Increasing the Accuracy of Intra-day Stock Return Predictions Using Textual Information

6.867 Term Project, Fall 2020: Strategy Document

***Abstract*—We will learn powerful word embeddings from news articles and use these textual features in conjunction with widely used numerical inputs to improve the accuracy of predicting stock returns. Our model will inform investor decisions, such as whether to buy, sell, or hold shares, or assist in the firm's corporate planning. We consider two learning tasks. The first is unsupervised learning on the textual data we scrape from the web to learn word embeddings. The second task is supervised learning to predict stock returns based on both textual (qualitative) and numeric (quantitative) data. We consider and experiment with a variety of possible approaches to generate word embeddings and predict returns. We compare these modeling approaches in terms of their predictive power and how well they generalize to different stocks, sectors, and/or time periods.**

I. Problem Motivation & Definition

Natural Language Processing (NLP) is advantageous as a wealth of knowledge which exists in textual form is not leveraged to the full extent by current models. The use of NLP in the financial sector is still relatively new, and thus has the power to yield substantial improvements over traditional techniques. Most of the current work at intersection of NLP and finance focuses on sentiment analysis, SEC forms, and social media data. The use of news articles has been explored less, and thus will be the focus of this project. Further descriptions on current work using NLP in the financial sector is described in the literature review (section II) and our discussion on future research (section VII).

We investigate the effect of including textual features on predicting daily stock return. We choose to predict return, rather than raw price, because (1) it is more challenging, and (2) returns are more readily used to evaluate investment decisions.

A. Objective

1) Predicting Intra-Day Returns: Define $r_{i,t}$ as the return of stock i on day t, $o_{i,t}$ on the opening price on day t, and $c_{i,t}$ on the closing price on day t. We will predict the daily return of the stock, which is given for stock i on day t by eq. (1):

$$r_{i,t} = \frac{c_{i,t} - o_{i,t}}{o_{i,t}} \tag{1}$$

2) Secondary Objectives: In addition, we may also consider the objectives of predicting stock price directly (regression) and/or stock price movements (i.e. classification as UP/-DOWN).

II. Literature Review

Heeyoung et al. [1] investigate using 8-K documents and textual analysis for classifying stock price movements (i.e. up, down, stay) over a short time period. They report a relative accuracy boost of over 10% from models using only traditional numerical features. Lastly, they conclude the impact of using textual features from the 8-K financial documents is most important in the short term, but the effect can continue for up to five days following the document's publication.

Oleg [2] conducts a survey and implementation of several stock prediction models which incorporate textual features. They investigate word and character-level embeddings, alongside a BiLSTM encoder to obtain a 'context vector' from news articles. This vector is then stitched with numeric features and used for basic regression or classification tasks to predict stock returns or price direction, respectively.

Neural networks have been used extensively to (1) learn word embeddings and extract textual features, and (2) predict future stock price movements via regression or classification-based approaches. There are several well-established methods for creating word embeddings based on neural networks, such as `Word2Vec` and `GloVe` [3]. Other methods (e.g. `Doc2Vec`) have been developed to create distributed representations of sentences or paragraph vectors and convert whole documents to vector representations [4]. Furthermore, several feed-forward networks have been proposed for tasks like dependency parsing [5], sentiment analysis, part-of-speech tagging and sentence compression [6]. Variants of recurrent neural networks (RNNs) such as Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) networks [7] have been found to be most effective for improving the accuracy of stock price predictions over an extended time horizon.

III. Methods

In this section we describe our methods to (1) learn word embeddings and their influence on stock prices, (2) infer the relevance of specific news articles to stock performance, (3) produce prediction models for stock prices based on numeric and textual features.

A. Word Embeddings

To include textual features in our model, we must first map news articles to vectors of real numbers, referred as 'word embeddings.' These vectors will be joined with the numeric data in the prediction task. There are numerous ways to obtain such vectors, with the simplest one being one-hot encoding of words. However, there are two main issues under this representation: (1) if the number of words is large, then training is nearly infeasible due to computational limitations, and (2) embeddings are all orthogonal to one another, which means that words with similar meanings (such as 'motel and 'hotel') will not be near each other in the subspace. We could impose restrictions on our vocabulary to remedy the first issue, but this does not remedy the latter one. Therefore, we will not consider this approach in our model.

