Stock Embeddings Acquired from News Articles and Price History, and an Application to Portfolio Optimization

Xin Du
Department of Advanced Interdisciplinary Studies, Graduate School of Engineering, The University of Tokyo
duxin@cl.rcast.u-tokyo.ac.jp

Kumiko Tanaka-Ishii
Research Center of Advanced Science and Technology, The University of Tokyo
kumiko@cl.rcast.u-tokyo.ac.jp

Abstract

Previous works that integrated news articles to better process stock prices used a variety of neural networks to predict price movements. The textual and price information were both encoded in the neural network, and it is therefore difficult to apply this approach in situations other than the original framework of the notoriously hard problem of price prediction. In contrast, this paper presents a method to encode the influence of news articles through a vector representation of stocks called a *stock embedding*. The stock embedding is acquired with a deep learning framework using both news articles and price history. Because the embedding takes the operational form of a vector, it is applicable to other financial problems besides price prediction. As one example application, we show the results of portfolio optimization using Reuters & Bloomberg headlines, producing a capital gain 2.8 times larger than that obtained with a baseline method using only stock price data. This suggests that the proposed stock embedding can leverage textual financial semantics to solve financial prediction problems.

1 Introduction

News articles influence the dynamics of financial markets. For example, after the release of breaking news, the share prices of related stocks are often observed to move. This suggests the possibility of using natural language processing (NLP) to aid traders by analyzing this influence between news article texts and prices.

Recent studies (Ding et al., 2015; Hu et al., 2018; Chen et al., 2019; Yang et al., 2018) have indeed reported that news articles can be leveraged to improve the accuracy of predicting stock price movements. These previous works have used deep learning techniques. They train neural networks with article texts and financial market prices, attempting to improve price prediction. In these approaches, the overall mutual effect between texts and prices is distributed over the neural network, which makes it difficult to extract this effect and apply it to tasks other than price prediction.

Therefore, we take a new approach by explicitly describing this mutual effect in terms of a vector. A stock is represented by a vector so that its inner product with an embedding of a text produces a larger value when the text is more related to the stock. In the rest of the paper, we call this vector a *stock embedding*.

The names of stocks, such as "AAPL" (the ticker symbol for *Apple Inc.*), typically appear in a financial news article text. Because these names form part of the text, usual NLP techniques can be applied to acquire an embedding of a stock. Such general textual embedding, however, does not incorporate the financial reality of stock price changes. Hence, the proposed stock embedding represents the price as well as the semantics of the text, as we acquire it by training on both news articles and stock prices. Precisely, our stock embedding is trained through a binary classification problem, namely, whether a stock price goes up or down in comparison with the previous day's price. As a result, an acquired stock embedding captures the relation between a stock name and a news article even when the article has no direct mention of the stock. Our stock embedding can be considered as one technique to specialize, or ground, a symbol that has a practical reality outside of text.

Furthermore, two major advantages come with the vector form of our stock embedding. The first is that the training can be effectuated for all stocks at once, rather than stock by stock. This is an important advantage to alleviate data sparseness and prevent overfitting, as discussed in Section 4.

The second advantage lies in the portability of a vector. In contrast to previous works, in which

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3353–3363

stock-specific information was distributed among the parameters of a neural network, a vector representing all the characteristics of a stock is much easier to extract and apply to other uses besides price prediction.

Hence, this paper shows an example of portfolio optimization, one of the most important applications in finance. To the best of our knowledge, this is the first report of incorporating NLP into *modern portfolio theory* (Markowitz, 1952). Our method differs from previous works that used NLP to enhance investment strategies. Many previous works focused on stock price forecasting only (Ding et al., 2015; Hu et al., 2018) and did not attempt to apply the learned results to other financial tasks. Another previous work (Song et al., 2017) investigated portfolios with texts. It obtained a ranking of stocks from texts by using a neural network technique and then evaluated investment in the highest/lowest ranked stocks. That work was not based on modern portfolio theory, however, nor did it integrate price and text data. In contrast, our method uses NLP in addition to price data to acquire a general representation in the form of an embedding applicable to different targets. In our experiments, a portfolio generated using stock embeddings achieved an annual gain 2.8 times greater than that of a portfolio generated with price data only. This provides evidence that the stock embedding well encodes both text and price information.

2 Related Work

The main idea of this article is based on important techniques of NLP. It is now common to represent discrete entities in natural language by continuous vectors. These vectors are called "embeddings" and usually obtained from neural network models. Examples include the word embedding (Mikolov et al., 2013), phrase embedding (Zhang et al., 2014), sentence embedding (Lin et al., 2017), and event embedding (Ding et al., 2016).

One advantage of these continuous representations is that the geometry of an embedding system contains rich semantic information, as has been discovered at many levels (Mikolov et al., 2013; Reif et al., 2019). The acquisition of stock embeddings in this paper is based on the original idea developed for linguistic entities. Here, we extend the idea further so that the embeddings reflect the reality of a stock market outside text.

A stock embedding is trained using the *attention* mechanism (Bahdanau et al., 2015), which is another current NLP technique. The basic idea of the original attention mechanism is to assign higher weights to more relevant word vectors and make the weights adaptive to different contexts.

