INTERNATIONAL UNIVERSITY

MACHINE LEARNING AND ITS APPLICATION IN FINANCE

by

Do Viet Ho Tam Thuc

A thesis submitted in partial fullfillment for the
degree of Bachelor of Science

in the
International University
Department of Mathematics

June 2017

If we knew what it was we were doing, it would not be called research, would it?

— Albert Einstein

INTERNATIONAL UNIVERSITY

Abstract

International University
Department of Mathematics

Bachelor of Science

by Do Viet Ho Tam Thuc

Recent developments in the field Artificial Intelligence have brought financial engineers new advance tools for modeling the stock market. Nowadays, supercomputers are integrated into most stock exchanges to capture important signals from market and try to predict the movements of the stocks. In this thesis, we propose a deep learning model to capture and understand the events in newspaper, then predict the movement of stock price. The proposed model is simple and does not need to use a supercomputer, which is suitable for a individual trader, it achieves more than 65% accurate performance and highly comparable with the state of the art methods.

Acknowledgements

I would first like to thank my thesis advisor Dr. Nguyen Tien Dung of the Mathematics Department at Ho Chi Minh City University of Technology. I also realize that without all knowledge received from all my professors in Mathematics Department at Ho Chi Minh City International University in four years, I can not complete my thesis. So i dedicated all my work to them.

I would also like to acknowledge Dang Gia Huy, Nguyen Ngoc Son An, Mr.Phan Thai Duy Tan and Mrs.Tran Ngoc Nam Tran. I am gratefully indebted to them for their very valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author

Tam Thuc

Contents

Abstract **ii**

Acknowledgements **iii**

Abbreviations **vi**

1 Introduction **1**
- 1.1 Literature review . . . 1
- 1.2 Thesis structure . . . 3

2 Deep learning Models **4**
- 2.1 What is Deep Learning ? . . . 4
  - 2.1.1 Deep Feedforward Networks . . . 6
  - 2.1.2 Convolutional Neural Networks . . . 7
  - 2.1.3 Recurrent Neural Networks . . . 10
- 2.2 Optimization for training deep models . . . 12
  - 2.2.1 Learning process and traditional optimization . . . 12
  - 2.2.2 Basic learning algorithm . . . 13

3 Methods **16**
- 3.1 Data Preprocessing . . . 16
- 3.2 Information Extraction . . . 16
- 3.3 Embedding Vector of Information . . . 17
  - 3.3.1 Model Architecturre . . . 17
  - 3.3.2 Training Process for Events Embedding . . . 18
- 3.4 Stock Price Movement Prediction Model . . . 19

4 Experimental Setup **21**
- 4.1 Data Collection . . . 21
- 4.2 Baseline stock prediction models and proposed models . . . 22
- 4.3 Apple Stock Prediction . . . 22

5 Results, Discussion & Conclusion **24**
- 5.1 Prediction Accuracy and Baseline Comparing . . . 24
- 5.2 Conclusion . . . 25

A Detail data collecting method **26**

Bibliography **28**

Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
MLPs	**M**ulti**L**ayer **P**perceptrons
LSTM	**L**ong-**S**hort **T**erm **M**emory network
SGD	**S**tochastic **G**radient **D**escent

Dedicated to all my teachers, especially Dr. Nguyen Minh Quan,
my linear algebra knowledge was built base on his amazing lectures

Chapter 1

Introduction

Nowadays, the community of data scientists has turn their attention to Deep Learning and used it to solve complex problem. My thesis will present a model, the input is raw text in newspaper from some previous days and the output is the prediction that stock price in the next day will increase or not. It achieves more than 65% accurate performance

We realize that Natural Language Processing (NLP) technologies had huge advancement in recent years, such as *OLLIE*, an Open Information Extraction Software which can break the sentences into simple and processable structures. Hence, we use *OLLIE* to extract events from newspapers. An event is a tuple extracted from a sentence, it is presented as (O_1, P, O_2) where O_1, and O_2 are subjects and P is the verb in the sentence.