Ideally, we would like our embeddings to live in a subspace of relatively small dimension, and where words similar in meaning are close to one another. Thus, we must consider stronger and more complex word embedding methods. We highlight three such methods in the below subsections.

1) `Word2Vec`: `Word2Vec` uses a simple, two-layer neural architecture to map every word to a k dimensional subspace, where each dimension represents a peculiar characteristic of the embedding. For instance, one dimension might represent the gender or the plural of a noun. As a result, words with similar attributes will live close together in this subspace, which can be measured via the cosine similarity between embeddings. Moreover, by nature of our embeddings living in this subspace, we can take linear combinations of words to obtain new embeddings that preserve the overall meaning. To illustrate this principle, consider the following quintessential example, which we obtain using Google's pre-trained `Word2Vec` model. Letting v_w represent the trained embedding of word w, the embedding defined by the linear combination $v_{king} - v_{man} + v_{woman}$ is closest to v_{queen} in our subspace. This highlights the transfer of meaning across combinations of embeddings, unlike the one-hot representation.

We will first use Google's pre-trained `Word2Vec` model as a starting point [1]. We will also train our own `Word2Vec` model using the scraped news articles. Depending on the success of `Word2Vec` models, we may consider additional methods to determine word embeddings.

2) TF-IDF & PCA: The Term Frequency - Inverse Document Frequency (TF-IDF) is yet another method to obtain word embeddings. Define each word/term as t, and document $d \in (1, ..., n)$ as the document number. Then, define TF(t, d) as the term frequency, where t is the number of times term t appears in document d. Similarly, let DF(d, t) represent the document term frequency, which is the number of documents term t appears in divided by the size of the corpus. We can then compute the TF-IDF score for a given term with eq. (2).

$$\begin{aligned} \text{TF-IDF}_{t,d} &= \text{TF}(t, d) \cdot \text{IDF}(t) \\ \text{IDF}(t) &= \log\left(\frac{1+n}{1+\text{DF}(d,t)} + 1\right) \end{aligned} \tag{2}$$

[1] `https://code.google.com/archive/p/word2vec/`

This approach is sensible because words with high TF-IDF scores can be seen as more discriminating with respect to the overall meaning of a given article. However, since we are considering all words in our corpus, we need to use dimension reduction techniques to obtain embeddings amenable for training. As such, we will couple this approach with Principal Component Analysis (PCA) and sparse non-negative matrix factorization to extract only the dominant features [1], [8].

3) `GloVe`: Lastly, we may consider `GloVe` as an additional method to obtain word embeddings. Whereas `Word2Vec` trains embeddings based on *local* context by considering only the closest words in a sentence, `GloVe` extends this principle by adding a *global* co-occurrence matrix of all the words in the training corpus, which may yield more precise embeddings.

B. *Relevancy of Articles to Stock Price Performance*

To ensure that the articles we use to generate daily word embeddings are relevant to the financial sector, we restrict ourselves to a set of trusted news sources: Wall Street Journal, New York Times, The Economist, Washington Post, Seeking Alpha, Bloomberg, NPR, CNBC, NBC, ABC, Fox News, USA Today, BBC News, CNN, Market Watch, Reuters, and the Associated Press. After restricting to trusted sources, we filter the list of articles to only those related to the company's financial performance of the company. We consider two approaches for this task:

1) Topic modelling via Latent Dirichlet Allocation (LDA): LDA is a generative probabilistic modelling technique that treats documents as mixtures of topic probabilities and topics as mixtures over the underlying vocabulary of words. This method will be used to determine the main topics of the articles scraped for a particular ticker for each day.