Our framework is based on the classification task for text-driven stock price movement, which has been studied intensely as follows. Early research on exploiting financial news articles for better stock price prediction dates back to Ou and Penman (1989), in which financial indicators were extracted manually from financial statements. Later, in Fung et al. (2002), NLP methods were adopted for automatic text feature extraction. Since the 2000s, Twitter and other text-centered social media platforms have become essential sources of financial signals. Bollen et al. (2011) found evidence for causality between the public mood extracted from tweets and the Dow Jones Industrial Average index. In Nguyen et al. (2015), post texts collected from the *Yahoo! Finance Message Board* were used to predict whether the prices of 18 US stocks would rise or drop on the next trading day.

As deep learning methods for NLP have become more common, many papers have reported the use of neural networks for text-driven stock classification (or prediction) tasks. Ding et al. (2015) proposed an event embedding to represent a news headline with a vector and used a convolutional neural network for classification. In that work, all the event embeddings of news articles published on the same day were simply averaged to summarize that day's market information.

Hu et al. (2018) was among the first works that applied the attention mechanism to the task of news-driven stock price movement classification. They developed a dual-level attention framework, in which news articles were assigned different weights depending on the output of a logistic regression component with a bias term, so that the most informative news articles were "highlighted." The method of weighting news articles in this paper is similar to that previous work. The stock-specific information in Hu et al. (2018) was encoded in the neural network, however, making it focused on the price prediction task. In contrast, we represent such stock-specific information by the stock embedding, i.e., a vector, which is easy to interpret geometrically and extract for other applications.

For one such application, we evaluated our stock embedding in terms of portfolio optimization. To

the best of our knowledge, this is the first paper applying NLP techniques to modern portfolio theory. We use the mean-variance minimization portfolio model (introduced in Section 7) proposed in Markowitz (1952), which directly led to the capital asset pricing model (Sharpe, 1964).

3 News-Driven Stock Price Classification

In this paper, the stock embedding is trained with a deep learning system through binary classification of price movements. Let p_t be the stock price on day t, and let y_t be the desired output of the system. Here, $t \in \{1, 2, \ldots, T\}$, and T is the number of trading days in the considered time period. The binary classification problem indicates that y_t is classified in the following way:

$$y_t = \begin{cases} 1, & p_t \geq p_{t-1} \\ 0, & p_t < p_{t-1}. \end{cases} \quad (1)$$

To train such a deep learning system, news articles are used as the input. In this work, news articles are considered daily (i.e., treated in units of days). We denote the set of articles published on day t by N_t, and each article by $n_i \in N_t$, with $i = 1, \ldots, |N_t|$. This paper considers a time window around day t, denoted as $[t - d_1, t + d_2]$ given two constants d_1, d_2. Let $N_{[t-d_1,t+d_2]}$ be the set of news articles published within the time window.

When $d_2 = -1$, indicating the use of articles until day $t - 1$, the task is called *prediction*, as the training does not use any articles published on or after day t. In general, this task is acknowledged as *very* hard (Fama, 1970; Basu, 1977; Timmermann and Granger, 2004) according to the *efficient-market hypothesis* (EMH)[1], and such prediction provides only a limited gain, if any. Note that previous NLP studies concerning stock prices were all aimed at this hard problem (Ding et al., 2015; Hu et al., 2018; Xu and Cohen, 2018; Yang et al., 2018).

On the other hand, when $d_2 \geq 0$, this paper refers to the task as *classification*. The performance on classification shows how well the model *understands* a news article. Because the *prediction* problem is too hard and offers limited gain, as proven by many previous works, our target lies in *classification*. The aims are thus to acquire embeddings that are highly sensitive to textual context and to apply them to tasks other than price prediction. Therefore, in this paper, we set $d_1 = 4$ and $d_2 = 0$.

Let the classification model be represented by a mapping f. The probability that the price of a stock j, where $j = 1, \ldots, J$, goes up on day t is

$$\hat{y}_t^j = f\left(N_{[t-4,t]}\right). \quad (2)$$

In the process of model optimization, the model should reduce the mean cross-entropy loss between every true label y_t^j and its corresponding estimate $\hat{y}_t^j$, as follows:

$$l^j = -\frac{1}{T}\sum_{t=1}^{T}\left(y_t^j \log \hat{y}_t^j + (1 - y_t^j)\log(1 - \hat{y}_t^j)\right).$$

This function describes the loss for only one stock, but a stock market includes multiple stocks. This work considers all stocks in a market equally important. The overall loss function is therefore a simple average of the cross-entropy loss for all stocks, i.e., $l = (\sum_{j=1}^{J} l^j)/J$.

4 Method to Acquire Stock Embeddings

Let s_j represent a stock embedding, where $j = 1, 2, ..., J$. This is initialized as a random vector and then trained via a neural model to obtain s_j, whose inner product with the embedding of a related text becomes large. This section describes the proposed method to acquire stock embeddings by building up a neural network for price movement classification.

The neural network consists of two parts: a text feature distiller and a price movement classifier.