Then, we train an event embedding model using the architecture proposed in Ding et al. [1], this will learn a vector to represent a event in the newspaper. We use these vectors as input for the movement price prediction model. Many different deep learning models are used to model the movement of stock prices, this is done to compare and find the most favorable model.

1.1 Literature review

Recently, the deep learning approach has been successfully applied to tasks such as objects classification in Lee et al. [2], speech recognition in Zhang and Wu [3], and some famous deep machine learning approaches such as convolutional neural networks, deep belief networks in Hinton et al. [4] and Long-Short Term Memory Network. However, its application in financial forecasting remain a relatively unexplored area. The challenge is that whether some deep machine model can be modified and learning to reduce the overall risk for trading strategies.

In 2000, a model with the name Enalyst was introduced in Lavrenko et al. [5]. Their goal is to predict stock intraday price trends by analyzing news articles published in the homepage of YAHOO finance. The author included only those 127 U.S. stocks that showed largest positive or negative price return. But such selection will leads to bias towards highly volatile stock as they concluded.

In 2006, Mittermayer and Knolmayer [6] implemented several prototypes for predicting the short-term market reaction to news based on text mining techniques. Their model forecast 1-day trend of the five major companies indices: Dow Jones, the Nikkei, the FTSE, the Hang Seng, and the Straits Times. The authors trained a Naive Bayes classifier for the stock movement. However, the simulated performance results of their model cannot be achieved in reality as reported in their paper.

In 2010, Borsje et al. [7] propose a semi-auto method to construct an engine. This engine use rules to extract text pattern in newspaper and execute appropriate update actions when the extracted pattern matches with a known pattern from the past news. The steps are described as follow:

- Mining raw text newspaper for patterns
- Creating an event if a pattern is found
- Determining the validity of an event by the user
- Executing appropriate update actions if an event is valid

Another example of deep learning in financial was introduced by Yoshihara et al. [8]. In their report, they used recurrent deep neural network to predict market trend by modeling temporal effects of past events.

The similar problem was approached in Ding et al. [1] where a combination of neural tensor network and a deep convolutional network was used to model short-term and long-term influences of events on stock price movements. This thesis bases on their approach about events modeling, however, we use another architecture to model the price movement. This will shorten the training process and reduce the computation resource which make our model become easy to applied on small machine but still gain high accuracy on stock movement prediction.

1.2 Thesis structure

To our knowledge, there are many deep learning model that were studied and applied to many field. As the goal of our thesis, we will conduct some experiments on "the best reported results in the literature" as shown in Ding et al. [1] but with some modification. The changes we make during the experiment will only help us gain insight about the model, how the events were capture by the neural network and presented as driven-factor of stock movements.

Before going further into detail, we will first summarize some famous deep learning model in chapter 2 such as neural tensor network, deep convolutional network and recurrent deep neural network. These model largely contributed to the present success of the field artificial intelligence.

In addition, we also design other model to compare to the proposed model, this will help us realize some key factor that lead to the success of the model. The baseline models and the main model will be illustrated in chapter 4, Experimental Setup along with the detail architecture of the proposed model in chapter 3. As the final chapter of the thesis, we discuss some further approach on the problem also some drawback of the model and how to improve it.

Chapter 2

Deep learning Models

2.1 What is Deep Learning ?

Deep learning can be considered as a special type of machine learning. In order to deeply understand it, one must need a strong background in machine learning. Therefore, as the beginning of ours thesis, we will provide the most important general principles about machine learning as warming up knowledge before going detail about our work.

According to the book Machine Learning, Tom Mitchell (1997), a machine learning algorithm is an algorithm that learn from data as the denition "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P,improves with experience E". We can image that how general machine learning algorithm is with this definition about task T, experience E and performance measure P. Hence, we will provide some example of the tasks and its performance measure.