2) Clustering Articles via K-Means: We will use K-Means clustering to group scraped articles based on their word embedding. We will then identify the 'financial' clusters and include only these articles when composing the textual features to join with numeric data.

C. *Machine Learning Prediction Models*

1) Traditional Techniques: As a baseline, we will implement classical methods such as Ordinary Least Squares (OLS) and Auto Regressive Integrated Moving Average (ARIMA) to fit models on historical training data. This will be done using (1) only the numerical features, and (2) both the numerical inputs and textual features. We will check for multicollinearity and drop numerical features as needed while performing OLS and ARIMA modeling. Textual features will be incorporated by adding the lower dimensional textual feature vector (obtained via word embeddings) as a regressor. These traditional models will serve as a reference point to judge the performance of more sophisticated supervised learning models.

2) Neural Network Models: The parametrized vector word embeddings (combined with numerical features) are then fed in as inputs to a neural network [9]. We will experiment with several different architectures using combinations of the following layers, both with and without textual features:

- Linear: Regular dense, fully connected feed-forward layers with varying numbers of neuron units.
- Convolutional Neural Networks (CNNs): Past studies have applied character-level CNNs directly to extract spatial information from raw text [10]. We will not be using character-level CNNs in this context, as we are working with already processed embeddings instead. However, 1D convolutional filters can also provide limited temporal memory by applying sliding windows of fixed size to time-series historical data. We will also tune the hyperparameters of CNN layers such as the number of output filters, kernel size or width, stride length, dilation rate etc. CNN layers will be followed by 1D pooling layers that either average or take the maximum value over time.
- Recurrent Neural Networks (RNNs): These layers include cycles which feed in outputs from previous time steps, making it well-suited to model sequential data. At each time step the RNN takes in the current input vector along with the previous hidden state vector to produce the new hidden state. This allows it to summarize historical information up to that point [10].

  We will also consider both GRUs and LSTMs to address some of the limitations of RNNs. GRUs use *reset* and *update* gates to adaptively control the information used inside each unit [11]. Similarly, LSTMs use a hidden memory cell and four gating units to 'remember' and 'forget' (or drop) information from previous states as needed. This allows the model better learn long-range dependencies, which could be useful when applied to our historical financial data [7]. Finally, we can further improve performance using self-attention weighting schemes and bidirectional RNNs (that process sequences from both start to end and vice versa) [2].

We will test several different architectures combining one or more of the above mentioned layer types. In addition to optimizing network design, we will tune various hyperparameters involved. We will tune the learning rate by including adaptive step-sizes and momentum-based methods like `ADAM`, the batch size, and the number of training epochs. Figure (a) shows the information workflow and Figure (b) shows one possible neural network configuration. In addition, we will consider several candidate losses, such as Mean Squared Error (MSE) and binary cross entropy depending on the task. We will also consider different activation functions (e.g. $ReLU$ or leaky $ReLU$ for regression; and $sigmoid$, $tanh$ or $softmax$ for classification tasks). Finally, we will explore using dropout as a means of regularization to prevent overfitting and increase generalizability. This is particularly relevant for more complex architectures, where the number of model parameters may be large relative to the amount of historical training data (numerical and textual) available.

IV. Data Acquisition and Preparation

A. News Articles via Google News API & Web Scraping

We scrape news articles via the Google News API. For each calendar date and company we scrape the first webpage, which gives at most 20 articles for that day. Our search term in the Google News API is simply the company name, for instance, 'Amazon.'

B. Numeric Financial Data via Yahoo Financials Package

We use Python to pull historic stock prices and market indicators. Using sources [12], [13], and [14] as a guide, we assembled 11 numeric features. Our numeric features are the opening stock price; opening VIX price; opening DOW, NASDAQ, and S&P 500 prices; and the daily Fama-French [14] factors.