Text feature distiller. The text feature distiller first converts every news article n_i into a pair of vectors (n_i^K, n_i^V) corresponding to "key" and "value" vectors, respectively. Let $N_t^K = \{n_i^K\}_t, N_t^V = \{n_i^V\}_t$ denote the sets of key/value vectors of the articles released on day t. Such dual-vector representation of a text was proposed and adopted successfully in Miller et al. (2016) and Daniluk et al. (2017). The pair of vectors contains the semantic information of the article text at two different levels. Roughly, n_i^K represents the article at the *word* level, whereas n_i^V represents it at the *context* level.

The text feature distiller calculates the attention score for every article i published on day t. The attention score between article i and stock j is given by the inner product of the two vectors n_i^K and s_j:

$$\text{score}_{i,j} = n_i^K \cdot s_j.$$

Note that there are other possible definitions of this inner product, such as the cosine similarity or

[1] According to the EMH, in an "efficient" market, prices reflect the true values of assets by having incorporated all past information, so nobody can predict the price. The EMH is hypothesized to hold but has also attracted criticism.

a generalized inner product using some arbitrary function. Because this work focuses on the most basic capability of the stock embedding, it uses the most basic inner product (i.e., the dot product).

Let α_i^j denote the weight put on news article i with respect to stock j, to classify whether the stock price will go up or down. With the use of $\mathrm{score}_{i,j}$ defined above, α_i^j is given as the following:

$$\alpha_i^j \equiv \frac{\exp(\mathrm{score}_{i,j})}{\sum_{i'} \exp(\mathrm{score}_{i',j})}.$$

α_i^j is thus acquired as the softmax function of the scores across the articles released on the same day.

By using α_i^j as the weights put on news articles, we compute the market status of stock j on day t as the following, which is the input to the classifier:

$$m_t^j = \sum_{n_i^V \in N_t^V} \alpha_i^j n_i^V. \quad (3)$$

Therefore, m_t^j is computed over a set of n_i^V, representing the context of texts on day t. We call m_t^j the *market vector*, to which we will return in Section 6.

Price movement classifier. The input of the price movement classifier is a sequence of vectors, $M_{[t-4,t]}^j = [m_{t-4}^j, m_{t-3}^j, \ldots, m_t^j]$, with respect to stock j. This is processed by a recurrent neural network using a *bidirectional gated recurrent unit* (Bi-GRU). The choice of a Bi-GRU was made by considering the model capacity and training difficulty. The classifier estimates the probability $\hat{y}_t^j$:

$$\begin{aligned} h_t^O &= \mathrm{GRU}(M_{[t-4,t]}^j), \\ \hat{y}_t^j &= \sigma(\mathrm{MLP}(h_t^O)), \end{aligned} \quad (4)$$

where $\sigma(x) = 1/(1+\exp(-x))$, and GRU and MLP stand for the Bi-GRU and a multilayer perceptron, respectively. An optional re-weighting technique over the GRU's output vectors $h_\tau^O (\tau \in [t-4, t])$ (Hu et al., 2018) can be applied. In this case, after the first line of formula (4), the re-weighting is conducted in the following way:

$$h^O = \sum_{\tau=t-4}^{t} \beta_\tau h_\tau^O,$$

and this h^O becomes the input of the second line instead of h_t^O. Here, β_τ, the weight for day τ, decides how much one day is considered in the classification. In our implementation,

$$\beta_\tau = \frac{\exp(v_{\tau-t} \cdot h_\tau^O)}{\sum_{\xi=-4}^{0} \exp(v_\xi \cdot h_{t+\xi}^O)},$$

where the vector v_ξ differentiates the temporal effects of news articles released around day t. v_ξ is initialized randomly and trained via the neural network. See Hu et al. (2018) for the details.

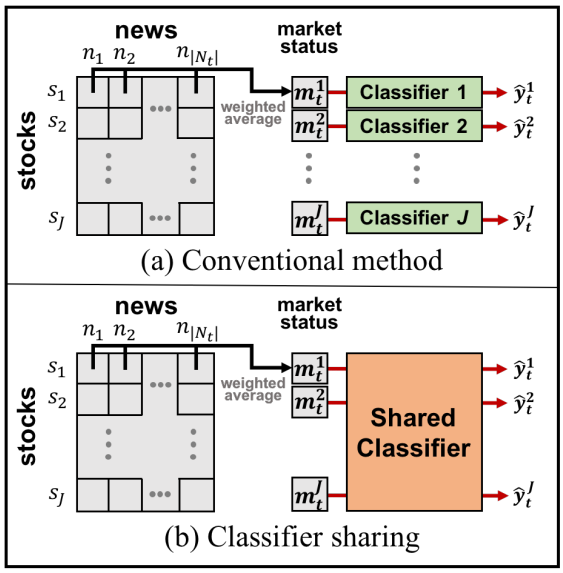


Figure 1: Illustration of the classifier sharing mechanism across stocks on day t: (a) one independent classifier per stock, and (b) a shared classifier across stocks. $|N_t|$ denotes the number of news articles on day t.

Such formulation of neural network training has the advantage of avoiding overfitting. A common problem in the task of stock movement classification or prediction is small sample sizes, especially when adopting units of days. In contrast, the proposed model does not suffer from small sample sizes, because the *price movement classifier* can be trained across all the stocks by sharing one classifier, rather than by generating one classifier for each individual stock like in many previous works (Ding et al., 2015; Hu et al., 2018; Xu and Cohen, 2018). We call this a *classifier sharing* mechanism.

