Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Deep Learning for Event-Driven Stock Prediction

Xiao Ding[†][*], **Yue Zhang**[‡], **Ting Liu**[†], **Junwen Duan**[†]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{xding, tliu, jwduan}@ir.hit.edu.cn
[‡]Singapore University of Technology and Design
yue_zhang@sutd.edu.sg

Abstract

We propose a deep learning method for event-driven stock market prediction. First, events are extracted from news text, and represented as dense vectors, trained using a novel neural tensor network. Second, a deep convolutional neural network is used to model both short-term and long-term influences of events on stock price movements. Experimental results show that our model can achieve nearly 6% improvements on S&P 500 index prediction and individual stock prediction, respectively, compared to state-of-the-art baseline methods. In addition, market simulation results show that our system is more capable of making profits than previously reported systems trained on S&P 500 stock historical data.

1 Introduction

It has been shown that the financial market is "informationally efficient" [Fama, 1965] — stock prices reflect all known information, and the price movement is in response to news or events. As web information grows, recent work has applied Natural Language Processing (NLP) techniques to explore financial news for predicting market volatility.

Pioneering work mainly uses simple features from news documents, such as bags-of-words, noun phrases, and named entities [Kogan *et al.*, 2009; Schumaker and Chen, 2009]. Although useful, these features do not capture structured relations, which limits their potentials. For example, representing the event "Microsoft sues Barnes & Noble." using term-level features {"Microsoft", "sues", "Barnes", "Noble"} alone, it can be difficult to accurately predict the price movements of *Microsoft Inc.* and *Barnes & Noble Inc.*, respectively, as the unstructured terms cannot differentiate the accuser ("Microsoft") and defendant ("Barnes & Noble").

Recent advances in computing power and NLP technology enables more accurate models of events with structures. Using open information extraction (Open IE) to obtain structured events representations, we find that the actor and object

[*]This work was done while the first author was visiting Singapore University of Technology and Design

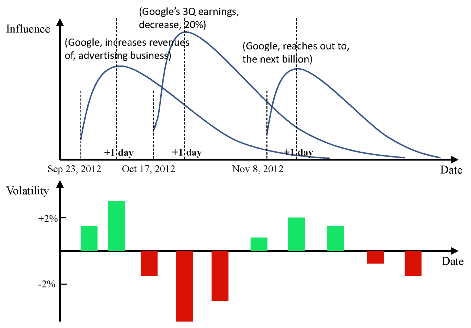


Figure 1: Example news influence of *Google Inc.*

of events can be better captured [Ding *et al.*, 2014]. For example, a structured representation of the event above can be (Actor = *Microsoft*, Action = *sues*, Object = *Barnes & Noble*). They report improvements on stock market prediction using their structured representation instead of words as features.

One disadvantage of structured representations of events is that they lead to increased sparsity, which potentially limits the predictive power. We propose to address this issue by representing structured events using *event embeddings*, which are dense vectors. Embeddings are trained such that similar events, such as (Actor = *Nvidia fourth quarter results*, Action = *miss*, Object = *views*) and (Actor = *Delta profit*, Action = *didn't reach*, Object = *estimates*), have similar vectors, even if they do not share common words. In theory, embeddings are appropriate for achieving good results with a density estimator (e.g. convolutional neural network), which can misbehave in high dimensions [Bengio *et al.*, 2005]. We train event embeddings using a novel neural tensor network (NTN), which can learn the semantic compositionality over event arguments by combining them multiplicatively instead of only implicitly, as with standard neural networks.

For the predictive model, we propose to use deep learning [Bengio, 2009] to capture the influence of news events over a history that is longer than a day. Research shows diminishing effects of reported events on stock market volatility. For example, Xie et al. [2013], Tetlock et al. [2008] and Ding et al. [2014] show that the performance of daily prediction is better than weekly and monthly prediction. As shown in Figure 1, the influences of three actual events for Google

Inc. in the year 2012 was the highest on the second day, but gradually weakened over time. Despite the relatively weaker effects of long-term events, the volatility of stock markets is still affected by them. However, little previous work quantitatively models combined short-term and long-term effects of events. To fill in this gap, we treat history news as daily event sequences, using a convolutional neural network (CNN) to perform semantic composition over the input event sequence, and a pooling layer to extract the most representative global features. Then a feedforward neural network is used to associate the global features with stock trends through a shared hidden layer and a output layer.