The task in most machine learning algorithms is to figure out how a machine learning system should process an example also called as a sample in data. A sample is mostly presented as a set of features which is measured from an object, an event that a machine can process. For instance, a picture is a sequence of vectors, each vector is a sequence of number measure a value proportional to the light intensity at that particular location, mostly known as pixel. For simplicity, in this thesis, let consider $\mathbf{x} \in \mathbb{R}^k$ and each x_i is a feature. Below is some task can be solved by machine learning:

- **Classication**: This task is the most fundamental task in machine learning problems. In this type of task, the machine must learn to classify which of **k categories** that the **input sample** belongs to. To be able to solve the task, the algorithm must return a function $f : \mathbb{R}^k \rightarrow \{1, ..., k\}$, the model $y = f(x)$ can assign the class of the object encoded as an index $y \in 1, ..., k$. An example of this type of task is stock price trend forecasting in Dai and Zhang [9], input is an vectors of feature such as PE ratio, PX volume, PX EBITDA etc, and output is the movement prediction in next n-days encoded as $0, 1$.

- **Regression**: In this type of task, the machine is asked to predict a numerical value given some input. The machine learning algorithm solves this task by learning the function $f : \mathbb{R}^k \rightarrow \mathbb{R}$. An example of this task in finance is predicting the expected claim amount that an insured person will make or predict the future prices of securities.

- **Feature learning**: In this task, feature learning or representation learning, the program is asked to transform raw data input to a presentation that other machine learning task can exploit effectively. This task is motivated by the fact that most machine learning tasks such as **Classification** often require the input to be computationally convenient to process. But some real world data such as words, literature, or speech are too complex and unique. Therefore, it is always recommended for any model in NLP to begin with a feature embedding process. An example of this task is the famous articles learn to transform words into dense vectors Pennington et al. [10].

The second important part of a machine learning algorithm is the performance measure P. It is a quantitative measure to evaluate the goodness of the algorithm, for instance, accuracy is often used as a measure for classification task. Usually, we use a **test set** to measure how well a model performs on unseen data and keep the training data separately.

The next part, which is no less important is the experience E. It is often defined as the dataset - a collection of many samples, or more formally call as data points. By the characteristic of the dataset, learning algorithms are divided into two well known types:

- **Unsupervised learning algorithms**: The dataset in this type containing many features, and its job is to learn useful characteristic, and structure of the data. The classic machine learning task that fits on this type of algorithm is **Feature learning** task. Some other learning algorithms perform other tasks, like clustering, which divides the origin dataset into clusters of similar data points.

- **Supervised learning algorithms**: experience a dataset containing features, but each data point is also associated with labels $\mathbf{y}$. A supervised learning algorithm can learn from data to classify objects into categories.

Roughly speaking, unsupervised learning involves in learning from data points of a random vector x and attempting output the probability distribution $p(x)$, or some interesting properties of that distribution, while supervised learning involves learning to predict y from x, often by estimating $p(y \mid x)$ where vector y is the associated value of random vector x.

2.1.1 Deep Feedforward Networks

Deep feedforward networks, also often called **feedforward neural networks**, or **multilayer perceptrons** (MLPs) is the most fundamental deep learning model. General goal of the model is to approximate some function f^*. For instance, a classification function $y = f^*(x)$ maps input vectors x to the label y. The model is formally defined as $y = f(x; \theta)$ and the learning algorithm will learn the parameter θ that outputs the best function approximation. The word *feedforward* originated from the fact that the information flows through the model, starting from vector x as input layer, then to the intermediate computational nodes used to define f and finally to the output y. There are no **feedback nodes** in which the computation nodes receive again its output from previous flows, these feedback nodes often known as **recurrent neural networks**, presented in subsection 2.1.4

Feedforward networks are the most important architecture to machine learning practitioners. They are the backbond of many follow architectures which will be presented later in this thesis. For instance, the convolutional networks used for recognizing object from images, presented in subsection 2.1.3.