Figure 1 illustrates the difference between models with and without classifier sharing. The upper figure (a) shows the conventional setting without sharing, in which J classifiers are generated, one for each stock. In contrast, the lower figure (b) shows one classifier generated for all stocks. This setting enables learning of the correlation among stocks, in addition to avoiding overfitting and the problem of small sample sizes. Specifically, the classifier is shared across all stocks, thus achieving a sample size about 50 to 100 times larger.

5 Dataset and Settings to Acquire Stock Embeddings

5.1 Dataset

We used two news article datasets to build stock embeddings: the *Wall Street Journal* (WSJ, in the

Dataset	Period	# of articles	# of days / trading days	Mean # of articles per day (std)	Mean # of words per article (std)
Wall Street Journal (WSJ)	2000/01-2015/12	403,207	5,649/4,008	71.4 (41.1)	28.5 (9.0)
Reuters & Bloomberg (R&B)	2006/10-2013/11	551,479	2,605/1,794	211.7 (257.1)	9.34 (1.73)

Table 1: Basic information on the two news article datasets.

following) dataset and the *Reuters & Bloomberg* (R&B) dataset[2], as listed in Table 1. WSJ contains around 400,000 news headlines published across 16 years, whereas R&B contains around 550,000 articles across 7 years. Compared with R&B, WSJ has a relatively more uniform distribution of news articles across time (see the standard deviations listed in parentheses in the fifth column of Table 1). Following previous studies reporting that the main body of a news text produces irrelevant noise (Ding et al., 2015), we extracted only the headlines in both datasets.

As for the stocks, we selected two subsets of the stocks in Standard & Poor's S&P 500 index, one for each of the WSJ and R&B datasets. These subsets consisted only of stocks that were mentioned in no fewer than 100 different news articles, so that mutual effects between the articles and the price history would appear pretty often in the texts. More importantly, this ensured that keyword retrieval-based methods that locate related articles by explicit keyword matching could be applied for comparison. For the WSJ and R&B datasets, the subsets had 89 and 50 stocks, respectively. All other stocks were removed from consideration.

As seen in formula (2), the input for the neural network is $N_{[t-4,t]}$, the set of articles around day t, and the output is y_t^j. The label y_t^j is the binarized price movement of stock j at day t. This is measured by the log-return between two subsequent days:

$$\log \text{return}_t^j = \log p_t^j - \log p_{t-1}^j.$$

The distribution of log-returns is typically bell shaped with a center close to 0, as also mentioned in Hu et al. (2018). The return values of the days were separated into three categories of "negative," "ambiguous," and "positive" by the use of thresholds[3]. Here, "ambiguous" refers to those samples close to 0.0, which were removed. Thus, by using only the clearly negative and positive days, the returns were binarized.

Through such filtering, the number of samples for each stock became about two-thirds of the number of all trading days, or around[4] 2600 and 1200 samples for each stock, for the WSJ and R&B datasets, respectively.

5.2 Deep Learner System Settings

The Adam optimizer (Kingma and Ba, 2015) was used with cosine annealing (Loshchilov and Hutter, 2017) to train the neural network. The initial learning rate was set to 5e-4. The mini-batch size was 64. We stopped the training process when the value of the loss function with respect to the validation set no longer dropped, and then we measured the accuracy on the test set for evaluation.

As for the dual-vector representation of news article texts, introduced in Section 4, the key and value vectors were calculated as described here. The key vector n_i^K is defined as follows[5] by using word embeddings w_k acquired by Word2vec:

$$n_i^K = \frac{\sum_k \gamma_k w_k}{\sum_k \gamma_k},$$

where $\gamma_k = \text{TF}_k \cdot \text{IDF}_k$ is the TFIDF (Manning and Schütze, 2001) score of word k. The dimension of n_i^K equals that of the Word2vec model trained on the news corpus, i.e., 64 in our implementation.

As for the value vector n_i^V, we used vectors acquired through a BERT encoder[6]. We used the pretrained BERT model available from *Google Research*, with 24 layers trained on an uncased corpus. This model outputs vectors of 1024 dimensions, but we reduced the dimensions to 256 by using principal component analysis (PCA), to suppress the number of parameters in the neural network. Along with the effect of the stock embedding, the effect of the *dual-vector representation* (DVR) is also

[2]This dataset was made open source in Ding et al. (2015).

[3]We used the thresholds $[-0.0053, 0.0079]$ for the WSJ dataset and $[-0.0059, 0.0068]$ for the R&B dataset. The margins were asymmetric around 0 because these datasets had slightly more "rising" days than "declining" ones.

[4]The number of samples after filtering differed slightly among stocks, because the distribution of log-returns differed, while the same thresholds were used.

[5]We chose this method after examining several options, including the smooth inverse frequency (SIF) (Arora et al., 2017), TFIDF-weighted word embeddings, and several other methods. We found that TFIDF-weighted word embeddings with Word2vec worked best.

[6]BERT (Bidirectional Encoder Representations from Transformer) is a neural network model (Devlin et al., 2019) that can be used to encode text into vectors with a fixed dimension.

evaluated in the following section.