Experiments on large-scale financial news datasets from Reuters and Bloomberg show that event embeddings can effectively address the problem of event sparsity. In addition, the CNN model gives significant improvement by using longer-term event history. The accuracies of both S&P 500 index prediction and individual stock prediction by our approach outperform state-of-the-art baseline methods by nearly 6%. Market simulation shows that our model is more capable of making profits compared to previous methods. To our knowledge, we are the first to use a deep learning model for event-driven stock market prediction, which gives the best reported results in the literature.

2 Neural Tensor Network for Learning Event Embeddings

2.1 Event Representation and Extraction

We follow our previous work [Ding *et al.*, 2014] and represent an event as a tuple $E = (O_1, P, O_2, T)$, where P is the action, O_1 is the actor and O_2 is the object on which the action is performed. T is the timestamp of the event, which is mainly used for aligning stock data with news data, and not for event embeddings. For example, the event "Jan 13, 2014 - Google Acquires Smart Thermostat Maker Nest For for $3.2 billion." is modeled as: (Actor = *Google*, Action = *acquires*, Object = *Nest*, Time = *Jan 13, 2014*).

We extract structured events from free text using Open IE technology and dependency parsing. Given a sentence obtained from news text, we first use ReVerb [Fader *et al.*, 2011] to extract the candidate tuples of the event (O'_1, P', O'_2), and then parse the sentence with ZPar [Zhang and Clark, 2011] to extract the subject, object and predicate. We assume that O'_1, O'_2, and P' should contain the subject, object, and predicate, respectively. If this is not the case, the candidate tuple is filtered out. Redundancy in large news data allows this method to capture major events with high recalls.

2.2 Event Embedding

Events are extremely sparse. Our previous work used back-off features (e.g. (O_1, P), (P, O_2), O_1, P, O_2) to address this issue, and we generalized verbs into verb classes, so that similar actions become one feature [Ding *et al.*, 2014]. In contrast, our goal is to automatically learn embeddings for structured event tuples $E = (O_1, P, O_2)$, which draw more fundamental relations between events, even if they do not share the same action, actor or object.

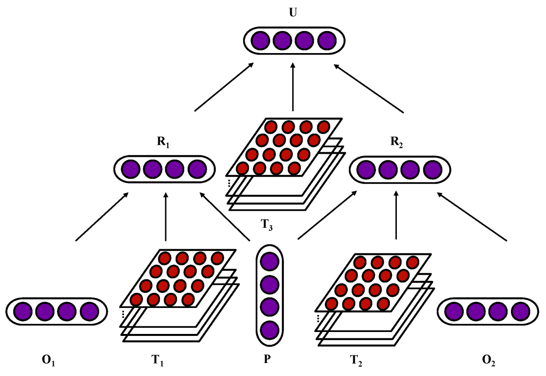


Figure 2: Neural tensor network for event embeddings.

Our task is related to previous work on learning distributed representations of multi-relational data from knowledge bases [Bordes *et al.*, 2011; Socher *et al.*, 2013], which learns the embedding of (e_1, R, e_2), where e_1 and e_2 are named entities and R is the relation type. However, learning structured event embedding has two significant differences.

First, the number of relation types in knowledge bases is limited. Hence, most previous work models a relation type by using a matrix or a tensor, and train a model for each specific relation type. However, as introduced in the previous section, we extract events based on Open IE technology, and the event types is an open set. Therefore, it is more difficult to train a specific model for each event type. To address this issue, we represent the action P as a vector, which shares the dimensionality with event arguments.

Second, the goal of relational database embedding is to be able to state whether two entities (e_1, e_2) are in a certain relation R. When R is symmetric, e_1 and e_2 have interchangeable roles. In contrast, each argument of the event has a specific role, which is not interchangeable. To address this difference, we design a novel neural tensor network to embed structured events, in which the role of argument is explicitly modeled. As illustrated in Figure 2, two tensors, T_1 and T_2, are used to model the roles of O_1 and O_2, respectively. $O_1 T_1 P$ and $P T_2 O_2$ are used to construct two role-dependent embeddings R_1 and R_2, respectively. A third tensor, T_3, is used for semantic compositionality over R_1 and R_2, and generate a complete structured embedding U for $E = (O_1, P, O_2)$.