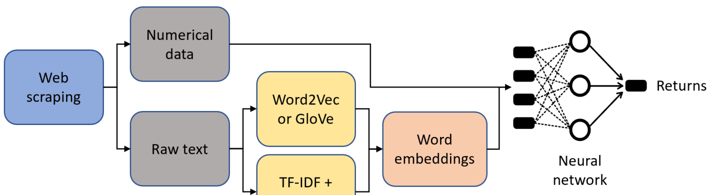
V. Analysis & Potential Issues

In this section we discuss several concerns data relevance, quality, and accuracy.

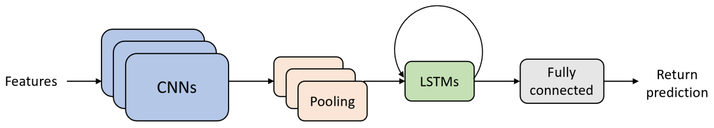
First, although we institute restrictions on which article sources we consider, it is possible that irrelevant articles are still included. For instance, consider an NBC article on new features of Amazon Alexa. This article is from a trusted source with wide circulation, but may not be directly relevant to stock price.

Second, we may include articles containing old or 'stale' information. For instance, an ABC article summarizing an AMZN stock downturn from months prior. While our word embeddings will capture the effect of this article (i.e. the downturn of the stock), it is likely that its effects have already been realized in the stock price. This is the Efficient Market Hypothesis [15], [16] coined by Eugene Fama in the 1960s, which states that at any given time in a liquid market the prices currently reflect all public information. If this information (i.e. the article) has already been priced into the stock, this may inhibit our model training. If we learn word embeddings which represent the downturn but we see no market effect, this would indicate a lag in our training and we would not be learning effectively.

Third, the demographics of daily articles could inhibit model performance. For a given stock the article demographics (i.e. the number of articles and their sources) may vary greatly from day-to-day. Consider our `Word2Vec` embedding. When we take the linear combination of words to make each day's features, this vector representation could over



(a) Overall workflow of our model.



(b) Illustrative example of a possible neural network architecture.

or underestimate certain dimensions because there is either too much or too little article saturation. To combat this, we may need to ensure we have close quantity symmetry within tickers and across days.

Fourth, as we are using the Google News API to scrape data, we need to provide a search term for the API. We discussed the chosen search term in section IV-A, but it is possible that this search could still include articles unrelated to financial and stock data. For instance, in our search for 'Amazon' on Google News could return an article on the Amazon rainforest.

Finally, rhetorical devices like sarcasm and irony may be present in certain articles, representing another layer of complexity that our embedding may not be able to capture. For instance, *The Economist* uses sarcasm and irony quite frequently, which can pollute word embeddings. We leave detection of sarcasm and irony as future work.

VI. Empirical Evaluation

In this section we describe and construct a proof of concept case. We will create a baseline stock price prediction model containing only numeric features. We will use a simple regression or random forest model and compute error metrics like Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Error (MAE).

Using this baseline comparison, we investigate whether textual features can and should be used to enhance predictive power. Thus, it follows that we should compare this baseline model against a similar model with textual features. Using the aforementioned regression or random forest model, we will then include textual features and compare performance with the same error metrics. To guarantee that adding textual features improves predictive performance, we will manually select articles to include in this model. For each day in our baseline model we will select one article which we *know* is 'fresh' and/or 'breaking news,' thereby avoiding the concerns addressed in section V. If we see more favorable error metrics by including textual features, then we can deduce that there is some gain in predictive power by using textual features in addition to numeric features. Although we do circumvent potential pitfalls by hand-selecting articles, this is impossible to repeat on a large scale and therefore we would need a more complicated process to automate this same procedure over a longer time frame and for a larger number of stocks.

In addition, we will report comparisons with models trained on different corpora. As some corpora may hold much more explanatory power for stock prices, we hypothesize that we will see nontrivial differences that arise from training on different corpora. We will also report on multiple weighting schemes for our embeddings. This will allow us to reasonably conclude whether textual features are helpful or not.

VII. Future Work

In this section we address areas for continued work and future research. First, we recommend including other news hosting sites besides Google News. By including other sources we would have access to additional, potentially popular and influential articles which may not be highlighted on the first page of Google News. For instance, we could include Apple News and Yahoo News.