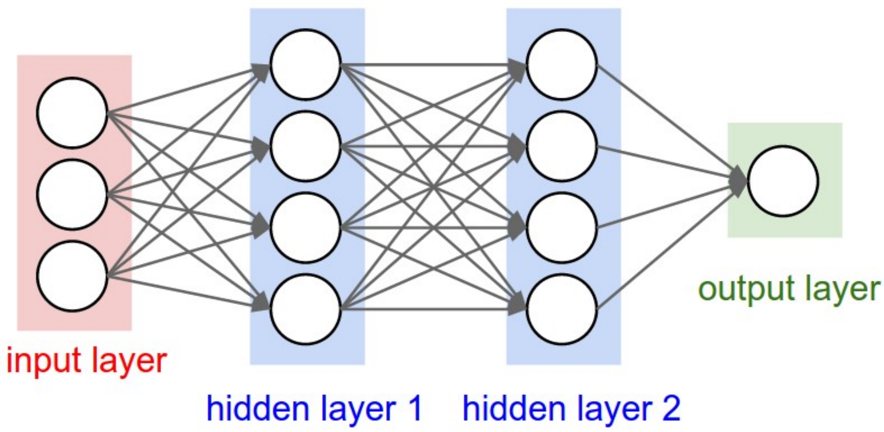


FIGURE 2.1: Feedforward networks model architecture

The word **network** describes the model as a composition of many different functions, they connect together and become a directed acyclic graph. A simple and most commonly graph can be writen as a chain structure $f(x) = f^3(f^2(f^1(x)))$, where f^3, f^2, f^1

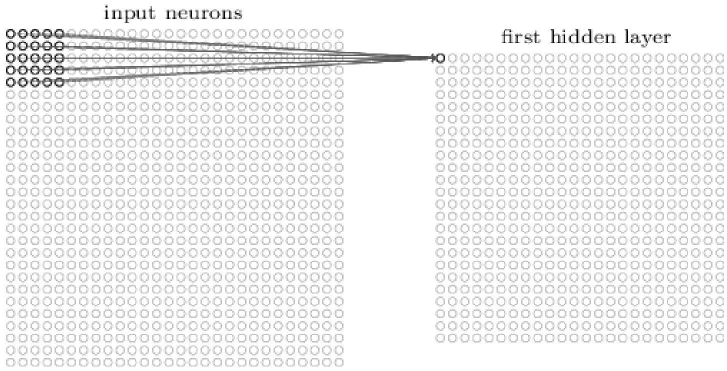
are separately called as **output layer, hidden layer 2, hidden layer 1** and x is **input layer**. The overall number of layer gives the **depth** of the model. It is from this terminology that the name *deep learning* arises. Then, during the training process, the learning algorithm drives the $f(x)$ to approximate $f^*(x)$. Last output $f^*(x)$ of the training process will best classify the object x in to categories y based on the observed data points.

2.1.2 Convolutional Neural Networks

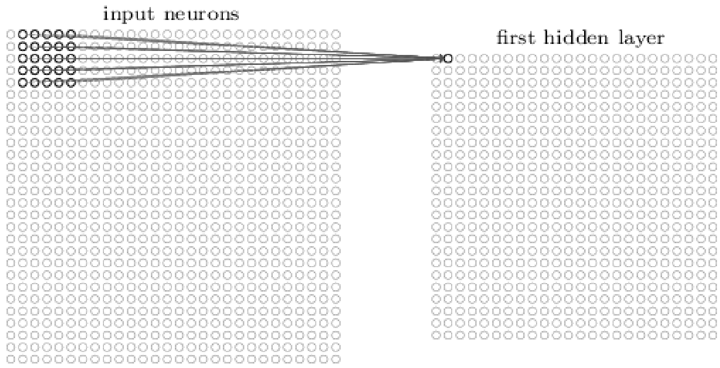
To describe convolutional networks architecture, we will first introduce a simple deep learning model, in which each layer is fully connected to the next layer as described in figure 2.1, with the function $f^i(x) = \sigma(W_i^T x + b_i)$, $i \in \{1, 2, 3\}$, $W_i \in \mathbb{R}^{k \times h}$ where k, h is number of nodes in input, output layers respectively and σ is an activation such as tanh function. This model works pretty well on simple input such as a feature vector x with no spatial structure. However, in real world problem, data points usually have spatial structure such as an images, price of many stocks in days or a sequence of events presented as vectors $x_i \in \mathbb{R}^k$, $i \in \{1, 2, ..., N\}$. Therefore, the convolutional networks (CNNs) was designed to take advantage of the spatial structure in data points.