Neural Tensor Network

The input of neural tensor network is word embeddings and the output is event embeddings. We learn the initial word representation of d-dimensions ($d = 100$) from large-scale financial news corpus, using the skip-gram algorithm [Mikolov *et al.*, 2013]. As most event arguments consist of several words, we represent the actor, action and object as the average of its word embeddings, respectively, allowing the sharing of statistical strength between the words describing each component (e.g. *Nokia's mobile phone business* and *Nokia*).

From Figure 2, $R_1 \in \mathbb{R}^d$ is computed by:

$$R_1 = f(O_1^T T_1^{[1:k]} P + W \begin{bmatrix} O_1 \\ P \end{bmatrix} + b) \tag{1}$$

Algorithm 1: Event Embedding Training Process

```
Input: $\mathcal{E} = (E_1, E_2, \cdots, E_n)$ a set of event tuples; the
       model $EELM$
Output: updated model $EELM'$
1 random replace the event argument and got the corrupted
  event tuple
2 $\mathcal{E}^r \leftarrow (E_1^r, E_2^r, \cdots, E_n^r)$
3 while $\mathcal{E} \neq [\,]$ do
4     $loss \leftarrow max(0, 1 - f(E_i) + f(E_i^r) + \lambda\|\mathbf{\Phi}\|_2^2$
5     if $loss\ > 0$ then
6         $Update(\mathbf{\Phi})$
7     else
8         $\mathcal{E} \leftarrow \mathcal{E}/\{E_i\}$
9 return $EELM$
```

where $T_1^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor, and the bilinear tensor product $O_1^T T_1^{[1:k]} P$ is a vector $r \in \mathbb{R}^k$, where each entry is computed by one slice of the tensor ($r_i = O_1^T T_1^{[i]} P, i = 1, \cdots, k$). The other parameters are a standard feed-forward neural network, where $W \in \mathbb{R}^{k \times 2d}$ is the weight matrix, $b \in \mathbb{R}^k$ is the bias vector, and $f = tanh$ is the activation function. R_2 and U are computed in the same way as R_1.

We also experiment with randomly initialized word vectors as the input of NTN, which is commonly used in related work on structured embedding [Bordes *et al.*, 2011; Jenatton *et al.*, 2012]. In our case, pre-trained word embeddings give slightly better results than randomly initialized embeddings.

Training

We extract more than 10 million events from Reuters financial news and Bloomberg financial news as the training data for event embeddings. The training algorithm repeats for N iterations over the training examples, which is a set of event tuples $E = (O_1, P, O_2)$, extracted from the training corpus using the method in Section 2.1. In each iteration, the training procedure is shown in Algorithm 1.

We assume that event tuples in the training set should be given a higher score than corrupted tuples, in which one of the event arguments is replaced with a random argument. The corrupted event tuple is denoted as $E^r = (O_1^r, P, O_2)$. Specifically, we replace each word in O_1 with a random word w^r in our dictionary $\mathcal{D}$ (it contains all the words in the training data) and derive a corrupted O_1^r. We calculate the *margin loss* of the above two event tuples as:

$$loss(E, E^r) = \max(0, 1 - f(E) + f(E^r)) + \lambda\|\Phi\|_2^2, \quad (2)$$

where $\Phi = (T_1, T_2, T_3, W, b)$ is the set of parameters. The standard L_2 regularization weight λ is set as 0.0001. If the loss $loss(E, E^r) = \max(0, 1 - f(E) + f(E^r))$ is equal to zero, the algorithm continues to process the next event tuple. Otherwise, the parameters are updated to minimize the loss using the standard back-propagation (BP) algorithm. The iteration number N is set to 500.

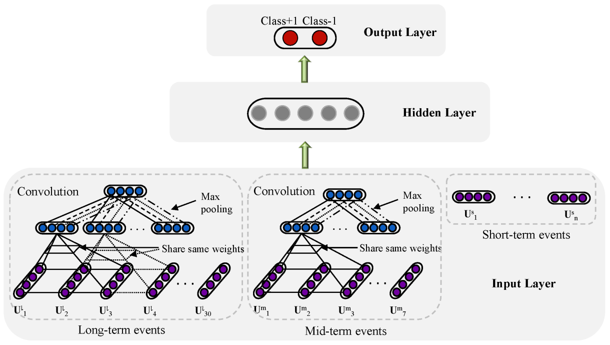


Figure 3: Architecture of the prediction model based on a deep convolutional neural network.