6 Effect of Stock Embedding on Price Movement Classification

The basic effect of the stock embedding was evaluated through the performance on the price movement classification task, as stated in Section 3.

The whole dataset described in Section 5.1 was randomly divided into nonoverlapping training/validation/test sets in the ratios of 0.6/0.2/0.2. The training/validation/test parts did not share any samples from the same dates. Every method below was tested for 10 different random divisions, and the average performance is reported here.

The proposed model is abbreviated as *WA+CS+DVR*, for *weighted average* with *classifier sharing* and *dual-vector representation*. For an ablation test, four models were considered, which varied the market vector of the day (defined in formula (3) in Section 4 (Ding et al., 2015)) and were with or without the dual-vector representation and classifier sharing (Ding et al., 2015; Hu et al., 2018; Xu and Cohen, 2018; Yang et al., 2018), as follows.

Simple average: The simple average of the text representations of the same day is taken as the market vector of the day, as proposed by Ding et al. (2015).

Weighted average (WA): As stated in formula (3), the market vector of the day is averaged by using the weights from the stock-text inner products, as proposed in Hu et al. (2018). Note again that their work did not apply classifier sharing but instead produced one classifier for each stock, nor did it adopt the dual-vector representation.

WA + classifier sharing (CS): This refers to *WA* with classifier sharing across stocks. This variant does not adopt the dual-vector representation, i.e., n_i^K is set equal to n_i^V for every news article i. Thus, the same BERT text embedding is used for both n_i^K and n_i^V.

WA + dual-vector representation (DVR): This refers to *WA* with the dual-vector representation of news texts. This variant does not adopt classifier sharing.

Furthermore, to examine the effect of the data size, we tested different dataset portions: 1 year, 3 years, and the whole dataset. Therefore, the experimental variants involved five methods (four comparison + our proposal) and three data sizes, or a total of 15 experiments.

Figure 2 summarizes the complete experimental results. The uppermost bar of each bar group, in red, corresponds to our model with classifier sharing (CS) and the dual-vector representation (DVR). The other bars, in orange, blue, purple, and green, correspond to the four ablation variants. The ablation datasets with only 1-year data contained around 150 training samples and were too small for most variants to work well, yet our proposed model, *WA+CS+DVR*, could still obtain positive results (classification accuracy over 50%). With the 3-year datasets, our *WA+CS+DVR* model widened the performance gap, whereas the *simple average* and *weighted average* models still failed to work better than random guessing. These results show the superiority of our model in handling the overfitting problem with small datasets.

Finally, the significant differences between *WA+CS+DVR* (in red) and *WA+CS* (in blue) and between *WA+DVR* (in orange) and *WA* (in purple) strongly supported the advantage of adopting the dual-vector representation (DVR), especially when classifier sharing was combined.

Overall, our model successfully achieved 68.8% accuracy for the R&B dataset, which was significantly better than any of the other four variants.

7 Portfolio Optimization

Thus far, the evaluation on *classification* has shown the capability of our framework in understanding news articles. For financial applications, however, the task must be in the form of *prediction*; that is, it must produce some gain ahead of the time when a news article is published. As one such predictive example, we present portfolio optimization, one of the most important financial tasks, and we show how our stock embedding can be applied to it.

A portfolio is essentially a set of weights assigned to stocks, representing the proportions of capital invested in them. Intuitively, a portfolio bears a bigger risk if a large proportion is invested in two highly positively correlated stocks, rather than two uncorrelated or negatively correlated stocks. Based on this idea, the mean-variance minimization model in Markowitz (1952) is formulated as follows:

$$\min_{w} \quad \text{risk} = w^T \Sigma w \tag{5a}$$

$$\text{subject to} \quad w^T r = E, \tag{5b}$$

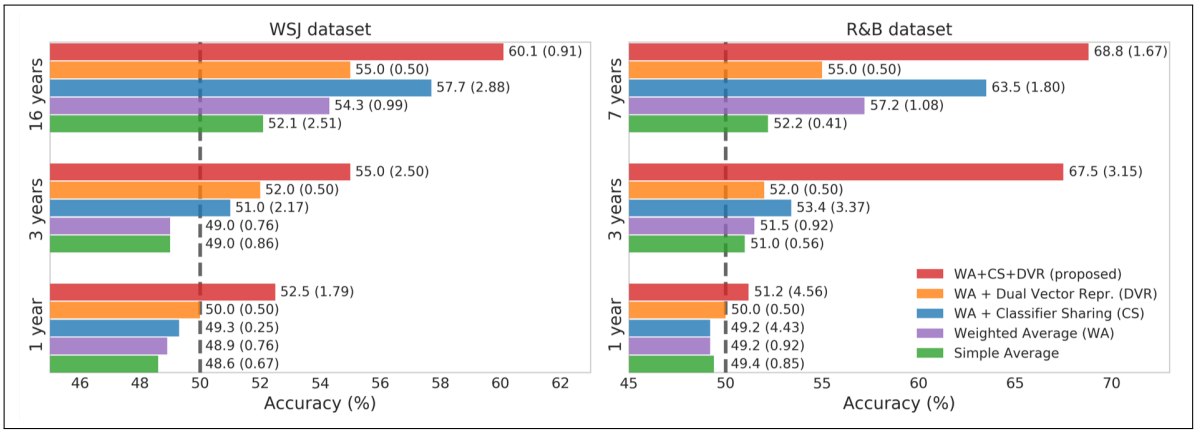


Figure 2: Mean classification accuracy percentages (with SD in parentheses) over 10 replications.

$$w^T \mathbf{1} = 1, \quad (5c)$$

$$0 \leq w_j \leq 1 \quad j = 1, ..., J, \quad (5d)$$

where Σ is the risk matrix; w is the vector of investment weights; r is a vector such that r_j equals the mean historic return of stock j; $\mathbf{1}$ is a vector of ones; and E, the expected portfolio return, is a parameter decided by an investor's preference. Note that higher E usually means higher risk born by the investor.

