing on the validation dataset, so a W-Tech model with different weighting was not tested. Even with the test dataset, the Tech model was less biased than the Comp and Text models.

The Comp model was heavily biased towards the upward direction, while the Text model was biased towards the downward direction. It is unclear what exactly the reasons are for the Comp model's bias, although it is likely the difference is due to the type of data used. The numerical fundamental data may have been more susceptible to the imbalanced classes. It is also possible that the data would lead to more upward predictions even if the classes were not imbalanced, which could be a potential area for future research. The Text model's bias was due to the weight on the downward direction for the loss function being too great. This suggests that the Text model was less susceptible to the imbalanced data. A possible explanation of this is the greater information in the text data, which often consisted of multiple sentences worth of word embeddings, compared to the numerical data.

Two additional models, W-Comp and W-Text, were trained and tested after observing the noticeable bias of the Comp and Text models during validation. These models were tested to attempt to compare the performance of the models beyond the limitations of the imbalanced dataset used.

The W-Comp model predicted a far more even amount of each class, although it was still slightly biased towards the upward direction. W-Comp predicted 141 upward directions compared to 109 downward directions, while Comp predicted 206 upward directions compared to 44 downward. However, the W-Comp model did not perform much better overall than the Comp model, with an MCC that was only marginally higher: 0.0363 compared to 0.0323. Observing the metrics for the positive and negative classes, W-Comp's performance predicting the upward direction worsened slightly. This was due to a significant drop in recall due to the smaller number of upward predictions. The F1 score for the negative class improved, though. While the inverse precision was slightly worse, W-Comp's inverse recall was far higher.

The W-Text model was also less biased than the Text model, although it was still slightly biased towards the negative class. It predicted 135 downward directions and 115 upward directions compared to the Text model's 206 downward directions and 44 upward directions. W-Text did noticeably improve compared to Text, although the difference was still not extreme. W-Text's MCC was 0.0881 compared to -0.0323 for Text. The metrics for the positive class were all improved for W-Text. The inverse F1 was slightly better as well. Inverse recall was lower due to less downward predictions, but inverse precision showed significant improvement.

Of the three original models, the Comp model had the best overall performance. It had the highest MCC, although the difference between it and the second highest was rather small: 0.0323 compared to -0.0278. The Comp model also had the highest test accuracy and positive class metrics, but it should be noted that this can likely be somewhat attributed to the higher number of upward predictions made by the model combined with the greater number of upward directions in the data. For the negative class, the inverse recall was very low, but the inverse precision was the highest of the three models. This may indicate good performance of the model beyond the imbalanced data skewing results. Even after the weighting was adjusted, the W-Comp model had the highest precision of all the models, and its inverse precision and MCC was second only to the W-Text model. This suggests that numerical fundamental data, including company data which made up the majority of the input dataset, is effective for predicting the market.

Textual news input also showed potential. The W-Text model had the highest final MCC and showed good performance in the other metrics as well. Its lower susceptibility to bias might also be a benefit. A way to improve the performance of the textual models might be to improve the quality of the news data used and the representation of the data in the models. Named entity recognition (NER), where named entities such as companies are identified, combined with sentiment analysis might be a more effective technique. NER could identify news directly pertaining to the company whose stock is being predicted, which would help ensure the data's usefulness. It would likely be more effective in this task than the keyword filtering used in this study was. Sentiment analysis could be used to determine if the news is positive or negative, and predictions could be made based on this. This approach has been found to be effective in [16]. Other techniques such as paragraph-level embedding rather than word-level embedding might also improve performance.

The performance of all the models was very poor, which indicates that further optimization of the datasets and model designs could be used to improve the performance of the models. It may also be beneficial to rebalance the dataset used. One possible future strategy would be to increase the time period of the dataset. The increased time period could be chosen to minimize the imbalance in the classes, and the additional data would also allow the model to learn more patterns and therefore improve prediction accuracy.

More thorough comparisons of the different data types are needed to allow for more solid results. This might include comparing the three data types with models using multiple hyperparameter combinations. This would prevent a particular model's performance on a particular combination from changing the results. Observing the performance of the data types across hyperparameters might also shed light into the different data and their relationships to the stock price. The different data types could also be used to predict various stocks other than Apple in order to ensure the universality of the results.

Any future models could be tested using a trading simulation in order to assess the models' performance in a real-life application. This test would determine if using the models' predictions to trade the stock being predicted would be effective, and it has been used in multiple studies [1], [4], [7], [9].
One slightly different direction for future study is predicting the stocks of smaller companies with less news coverage. There is a lot of news related to Apple available, but this is not true for most companies. It is desirable to develop models that can predict these stocks as well.

Acknowledgements

Thank you to the research faculty at my school, especially my advisors. Thank you to my parents for their help and support.