3 Deep Prediction Model

We model long-term events as events over the past month, mid-term events as events over the past week, and short-term events as events on the past day of the stock price change. As shown in Figure 3, the prediction model learns the effect of these three different time spans on stock prices based on the framework of a CNN.

The input to the model is a sequence of event embeddings, where events are arranged in chronological order. Embeddings of the events on each day are averaged as a single input unit (U). The output of the model is a binary class, where Class +1 represents that the stock price will increase, and Class -1 represents that the stock price will decrease. As shown in Figure 3, for long-term (left) and mid-term (middle) news, the narrow convolution operation is used to combine l ($l = 3$) neighbour events. It can be viewed as feature extraction based on sliding window, which can capture local information through combinations of vectors in a window.

For our task, it is necessary to utilize all local features and predict stock price movements globally. Hence, we use a max pooling layer on top of the convolutional layer, which forces the network to retain only the most useful local features produced by the convolutional layer. Note that the convolution operation is only applied to the long-term and mid-term event embeddings, because the unit of timing is one day.

Formally, given a series of input event embeddings $\mathcal{U} = (U_1, U_2, \cdots, U_n)$, where $U_i \in \mathbb{R}^d$, a one-dimensional convolution function takes the dot product of the weight vector $W_1 \in \mathbb{R}^l$ with each l events (sliding window) in $\mathcal{U}$ to obtain a new sequence Q:

$$Q_j = W_1^T U_{j-l+1:j} \quad (3)$$

To determine the most representative features globally, we perform a max pooling operation over Q.

$$V_j = \max Q(j, \cdot), \quad (4)$$

where $Q(j, \cdot)$ is the j-th row of matrix Q. After max pooling, we obtain the feature vector V. For long-term and mid-term events, we obtain the feature vector V^l and V^m, respectively.

	Training	Development	Test
#documents	442,933	110,733	110,733
#words	333,287,477	83,247,132	83,321,869
#events	295,791	34,868	35,603
time interval	02/10/2006 - 18/06/2012	19/06/2012 - 21/02/2013	22/02/2013 - 21/11/2013

Table 1: Statistics of datasets.

For short-term events, we obtain the feature vector V^s by directly using its averaged event embeddings U^s. The feature layer is the combination of long-term, mid-term and short-term feature vectors $V^C = (V^l, V^m, V^s)$.

To correlate the feature vector V^C and stock prices, we use a feedforward neural network with one hidden layer and one output layer. Formally, let the values of the output layer be y_{cls} $(cls \in \{+1, -1\})$, its input be net_{cls}, and Y be the neuron vector of the hidden layer; then: $y_{cls} = f(net_{cls}) = \sigma(W_3^T \cdot Y)$ and $Y = \sigma(W_2^T \cdot V^C)$, where σ is the sigmoid function, W_2 is the weight vector between the hidden layer and the feature layer, and W_3 is the weight vector between the neuron cls of the output layer and the hidden layer.

4 Experiments

4.1 Experimental Settings

We use financial news from Reuters and Bloomberg over the period from October 2006 to November 2013, released by Ding et al. [2014][1]. Randinsky et al. [2012] and Ding et al. [2014] show that news titles are more useful for prediction compared to news contents. This paper extracts events only from news titles. We conduct our experiments on predicting the Standard & Poor's 500 stock (S&P 500) index and its individual stocks, obtaining indices and prices from Yahoo Finance. Detail statistics of training, development (tuning) and test sets are shown in Table 1.

Following Das and Chen [2007] and Xie et al. [2013], the standard measure of accuracy (Acc) and Matthews Correlation Cofficient (MCC) are used to evaluate S&P 500 index prediction and individual stock prediction. Following Lavrenko et al. [2000], we also evaluate the profitability of our proposed model.