In the original model of Markowitz, Σ is the covariance matrix of the historic return time series of stocks, $\Sigma_{ij} = \mathrm{Cov}(\{r_i\}_t, \{r_j\}_t)$ $(i, j \in \{1, ..., J\})$. According to Markowitz (1952), the solution of this optimization problem, which can be obtained via quadratic programming, gives the portfolio with the smallest risk for an expected overall return E.

Using the covariance matrix as the risk matrix Σ is limited, however, for two reasons. First, the overwhelming noise in price movements prevents accurate estimation of the covariance. More importantly, it ignores the events described in news articles that indeed cause price movements.

On the other hand, the stock embeddings built here provide much abundant textual information for defining Σ. Concretely,

$$\Sigma_{i,j} = \cos(s_i, s_j).$$

This should work because the stock embedding reflects a stock's responsiveness to a certain class of news events. In other words, close stock embeddings indicate a correlated response pattern to an event described in news articles. Stock embeddings capture this *correlation* much better than the covariance matrix does, and this correlation is what a good portfolio relies on.

By solving the same optimization problem but with a different matrix Σ, we get another vector of investment ratios, w, with respect to the stocks. By virtually investing according to w and observing the result within a certain period, Σ can be evaluated. For each of the WSJ and R&B datasets, we ran one investment simulation for various definitions of Σ, as follows.

S&P 500 index: As a market baseline, we used an S&P 500 index portfolio, in which all 505 stocks in the index were considered and the investment weight w_j was in proportion to the market capitalization of stock j. The price history of the portfolio was provided by *Dow Jones*. This method did not use Σ to form the portfolio.

S&P 89*/50*: This approach was the same as above but with the set of stocks reduced to those tested in our work, as explained in Section 5.1: 89 stocks for the WSJ dataset[7], and 50 for the R&B dataset.

Covariance matrix of historic stock returns: Σ was the covariance matrix as originally proposed by Markowitz.

Word2vec-general: (text only) Σ was the cosine matrix of the word embeddings trained on general corpora (fastText word embeddings (Bojanowski et al., 2017) were used in our experiments). For each stock, we used the word embedding of its ticker symbol, e.g., the word embedding of "AAPL" for *Apple Inc.*

Word2vec-news: (text only) Σ was the cosine matrix of the word embedding vectors trained

[7]The S&P 89* portfolio was evaluated during the period of 2001 to 2016. The market capitalization history of the stocks before the year 2005 is not available, so the record was estimated for this missing period. First, the number of *shares outstanding* was extrapolated from the data of 2005-2016, in which the values were pretty stable during the whole period. The market capitalization was then acquired by multiplying the price by the shares outstanding.

on news text corpora. We used the full text of the R&B dataset for training, in which all mentions of a stock in the text were replaced by the stock's ticker symbol.

Covariance · stock embedding: (text and price) Σ was the result of element-wise multiplication of the covariance matrix and the cosine matrix of the stock embeddings.

Weighted BERT: (text only) Σ was the cosine matrix of stock vectors acquired as follows, where the BERT-based text representation n_i^V was used. For a stock j, the vector was obtained as a weighted average of n_i^V for which the text mentioned the stock or company. Here, the weight of article i was defined as follows:

$$\eta_i \equiv \frac{(\#\text{ of mentions of } j \text{ in } i)}{(\#\text{ of mentions of all stocks in } i)}.$$

Stock embeddings: Σ was the cosine matrix of the stock embeddings.

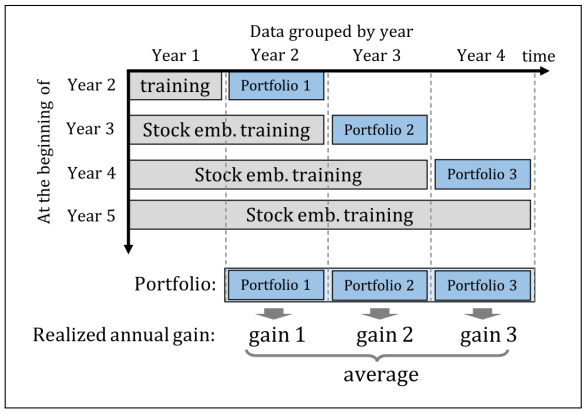


Figure 3: The process of portfolio generation and evaluation over several years. The vertical axis indicates the times when the portfolio is renewed, and the horizontal axis indicates the data grouped yearly. An average of the realized annual gains is computed to evaluate the portfolio's performance.