References

[1] M. R. Vargas, C. E. M. dos Anjos, G. L. G. Bichara and A. G. Evsukoff, "Deep Learning for Stock Market Prediction Using Technical Indicators and Financial News Articles," 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 2018, pp. 1-8, doi: 10.1109/IJCNN.2018.8489208.

[2] A. Arévalo, J. Niño, G. Hernández, and J. Sandoval, "High-Frequency Trading Strategy Based on Deep Neural Networks," Intelligent Computing Methodologies, pp. 424–436, 2016, doi: 10.1007/978-3-319-42297-8_40.

[3] E. Chong, C. Han, and F. C. Park, "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies," Expert Systems with Applications, vol. 83, pp. 187–205, Oct. 2017, doi: 10.1016/j.eswa.2017.04.030.

[4] X. Zhong and D. Enke, "Predicting the daily return direction of the stock market using hybrid machine learning algorithms," Financial Innovation, vol. 5, no. 1, Jun. 2019, doi: 10.1186/s40854-019-0138-0.

[5] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Using structured events to predict stock price movement: An empirical investigation," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, doi: 10.3115/v1/d14-1148.

[6] G. P. C. Fung, J. X. Yu, and W. Lam, “News Sensitive Stock Trend Prediction,” Advances in Knowledge Discovery and Data Mining, vol. , pp. 481–493, 2002, doi: 10.1007/3-540-47887-6_48.

[7] M. Hagenau, M. Liebmann, and D. Neumann, “Automated news reading: Stock price prediction based on financial news using context-capturing features,” Decision Support Systems, vol. 55, no. 3, pp. 685–697, Jun. 2013, doi: 10.1016/j.dss.2013.02.006.

[8] A. Khadjeh Nassirtoussi, S. Aghabozorgi, T. Ying Wah, and D. C. L. Ngo, “Text mining for market prediction: A systematic review,” Expert Systems with Applications, vol. 41, no. 16, pp. 7653–7670, Nov. 2014, doi: 10.1016/j.eswa.2014.06.009.

[9] G. Rachlin, M. Last, D. Alberg and A. Kandel, "ADMIRAL: A Data Mining Based Financial Trading System," 2007 IEEE Symposium on Computational Intelligence and Data Mining, Honolulu, HI, USA, 2007, pp. 720-725, doi: 10.1109/CIDM.2007.368947.

[10] M. R. Vargas, B. S. L. P. de Lima and A. G. Evsukoff, "Deep learning for stock market prediction from financial news articles," 2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), Annecy, 2017, pp. 60-65, doi: 10.1109/CIVEMSA.2017.7995302.

[11] T. Vu, S. Chang, Q. T. Ha, and N. Collier, “An experiment in integrating sentiment features for tech stock prediction in Twitter,” presented at the Proceedings of the Workshop on Information Extraction and Entity Analytics on Social Media Data, Mumbai, India, Dec. 2012, [Online]. Available: https://www.aclweb.org/anthology/W12-5503/.

[12] B. Wuthrich, V. Cho, S. Leung, D. Permunetilleke, K. Sankaran and J. Zhang, "Daily stock market forecast from textual web data," SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218), San Diego, CA, USA, 1998, pp. 2720-2725 vol.3, doi: 10.1109/ICSMC.1998.725072.

[13] P. D. Yoo, M. H. Kim and T. Jan, "Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation," International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 2005, pp. 835-841, doi: 10.1109/CIMCA.2005.1631572.

[14] Y. Zhai, A. Hsu, and S. K. Halgamuge, “Combining News and Technical Indicators in Daily Stock Price Trends Prediction,” Advances in Neural Networks – ISNN 2007, vol. 4493, pp. 1087–1096, Jun. 2007, doi: 10.1007/978-3-540-72395-0_132.

[15] X. Pang, Y. Zhou, P. Wang, W. Lin, and V. Chang, “An innovative neural network approach for stock market prediction,” The Journal of Supercomputing, vol. 76, no. 1, Jan. 2018, doi: 10.1007/s11227-017-2228-y.
[16] H. L. Siew and M. J. Nordin, "Regression techniques for the prediction of stock price trend," 2012 International Conference on Statistics in Science, Business and Engineering (ICSSBE), Langkawi, Malaysia, 2012, pp. 1-5, doi: 10.1109/ICSSBE.2012.6396535.

[17] S. Shen, H. Jiang, and T. Zhang, "Stock Market Forecasting Using Machine Learning Algorithms," Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, 2012. [Online]. Available: http://cs229.stanford.edu/proj2012/ShenJiangZhang-StockMarketForecastingusingMachineLearningAlgorithms.pdf

[18] Google Word2vec, “Tool for computing continuous distributed representations of words,” Google Code Archive, Jul. 29, 2013. https://code.google.com/archive/p/word2vec/