4.2 Baselines and Proposed Models

The baselines are two state-of-the-art financial-news-based stock market prediction systems: Luss and d'Aspremont et al. [2012] propose using bags-of-words to represent news documents, and constructing the prediction model by using Support Vector Machines (SVMs). Ding et al. [2014] report a system that uses structured event tuples $E = (O_1, P, O_2)$ to represent news documents, and investigates the complex hidden relationships between events and stock price movements by using a standard feedforward neural network.

In contrast to the baselines, we use a neural tensor network to learn event embeddings for representing news documents, and build a prediction model based on a deep CNN. To make detailed analysis, we construct the following five models:

[1]http://ir.hit.edu.cn/∼xding/index_english.htm

	Acc	MCC
Luss and d'Aspremont [2012]	56.42%	0.0711
Ding et al. [2014] (E-NN)	58.94%	0.1649
WB-NN	60.25%	0.1958
WB-CNN	61.73%	0.2147
E-CNN	61.45%	0.2036
EB-NN	62.84%	0.3472
EB-CNN	**65.08%**	**0.4357**

Table 2: Development results of index prediction.

- **WB-NN**: word embeddings input and standard neural network prediction model [Ding *et al.*, 2014];
- **WB-CNN**: word embeddings input and convolutional neural network prediction model (this paper);
- **E-CNN**: structured events tuple [Ding *et al.*, 2014] input and convolutional neural network prediction model (this paper);
- **EB-NN**: event embeddings input (this paper) and standard neural network prediction model [Ding *et al.*, 2014];
- **EB-CNN**: event embeddings input and convolutional neural network prediction model (this paper).

A word embedding input (WB) consists of the sum of each word in a document; it addresses sparsity in word-based inputs, and can serve as a baseline embedding method. The standard feedforward neural network (NN) is used as a baseline to compare with the deep CNN.

4.3 Development Results

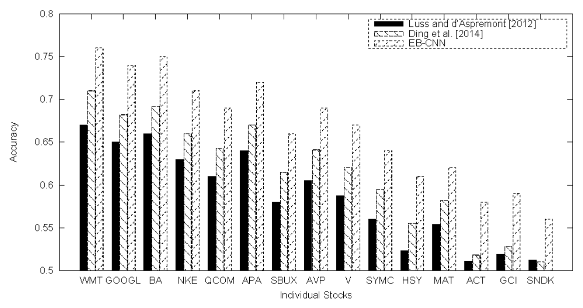
S&P 500 Index Prediction

We test the influence of event embeddings by comparing them with the structured event representation [Ding *et al.*, 2014], and the CNN model by comparison with the standard feedforward neural network model [Ding *et al.*, 2014]. The experimental results are shown in Table 2. We find that:

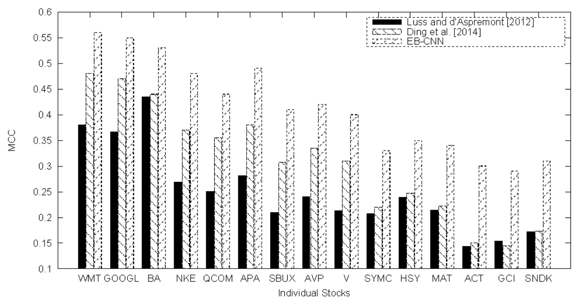
(1) Comparison between the word-embedding-based models and event-embedding-based models (e.g. WB-NN vs EB-NN and WB-CNN vs EB-CNN) confirms our previous conclusion [Ding *et al.*, 2014]: events are better features than words for stock market prediction.

(2) Event embedding is useful for the task of stock market prediction. Given the same prediction model (CNN or NN), the event embeddings based methods (EB-NN and EB-CNN) achieve consistently better performance than the events-based methods (E-NN and E-CNN). This is likely due to the following reasons. First, low-dimensional dense vector can effectively alleviate the problem of feature sparsity. In fact, using word embeddings of events (WB-NN) only, we can achieve better performance than Ding et al. [2014] (E-NN). This contrast demonstrates the importance of reducing sparsity, which rivals the effect of structured information.

Second, we can learn deeper semantic relations between event embeddings, by modeling the semantic compositionality over word embeddings. For example, the two events $E_1 =$ (Actor = *Nvidia fourth quarter results*, Action = *miss*, Object = *views*) in the training data and $E_2 =$(Actor = *Delta profit*,



(a) Accuarcy



(b) MCC

Figure 4: Development results of individual stock prediction (companies are named by their ticker symbols).