CNNs was base on three basic ideas: local receptive fields, shared weights, and pooling. Each of these ideas are presented as follow:

- **Local receptive fields:** In figure 2.1, the inputs were depicted as a vertical line of neurons. In CNNs, input is usually presented as a matrix $N \times N$ square of neurons (nodes). As the logic of feedforward networks model, we will connect the input neuron to a layer of hidden neurons. However, instead of connect all $N \times N$ neurons to the next layers, many small, localized regions of the input matrix of neurons are connected with next neurons. To be more specific, each neuron in the hidden layer next to the input will be connected to a small region of the input neurons, for example, a 5×5 region of 28×28 matrix of neurons. To illustrate this concretely, next figure contains a local receptive field in the top-left corner that connect to a neuron in next hidden layer:



Then we move the local receptive field to the right by one nodes (i.e., by one neuron), to create a second hidden neuron as below figure:



And so on, the first hidden layer was built next to the input matrix of neurons. In fact, different moving steps which mostly known as stride length are used in other model.

- **Shared weights and biases:** In the example above, each hidden neuron has a bias and 5×5 weights connected to its local receptive field at input matrix neurons, and the same weights and bias and used for all hidden neuron. In other words, for the j, kth hidden neuron $h_{j,k}$, it is calculated by:

$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right) \tag{2.1}$$

where σ is the neural activation function, for instance, sigmoid function, b and $w_{l,m}$ are the shared parameters across all local receptive field and $a_{x,y}$ is the input neuron at position x, y. This shows that at different location in the input matrix, the exactly same feature was detected and presented in the first hidden layer. For instance, if the input is an image, and b, $w_{l,m}$ can recognize a circle in the local receptive field, this ability must be useful at other location, so all location should be applied with b and $w_{l,m}$. To make it more abstract, CNNs are well suitable for finding location variation of a input matrix. If we move the matrix of neurons away, CNNs still detect the local feature in the matrix.

For this reason, the mapping from the input matrix to the hidden matrix is often called as a *convolutional kernels* with b and $w_{l,m}$ are said to define a kernel or filter. Therefore, a complete convolutional layer contains many different convolutional kernels associated to with different mapping as described below. Note that some deep architecture can use many more kernels than 3 kernels:

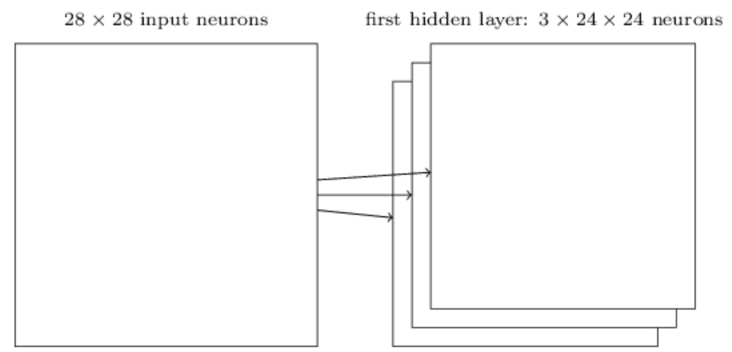


FIGURE 2.2: A complete convolutional layer

- **Pooling layers:** In addition to the convolutional layer, convolutional neural networks also consist of pooling layers. The job of pooling layers is to summarize the output from convolutional layer. In detail, each neuron in it summarizes a region of $n \times n$ neurons in the output from previous convolutional layer. A common procedure is max-pooling layer, whose each unit simplifies outputs in the 2×2 input region by the max function as illustrated in the following diagram:

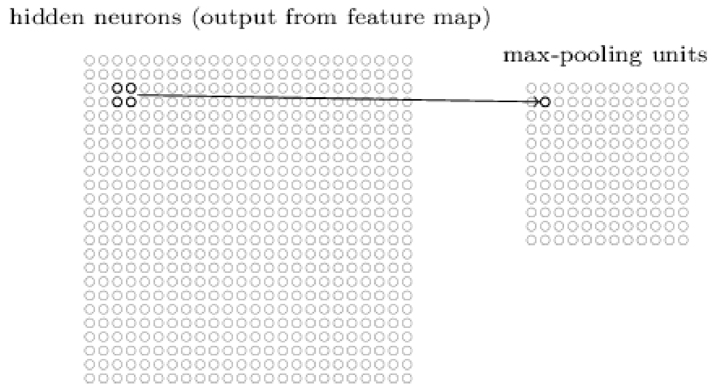


FIGURE 2.3: A pooling layer

We can interpret the pooling layer as a way to collect information from larger local field using information from smaller local receptive field from input layer. This method will output a condensed feature map. Another benefit from this layer is that it reduces computational power needed in later optimization by reducing number of parameters but still captures needed information.

We now can have a full understanding about a complete convolutional neural network. The architecture of the network can be visualized by a simple image:

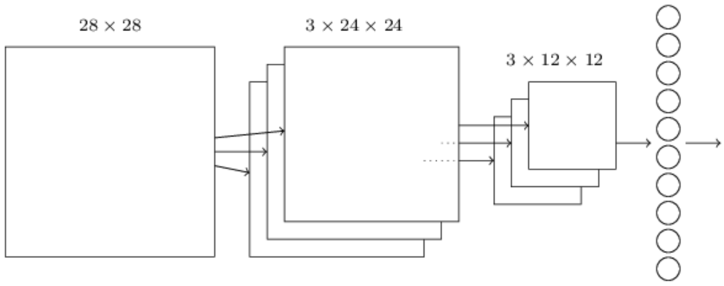


FIGURE 2.4: Network architecture

2.1.3 Recurrent Neural Networks

Human always need context to fully understand a word, for instance the word "bodies" in the sentence "Some celestial bodies, such as the planets and stars, can be seen with the naked eye". Words combine together to have other meaning, as well as events in newspapers in different days or sound when we talk. So, traditional feedforward network seems to fall when trying to address this issue, but Recurrent Neural Networks do not. They are networks that built from smaller and repeated network in them, allowing information to be learned repeatedly. In the last couple years, there have been many dramatic innovations in applying RNNs to AI problems such as: speech recognition, language modeling, translation, image captioning. Essential to these innovations is **LSTM** architecture, .