Second, we would like to learn word embeddings and their predictive power for articles in a variety of languages. For instance, [17] discusses using `Word2Vec` on Chinese articles while [18] implements it on English and Russian articles. In [19], a team at Google discusses how to exploit similarities in languages for machine learning. For this project, we restrict our corpus to articles in English; and in doing so we may potentially lose some influential and highly telling articles.

Third, as aforementioned, some articles use sarcasm and/or irony. We propose future work to identify, extract, and use sarcasm and irony to enhance predictive power. [20] presents work using Support Vector Machines to classify tweets as sarcastic, and [21] conduct a similar investigation for irony. These sources, and many more, could serve as a starting point for this extension.

Fourth, we would like to include textual sources other than news sources. We propose to use SEC 8-K forms and social media data (e.g. Twitter, Facebook, Reddit, etc.) to enhance the feature set used for return prediction.

Fifth, as described in section IV-A, we use the Google News API to scrape articles data using a specific, stock-specific search term. This forms our corpus to train `Word2Vec`. As a further line of work we propose forming an industry-specific corpus and retraining a `Word2Vec` model. For instance, consider training a `Word2Vec` model for the Technology sector. This model could allow us to enhance stock price prediction using articles and features that a simple search for 'Amazon' on Google news would not have gathered.

REFERENCES

[1] H. Lee, M. Surdeanu, B. MacCartney, and D. Jurafsky, "On the importance of text analysis for stock price prediction," in *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, pp. 1170–1175, European Language Resources Association (ELRA), Jan. 2014. 9th International Conference on Language Resources and Evaluation, LREC 2014 ; Conference date: 26-05-2014 Through 31-05-2014.

[2] O. Mitsik, "A text-based forecasting model for equity trading," 2019.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[4] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, 2014.

[5] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 740–750, 2014.

[6] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins, "Globally normalized transition-based neural networks," *arXiv preprint arXiv:1603.06042*, 2016.

[7] H. Jia, "Investigation into the effectiveness of long short term memory networks for stock price prediction," *arXiv preprint arXiv:1603.07893*, 2016.

[8] F. Ming, F. Wong, Z. Liu, and M. Chiang, "Stock market prediction from wsj: text mining via sparse matrix factorization," in *2014 IEEE International Conference on Data Mining*, pp. 430–439, IEEE, 2014.

[9] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[10] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *arXiv preprint arXiv:1508.06615*, 2015.

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[12] I. Shah, "Historical stock price data in python." https://towardsdatascience.com/historical-stock-price-data-in-python-a0b6dc826836, Feb. 2020.

[13] R. Aroussi, "Reliably download historical market data from yahoo! finance with python," Apr 2019.

[14] R. Low, "Asset pricing factor regressions," Feb 2019.

[15] B. G. Malkiel, "The efficient market hypothesis and its critics," *Journal of Economic Perspectives*, vol. 17, pp. 59–82, March 2003.

[16] T. Delcey, "Efficient Market Hypothesis, Eugene Fama and Paul Samuelson: A reevaluation." working paper or preprint, Oct. 2017.

[17] X. Chengzhang and L. Dan, "Chinese text summarization algorithm based on word2vec," *Journal of Physics: Conference Series*, vol. 976, p. 012006, feb 2018.

[18] V. Romanov and A. Khusainova, "Evaluation of morphological embeddings for English and Russian languages," in *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, (Minneapolis, USA), pp. 77–81, Association for Computational Linguistics, June 2019.

[19] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," 2013.

[20] P. Tungthamthiti, K. Shirai, and M. Mohd, "Recognition of sarcasms in tweets based on concept level sentiment analysis and supervised learning approaches," in *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*, (Phuket,Thailand), pp. 404–413, Department of Linguistics, Chulalongkorn University, Dec. 2014.

[21] L. Freitas, A. Vanin, and R. Vieira, "Irony detection on twitter: a natural language processing investigation," 07 2013.