	Average Profit
Luss and d'Aspremont [2012]	\$8,694
Ding et al. [2014]	\$10,456
EB-CNN	\$16,785

Table 3: Averaged profit of 15 individual companies.

	Index Prediction		Individual Stock Prediction		
	Acc	MCC	Acc	MCC	Profit
Luss [2012]	56.38%	0.07	58.74%	0.25	\$8,671
Ding [2014]	58.83%	0.16	61.47%	0.31	\$10,375
EB-CNN	64.21%	0.40	65.48%	0.41	\$16,774

Table 4: Final results on the test dataset.

Action = *didn't reach*, Object = *estimates*) in the development data result in different features by Ding et al. [2014] even after backing-off, as the actor and the object of E_1 and E_2 are different, and the predicates of these two events cannot be generalized to the same verb class. However, the semantic distance of these two event embeddings are very small in our model, even though they do not have similar word embeddings. As a result, E_1 can serve as a relative training example for predicting using E_2 in our model.

(3) CNN-based prediction models are more powerful than NN-based prediction models (e.g. WB-CNN vs WB-NN, EB-CNN vs EB-NN, and E-CNN vs E-NN). This is mainly because CNN can quantitively analyze the influence of the history events over longer terms, and can extract the most representative feature vector for the prediction model.

Individual Stock Prediction

We compare our approach with the baselines on individual stock prediction using the development dataset. We use the 15 companies selected by Ding et al. [2014] from S&P 500. The list consists of samples from high-ranking, middle-ranking, and low-ranking companies from S&P 500 according to the Fortune Magazine. The results are shown in Figure 4 (as space is limited, we only show comparison between EB-CNN and the two baselines). We find that:

(1) Our model achieves consistently better performance compared to the baseline methods, on both individual stock prediction and S&P 500 index prediction.

(2) Our model achieves relatively higher improvements on those lower fortune ranking companies, for which fewer news are available. For the baseline methods, the prediction results of low-ranking companies dramatically decrease. However, our model considers the diminishing influence of monthly news and weekly news, which are important features for individual stock prediction. Hence, even without daily news, our model can also give relatively accurate prediction results.

Market Simulation

We simulate real stock trading by following the strategy proposed by Lavrenko et al. [2000], which mimics the behavior of a daily trader who uses our model in a simple way. If the model indicates that an individual stock price will increase the next day, the fictitious trader will invest in \$10,000 worth of that stock at the opening price. After a purchase, the trader will hold the stock for one day. During the holding time, if the stock can make a profit of 2% or more, the trader sells immediately. Otherwise, at the end of the day, the trader sells the stock at the closing price. The same strategy is used for shorting, if the model indicates that an individual stock price will decrease. If the trader can buy the stock at a price 1% lower than shorted, he/she buys the stock to cover. Otherwise, the trader buys the stock at the closing price.

We use the same training, development and test dataset as shown in Table 1, for the simulation. Table 3 summarizes the average cumulative profits over the 15 companies in Section 4.3. These results are obtained on the development data. The cumulative earnings of our model averaged \$16,785 (which means that trading \$10,000 worth of stocks would result in a net profit of \$16,785), which is higher than Luss and d'Aspremont [2012] (\$8,694) and Ding et al. [2014] (\$10,456). Except for the reasons analyzed in Sections 4.3, we notice that if there is no news reported for an individual stock on the previous day, their models cannot predict the trend of the stock price movements on a day, because they do not leverage long-term and mid-term news. This does not hurt the evaluation results of accuracy and MCC, but can hurt the real profit.

To verify the statistical significance of our earnings, we perform a randomization test [Edgington and Onghena, 2007] by randomly buying or shorting for 1000 trials. The mean profit over the randomized test is -\$9,865 and the performance of our model is significant at the 1% level.