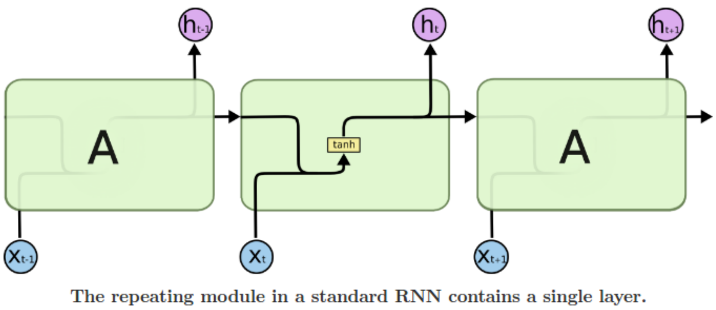


FIGURE 2.5: Standard RNN architecture

Long Short Term Memory networks mostly known as LSTM are a kind of RNN, capable of learning long-term relationship in sequence of samples. This network was introduced in Hochreiter and Schmidhuber [11] and designed to avoid the long-term dependency problem in RNN. This problem arises when an information, in present time t, depends on another one, which is long time ago at time $t - L$, so that it slowly vanishes when the network processes other irrelevant information between time t and $t - L$. The figure 2.5 above illustrates standard RNNs, where the repeating cell has a simple tanh layer structure. For each time t, the loop processes the information x_t while considering all previous information embedded in y_{t-1}, then produces a output h_t and new state of information y_t which presented as the new summary of all processed information. The detail mathematical calculation is given below:

$$y_t = \sigma(W_y x_t + U_y y_{t-1} + b_y)$$
$$h_t = \sigma(W_h y_t + b_h)$$

Hence, we can realize that the information is embedded into vector y_t through each time step t, but the longer the time of the information the less likely that y_t can capture it, since the same parameters are applied at each step and the later information will overwrite the old one. Therefore, LSTM network has a different structure to address the issue, the cell of it is described in figure 2.6.

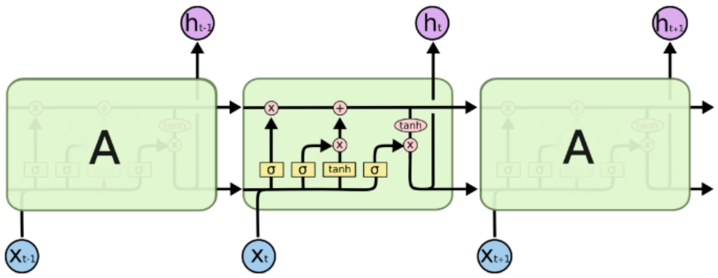


FIGURE 2.6: LSTM architecture

At first step, the module decides which information can be eliminated from the previous cell state C_{t-1} - which presented in the horizontal line running into every cell in the figure 2.6:

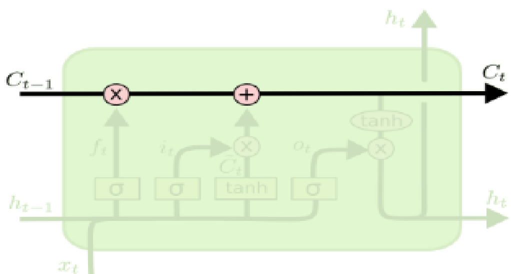


FIGURE 2.7: Cell state line

This is done by calculating the forgeting gate $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$ where σ is sigmoid activation function, h_{t-1} is previous output of the LSTM cell and x_t is the present information. Then, the cell makes a pointwise multiplication operation $f_t * C_{t-1}$, where $f_t \in [0, 1]$, a 0 represents "forget all previous information", and 1 represents "keep all previous information".

The next part of the cell is deciding what new information will be stored in cell state. This is done by calculating two values, input gate i_t and the candidate state values $\tilde{C}_t$.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$\tilde{C}_t = \sigma(W_{\tilde{C}} \cdot [h_{t-1}, x_t] + b_{\tilde{C}}) \tag{2.3}$$

Then the cell will calculate the new cell state $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$. Lastly, the cell continue to decide what is the output of this state though calculation of h_i given by:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.4}$$

$$h_i = o_t * \tanh(C_t) \tag{2.5}$$

What we illustrated so far is a standard LSTM. But not all LSTMs are the same as above. A dramatic different LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho et al. [12]. It replaces the forget and input gates by a single update gate. It also combines the cell state and hidden state in to a simple state, and makes some other changes in ouput gate. The GRU model is simpler than LSTM models, and has been become popular lately due to its fast convergence ability compares with LSTM in the training process. However, in this context, we use LSMT as ours main model because of the complexity of ours problems.