The portfolio evaluation was conducted in a yearly setting, as illustrated in Figure 3. At the beginning of each year, given some expected gain E, the portfolio was computed by using all news articles and historic prices *until* the end of the previous year. In other words, for each year, the training set in the experiment consisted of samples *strictly earlier* than those constituting the test set. Therefore, the evaluation was conducted in a *prediction* setting. Then, investments were made according to the yearly renewed portfolio as in Figure 3; that is, capital was allocated to stocks according to w. The realized annual gain of the portfolio followed this equation:

$$\text{annual gain} = \sum_{j=1}^{J} w_j \left(\frac{p^j_{\text{end-of-year}}}{p^j_{\text{begin-of-year}}} - 1\right),$$

where w_j is the proportion of investment in stock j, and p^j is the price of j.

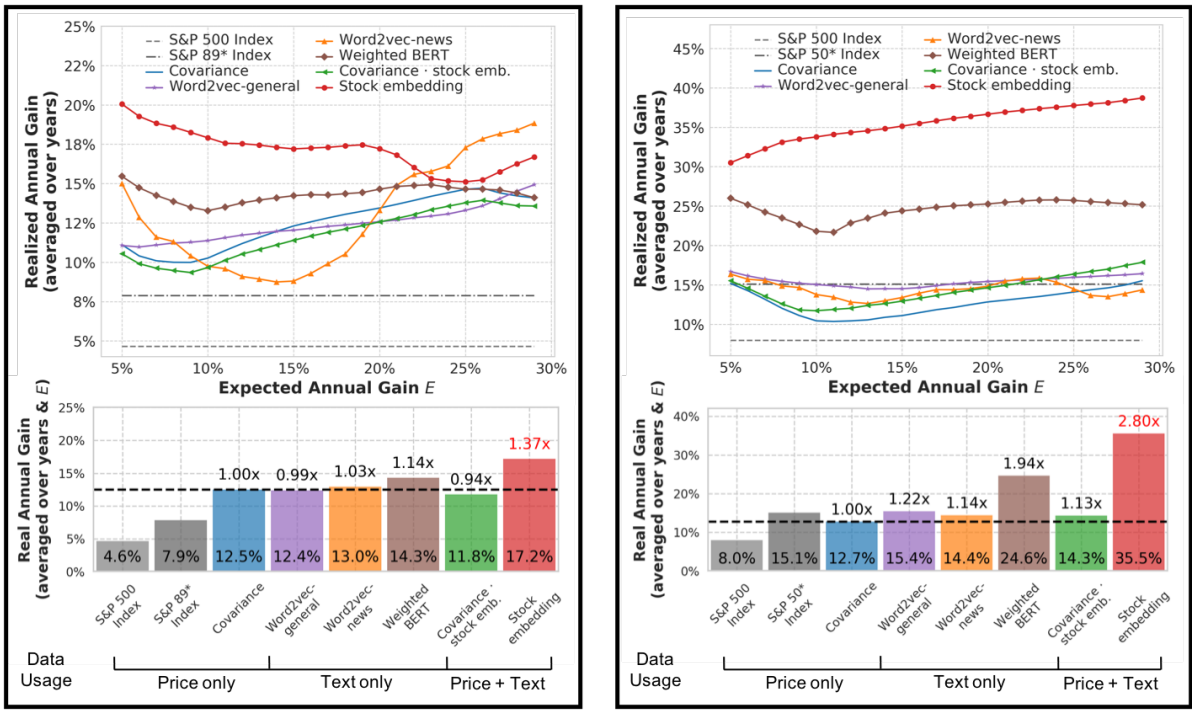
In this way, for each of the WSJ and R&B, we obtained results over 16 and 7 years, respectively. For different expected gains $E \in \{0.05, 0.06, ..., 0.29\}$, which cover typical cases in real-world portfolio construction, the average annual gain was computed.

Figure 4 shows the experimental results. The upper graphs show the annual gain with respect to different values of E (horizontal axes) for (a) the WSJ and (b) the R&B, averaged over years. Every curve corresponds to a different definition of Σ. It can be seen that the proposed stock embedding method outperformed the other methods, except for larger E with WSJ[8]. Especially for the R&B dataset, stock embedding greatly outperformed all other methods at all E.

The lower bar graph summarizes the overall aggregate gain for each method. The values in the bars indicate the average realized annual gains, while those above the bars are the ratios of the gains in comparison with that of the standard covariance method (in blue). The leftmost two bars in each bar graph show the gains of the S&P 500 portfolio and the S&P 89*/50* portfolio, respectively. As described above, the S&P 500 portfolio consisted of an average of around 500 stocks traded in the US, while the S&P 89*/50* portfolio, which was calculated with the same method but on a smaller set of stocks (89 for the WSJ, and 50 for the R&B), achieved higher gains than its S&P 500 sibling did. The values of the S&P portfolios generally went up during the periods of both datasets, and therefore, the gains were positive.