Stock	Profit of Lavrenko et al. [2000]	Profit of EBCNN
IBM	$47,000	$42,000
Lucent	$20,000	$27,000
Yahoo	$19,000	$32,000
Amazon	$14,000	$35,000
Disney	-$53,000	$7,000
AOL	-$18,000	$14,000
Intel	-$14,000	$8,000
Oracle	-$13,000	$17,000

Table 5: Profit compared with Lavrenko et al. [2000]. (there are 4 negative profit stocks out of 15 which are not included in this table)

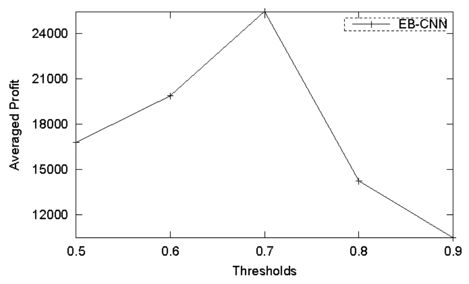


Figure 5: Influence of threshold.

4.4 Final Results

Table 4 shows the final experimental results on the test dataset, where Luss [2012] is the model of Luss and d'Aspremont [2012] and Ding [2014] is the model of our previous work. As space is limited, we only show the average prediction results of 15 individual stocks. The results demonstrate consistently better performance, which indicates the robustness of our model.

Market Simulation

To compare with Lavrenko et al. [2000], Table 5 shows the profit for eight companies (i.e., IBM, Lucent, Yahoo, Amazon, Disney, AOL, Intel and Oracle) selected by them. We use for training data news between October and December 1999, and for test data news of 40 days starting on January 3rd, 2000, which is the same with Lavrenko et al. [2000]. Except for IBM, we achieve consistently better performance.

Better trading strategies. To further investigate the effectiveness of our model, we buy or sell stocks according to the classification probability. If the uptrend probability is higher than a threshold β, we buy $10,000 worth of the stock. If the downtrend probability is higher than β, we short $10,000 worth of the stock. Otherwise, we do not buy or short the stock. The results are shown in Figure 5. We find that the best profit can be achieved when the threshold β is set as 0.7. Using this strategy, the overall profit is $82,000, significantly higher than $21,000 by using Lavrenko et al. [2000]'s strategy. The results suggest space for further improvement. Exploration of sophisticated trading strategies are beyond the scope of this paper.

5 Related Work

Efficient Market Hypothesis (EMH) [Fama, 1965] states that the price of a security reflects all of the information available, and that everyone has a certain degree of access to the information. Despite 50 years of studies from the fields of finance, computer science and other research communities, the debate continues over what kinds of information can be useful for stock market prediction. In Artificial Intelligence (AI), three sources of information has been the most exploited for algorithmic stock market prediction.

First, some prediction techniques leverage historical and time-series data [Taylor and Xu, 1997; Andersen and Bollerslev, 1997; Taylor, 2007]. Researchers believed that predictions can be made through careful averaging of historical price and volume movements and comparing them against current prices. It is also believed that there are certain high or low psychological price barriers, such as support and resistance levels. However, these methods ignore one key source of market volatility: financial news.

With advances of NLP techniques, various studies have found that financial news can dramatically affect the share price of a security [Cutler *et al.*, 1998; Tetlock *et al.*, 2008; Luss and d'Aspremont, 2012; Xie *et al.*, 2013; Wang and Hua, 2014]. Recently, we proposed using structured events to represent news, which can indicate the actors and objects of events [Ding *et al.*, 2014] . However, modeling complex event structures directly, their work is challenged by the new problem of sparsity. To this end, this paper proposes learning event embedding.

Apart from events, sentiment is another perspective of deep semantic analysis of news documents [Das and Chen, 2007; Tetlock, 2007; Tetlock *et al.*, 2008; Bollen *et al.*, 2011; Si *et al.*, 2013]. Tetlock [2007] examines how qualitative information (i.e. the fraction of negative words in a particular news column) is incorporated in aggregate market valuations. Bollen and Zeng [2011] find that large-scale collective emotions (representing public moods) on Twitter is correlated with the volatility of Dow Jones Industrial Average (DJIA). Si et al. [2014] propose to regress topic-sentiment time-series and stock's price time series. Their work is orthogonal to event-driven stock market prediction.

6 Conclusion

We demonstrated that deep learning is useful for event-driven stock price movement prediction by proposing a novel neural tensor network for learning event embeddings, and using a deep convolutional neural network to model the combined influence of long-term events and short-term events on stock price movements. Experimental results showed that event-embeddings-based document representations are better than discrete events-based methods, and deep convolutional neural network can capture longer-term influence of news event than standard feedforward neural network. In market simulation, a simple greedy strategy allowed our model to yield more profit compared with previous work.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Basic Research Program (973 Program) of China via Grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via Grant 61133012 and 61472107, the TL SUTD grant IGDST1403012 and SRG ISTD 2012 038 from Singapore University of Technology and Design. We are grateful to Siddharth Agrawal for discussions in working with neural tensor network.

