CS230

Long-Term Stock Prediction using Quantitative and Text-Based Data

Category: Natural Language Processing & Recurrent Neural Network

Dylan M. Crain
Department of Energy Resources Engineering
Stanford University
cooper96@stanford.edu

Project Description

Stock market prediction is a challenging and fairly well-researched topic within deep learning. Due to the *Efficient Market Hypothesis*[3], all available public data should already be incorporated into the current stock price for a given company. This means that it should be unfeasible to garner any predictive information from these trends. However, according to my literature review, this appears to not be the case in reality.[3][4] Using stock price data can lead to excellent next day predictions.

Furthermore, there has been work on honoring the *Efficient Market Hypothesis* by taking in news reports or even Tweets to inform a deep learning network that predicts future stock prices.[2] What is not as common, though there are cases such as in reference [1], is the idea of combining quantitative information, e.g., end of day price & volume, with text-based data, such as news articles or social media posts.

Moreover, I have currently found no source that combines this data space with longer term stock prediction, i.e., beyond just the next day. Therefore, my goal in this project is to use both bases of data (text as well as quantitative) to build a Long-Short Term Memory structure to predict stock prices at different time frames, i.e., not just the next day. These methods are chosen, since they appear to be well used for such a problem due to the time series nature of stock prices. If time and results permit, I would like to use this model to possibly predict economic downturns close to a week in advance.

Dataset

Thoughts concerning the dataset have been the main focus thus far on the project. In terms of numeric data, e.g., end of day price, high, low, and trading volumes, the NASDAQ stock exchange[5] offers data up to ten years prior (late 2011) for the securities under its umbrella. This is a convenient, and well organized, set of data that will be taken advantage of.

The acquisition of textual data is markedly more challenging. A primary question comes up in terms of which source to focus on. Twitter would be an interesting and insightful one to draw from, yet this may focus more on lower spenders than more established sources of information such as the New York Times or the Wall Street Journal. One of the challenges of this setup is finding news articles that more or less provide data on each stock of interest for every day under investigation. Thus, with this in mind, the idea moving forward is to collect both sources of textual data, i.e., from Twitter and news reports.

The next step, then, is to decide on a method to gather the textual data. One approach would be to use web-scraping tools in Python, such as *Beautifulsoup* to gather this information from the web.

This would be most effective on Twitter, since it could just be taken from the main feed. However, it can certainly be done for reputable news sources as well, as can be seen from the *GitHub* repository in [6].

A secondary approach, which may prove more lucrative, would be to use public APIs for the textual gathering approach. Clearly, there is such an API available for Twitter, but there are also more generally available approaches such as the "Free stock market and financial news API"[7], which is directly applicable to my problem.

Regardless of the approach, the text can be selected which contains either the financial keyword, e.g., AAPL, or the actual company name. Along this line of thought, the incorporation of the text data, in terms of the entire ten year period of available closing day prices needs to be taken into account. For example, if there is serious textual data sparsity in earlier dates, it may be best to leave that out of the used dataset.

Approach

Work Completed

The work done so far has primarily focused on further literature review and refining the problem setup. Initially, the possibility of using a simple RNN or a LSTM was debated. In fact, it was implied in the *Proposal* that both methods may be used with a comparison. Upon further research, this extra work appears to be unnecessary. LSTM has many advantages over the simple RNN – in this field as well as in general. The simple RNN possess some serious *vanishing gradient* problems that LSTM overcomes. Furthermore, LSTM operates well when there is a sparsity of data. For example, with the textural portion of the inputs, there may be days when there is no news reporting on a given stock – Akita et al. 2016[1] sets the corresponding text vector to zero in such instances. Furthermore, one of the goals of this project is to see how far out the prediction can accurately predict stock behavior. LSTM has the advantage of learning more long-term characteristics of the data, which make it an ideal learning model for this setup.

With this established, the next step is to use natural language processing to determine how to deal with the textual data as an input. For example, each article or tweet has a variable number of words contained within them, so how an input vector can be created for this setup is an important question to answer. One straight-forward approach is to create a dictionary based on words that appear more than k times throughout all of the data. From here, one can use a simple setup of 1 indicating that the word exists and a 0 if it does not. A more realistic approach, in terms of natural language processing, would be to account for how the words line up with one another. A similar approach to this was what I did for my project in CS 229. One method, that has shown a great amount of success[2], [8], is to use a *Word2vec*[9] model to encode each word in a text submission and then average across all words to get an unique encoding for each article of a given length. In fact, reference Hu et al. 2018[2] used a pre-trained model for their setup, which I may replicate if need be.

Finally, the model setup will consist of two input streams to the LSTM. The first is from the stock information. The second will be the text encodings that are relevant for a given time-frame. An important question will be how far back the history should be incorporated. Stoean et al. (2019)[4] used 14 trading days prior, so that may be a good place to start. The output from the LSTM will go to a fully-connected layer that uses a softmax activation function to provide a binary class label for each stock. That is, the output of the model will be to predict whether a given stock will increase or decrease for a given time period, e.g., one day, three days, or a week.

Next Steps

With the data collection and model setup more or less fleshed out, the next steps involve implementing the ideas laid out. First of all, the data collection needs to be completed. The process of gathering the quantitative data is straight forward, but one question to be answered is how many and which stocks should be included in the study. For example, some securities may not have existed ten years ago, so this needs to be determined.

The next step, and by far the more challenging, is to decide on a textural collection strategy as well as exactly which sources to collect from. The API for news articles related to the stock market seems

promising, but more investigation is necessary. After collecting the data, it must be cleaned up by creating a dictionary of relevant words and eliminating end of sentence words.

Finally, the deep learning strategy proposed is to be fully implemented and used upon the collected data sets. This, of course, will not be the end of the project since deep learning is an inherently iterative process, so many more steps are stored here.

All-in-all, there is a lot to do in approximately three weeks. However, with focus and determination, I believe an excellent project and presentation is still on the horizon.

References

[1] R. Akita, A. Yoshihara, T. Matsubara and K. Uehara, "Deep learning for stock prediction using numerical and textual information," *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 2016, pp. 1-6, doi: 10.1109/ICIS.2016.7550882.

[2] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. "Listening to Chaotic Whispers: A Deep Learning Framework for News-oriented Stock Trend Prediction". *In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18).* Association for Computing Machinery, New York, NY, USA, 261269. DOI:https://doi.org/10.1145/3159652.3159690

[3] F. Kamalov, L. Smail and I. Gurrib, "Stock price forecast with deep learning," *2020 International Conference on Decision Aid Sciences and Application (DASA)*, 2020, pp. 1098-1102, doi: 10.1109/DASA51403.2020.9317260.

[4] Stoean C, Paja W, Stoean R, Sandita "Deep architectures for long-term stock price prediction with a heuristic-based strategy for trading simulations". PLOS ONE 14(10): e0223593. https://doi.org/10.1371/journal.pone.0223593

[5] NASDAQ Stock Exchange. "https://www.nasdaq.com/"

[6] sky-97. scraping-wall-street-journal. "https://github.com/sky-97/scraping-wall-street-journal"

[7] marketaux. "https://www.marketaux.com/"

[8] Manuel Vargas, Beatrix de Lima, and Alexandre Evsukoff. 2017. "Deep learning for stock market prediction from financial news articles". *2017 IEEE.* https://d1wqtxts1xzle7.cloudfront.net/62276961/Deep_learning_for_stock_market_prediction_from20200304

[9] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space".