The dashed horizontal line in each bar graph indicates the result for the standard covariance method as a baseline. Its gains were only 12.5% and 12.7% for the WSJ and R&B, respectively, but with stock embeddings, the gains increased to 17.2% and 35.5%, or 1.37 and 2.80 times greater than the baseline results, respectively. This per-

[8]Our method did not perform well only for large E. The mean-variance minimization model has been reported to become *unstable* under the two conditions of large E and low overall market gain (Dai and Wang, 2019). The return of the WSJ period (2000-2015) was lower than that of the R&B period (2006-2013), and therefore, these two conditions were more likely to be met for WSJ.



(a) (b)

Figure 4: Expected and realized annual portfolio gain in the investment simulations on both datasets: (a) results on the WSJ dataset (2000-2015), and (b) results on the R&B dataset (2006-2013).

formance largely beat all other variants and gives evidence of how well the stock embeddings integrated both price and textual information.

The results for the method that integrated the covariance matrix and stock embedding (in green) did not much outperform the baselines. A possible reason is that the stock embedding had already integrated the price information. As for the other variants based on pure text (in purple, orange, and brown), the results improved slightly. Among them, weighted BERT outperformed the other methods for both datasets. This indicates the potential of BERT and other recent neural language models for portfolio optimization.

8 Conclusion

This paper has proposed the idea of a *stock embedding*, a vector representation of a stock in a financial market. A method was formulated to acquire such vectors from stock price history and news articles by using a neural network framework. In the framework, the stock embedding detects news articles that are related to the stock, which is the essence of the proposed method. We trained stock embeddings for the task of binary classification of stock price movements on two different datasets, the WSJ and R&B. The improvements in classification accuracy with our framework, due to the classifier sharing and dual-vector text representation proposed in this paper, implied that the stock embeddings successfully incorporated market knowledge from both the news articles and price history.

Because the stock embedding is a vector that can be separated from the other components of the classification model, it can be applied to other tasks besides price movement classification. As an example, we showed the use of stock embeddings in a portfolio optimization task by replacing the risk matrix in the portfolio objective function with a cosine matrix of stock embeddings. In investment simulations on the R&B dataset, our stock embedding method generated 2.80 times the annual return obtained using the covariance matrix of the historic return series. This significant gain suggests further potential of our stock embedding for modeling the correlation among stocks in a financial market, and for further applications, such as risk control and asset pricing.

Acknowledgments

We sincerely thank the anonymous reviewers for their comments. This paper was supported by the Research Institute of Science and Technology for Society (HITE 17942497), and by the University of Tokyo Gap Fund Program. The paper reflects the view of the authors only.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sanjoy Basu. 1977. Investment performance of common stocks in relation to their price-earnings ratios: A test of the efficient market hypothesis. *The journal of Finance*, 32(3):663–682.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Johan Bollen, Huina Mao, and Xiao-Jun Zeng. 2011. Twitter mood predicts the stock market. *J. Comput. Science*, 2(1):1–8.

Deli Chen, Yanyan Zou, Keiko Harimoto, Ruihan Bao, Xuancheng Ren, and Xu Sun. 2019. Incorporating fine-grained events in stock movement prediction. In *Proceedings of the Second Workshop on Economics and Natural Language Processing*, pages 31–40, Hong Kong. Association for Computational Linguistics.

Zhifeng Dai and Fei Wang. 2019. Sparse and robust mean–variance portfolio optimization problems. *Physica A: Statistical Mechanics and its Applications*, 523:1371 – 1378.

Michal Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2327–2333.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2016. Knowledge-driven event embedding for stock prediction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2133–2142, Osaka, Japan. The COLING 2016 Organizing Committee.

Eugene F. Fama. 1970. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Wai Lam. 2002. News sensitive stock trend prediction. In *Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference, PAKDD 2002, Taipei, Taiwan, May 6-8, 2002, Proceedings*, pages 481–493.

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 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 261–269.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Christopher D. Manning and Hinrich Schütze. 2001. *Foundations of statistical natural language processing*. MIT Press.

Harry Markowitz. 1952. Portfolio selection. *The journal of finance*, 7(1):77–91.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Thien Hai Nguyen, Kiyoaki Shirai, and Julien Velcin. 2015. Sentiment analysis on social media for stock movement prediction. *Expert Syst. Appl.*, 42(24):9603–9611.

Jane A Ou and Stephen H Penman. 1989. Financial statement analysis and the prediction of stock returns. *Journal of accounting and economics*, 11(4):295–329.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. In *Advances in Neural Information Processing Systems 32*, pages 8592–8600. Curran Associates, Inc.

William F Sharpe. 1964. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442.

Qiang Song, Anqi Liu, and Steve Y. Yang. 2017. Stock portfolio selection using learning-to-rank algorithms with news sentiment. *Neurocomputing*, 264:20 – 28. Machine learning in finance.

Allan Timmermann and Clive WJ Granger. 2004. Efficient market hypothesis and forecasting. *International Journal of forecasting*, 20(1):15–27.

Yumo Xu and Shay B. Cohen. 2018. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1979, Melbourne, Australia. Association for Computational Linguistics.

Linyi Yang, Zheng Zhang, Su Xiong, Lirui Wei, James Ng, Lina Xu, and Ruihai Dong. 2018. Explainable text-driven neural network for stock prediction. In *5th IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2018, Nanjing, China, November 23-25, 2018*, pages 441–445.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 111–121, Baltimore, Maryland. Association for Computational Linguistics.