References

[Andersen and Bollerslev, 1997] Torben G Andersen and Tim Bollerslev. Intraday periodicity and volatility persistence in financial markets. *Journal of empirical finance*, 4(2):115–158, 1997.

[Bengio *et al.*, 2005] Yoshua Bengio, Hugo Larochelle, and Pascal Vincent. Non-local manifold parzen windows. In *Proc. of NIPS*, pages 115–122, 2005.

[Bengio, 2009] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[Bollen *et al.*, 2011] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[Bordes *et al.*, 2011] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *Proc. of AAAI*, 2011.

[Cutler *et al.*, 1998] David M Cutler, James M Poterba, and Lawrence H Summers. What moves stock prices? *Bernstein, Peter L. and Frank L. Fabozzi*, pages 56–63, 1998.

[Das and Chen, 2007] Sanjiv R Das and Mike Y Chen. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388, 2007.

[Ding *et al.*, 2014] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proc. of EMNLP*, pages 1415–1425, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Edgington and Onghena, 2007] Eugene Edgington and Patrick Onghena. *Randomization tests*. CRC Press, 2007.

[Fader *et al.*, 2011] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proc. of EMNLP*, pages 1535–1545. Association for Computational Linguistics, 2011.

[Fama, 1965] Eugene F Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.

[Jenatton *et al.*, 2012] Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. A latent factor model for highly multi-relational data. In *Proc. of NIPS*, pages 3167–3175, 2012.

[Kogan *et al.*, 2009] Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. Predicting risk from financial reports with regression. In *Proc. of NAACL*, pages 272–280, 2009.

[Lavrenko *et al.*, 2000] Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*, pages 37–44, 2000.

[Luss and d'Aspremont, 2012] Ronny Luss and Alexandre d'Aspremont. Predicting abnormal returns from news using text classification. *Quantitative Finance*, (doi:10.1080/14697688.2012.672762):1–14, 2012.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Radinsky *et al.*, 2012] Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. Learning causality for news events prediction. In *Proc. of WWW*, pages 909–918. ACM, 2012.

[Schumaker and Chen, 2009] Robert P Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *TOIS*, 27(2):12, 2009.

[Si *et al.*, 2013] Jianfeng Si, Arjun Mukherjee, Bing Liu, Qing Li, Huayi Li, and Xiaotie Deng. Exploiting topic based twitter sentiment for stock prediction. In *Proc. of ACL*, pages 24–29, Sofia, Bulgaria, August 2013.

[Si *et al.*, 2014] Jianfeng Si, Arjun Mukherjee, Bing Liu, Sinno Jialin Pan, Qing Li, and Huayi Li. Exploiting social relations and sentiment for stock prediction. In *Proc. of EMNLP*, pages 1139–1145, Doha, Qatar, 2014.

[Socher *et al.*, 2013] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Proc. of NIPS*, pages 926–934, 2013.

[Taylor and Xu, 1997] Stephen J Taylor and Xinzhong Xu. The incremental volatility information in one million foreign exchange quotations. *Journal of Empirical Finance*, 4(4):317–340, 1997.

[Taylor, 2007] Stephen J Taylor. Modelling financial time series. 2007.

[Tetlock *et al.*, 2008] Paul C Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance*, 63(3):1437–1467, 2008.

[Tetlock, 2007] Paul C Tetlock. Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3):1139–1168, 2007.

[Wang and Hua, 2014] William Yang Wang and Zhenhao Hua. A semiparametric gaussian copula regression model for predicting financial risks from earnings calls. In *Proc. of ACL*, pages 1155–1165, 2014.

[Xie *et al.*, 2013] Boyi Xie, Rebecca J. Passonneau, Leon Wu, and Germán G. Creamer. Semantic frames to predict stock price movement. In *Proc. of ACL*, pages 873–883, 2013.

[Zhang and Clark, 2011] Yue Zhang and Stephen Clark. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151, 2011.