2.2 Optimization for training deep models

2.2.1 Learning process and traditional optimization

Optimization process used in machine learning for deep models have several differences with traditional optimization. In most machine learning cases, the objective of the model is minimizing the performance measure P that use on the test set, but during the training process, only training data is used. Therefore, the learning algorithm uses a indirect optimization process for P, it trained on $\mathbb{J}(\theta)$, a different cost function, and this will improve P. This contrasts to traditional approach, in which minimizes performance measure P directly is the only goal.

Generally, the $\mathbb{J}(\theta)$ function can be presented as an average value of all training samples, such as

$$\mathbb{J}(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{data}} \mathbb{L}(f(x;\theta), y) \tag{2.6}$$

This $\mathbb{J}(\theta)$ defines the objective of the model with respect to the *data*, where $\hat{y} = f(x;\theta)$ is the predicted value when $\mathbf{x}$ is the input, the target output is y in the case supervised learning. The equation 2.6 can be re-written as follow, where m is the number of samples in data:

$$\mathbb{J}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{L}(f(x^{(i)};\theta), y^{(i)}) \tag{2.7}$$

The optimizing process which relies on minimizing this objective is often called as **empirical risk minimization**. Empirical risk minimization is prone to overtting, the situation in which deep learning model can memorize training set and poorly perform on test set. In addition, some useful loss function, such as 0-1 loss, have the first derivatives is zero or undefined everywhere else. This is not useful for some effective modern optimization algorithms that base on gradient descent. Hence, in the context of deep learning, another slightly approach is taken.

We use another loss function called as **surrogate loss function**. For instance, the negative log-likelihood (NLL) of the correct class is used as an alternative for the 0-1 loss. NLL allows the model to calculate the conditional probability of target y, given the input x, then choose the class with higher probability, and also yield least error. In some cases, the NLL helps the model to learn more about data, this can be seen as the 0-1 loss of test set continues to decrease even when 0-1 loss of trainnig data has reached zero. This can be done when the learning algorithm continue to push the probability between the classes further apart and learn deeper feature in training data.

2.2.2 Basic learning algorithm

We have briefly described the objective of the learning process in previous section, but largely account for model correctness is the algorithm used to minimize the objective function. Therefore, presented below is the most used optimization algorithms for deep learning.

Input: Learning rate ϵ_k
Input: Initial parameter θ
while *Objective function value larger stopping criterion* **do**
 Sample a minibatch of m examples from the training set $\{x_1, ..., x_N\}$ with corresponding targets y_i.;
 Compute gradient w.r,t all parameters: $\hat{g} \leftarrow +\frac{1}{m} \cdot \frac{\delta \Sigma_i \mathbb{L}(f(x_i;\theta), y(i))}{\delta\theta}$.;
 Apply update at step k: $\theta \leftarrow \theta - \epsilon_k \cdot \hat{g}$
end

Algorithm 1: Stochastic gradient descent (SGD) update at training iteration k

In SGD algorithm, the learning rate is considered as the most crucial hyper-parameter, this is due to the source of noise (the random sampling of m training examples) that keeps the variance of data remains low after the objective function arrives at minimum. Therefore, in real world problem, it is recommended to decrease the learning rate over iteration, such as a linearly decay of the learning rate until iteration τ: $\epsilon_k = (1-\alpha)\epsilon_0 + \alpha\epsilon_0$ where $\alpha = \frac{k}{\tau}$.

While SGD remains as a statue of learning algorithm, in some complex deep learning models, their convergence took very long and huge computational resources. Therefore, the method of momentum is designed to overcome this issue in Qian [13]. The momentum algorithm accumulates past gradients and continues to move in their direction. The update rule is given by:

$$\upsilon \leftarrow \alpha\upsilon - \epsilon_k \frac{\delta \Sigma_i \mathbb{L}(f(x_i;\theta), y(i))}{\delta\theta} \tag{2.8}$$

$$\theta \leftarrow \theta + \upsilon \tag{2.9}$$

The velocity υ accumulates the gradient elements exponentially, the larger α relative to learning rate at iteration k, the more gradient affect the direction of parameters. As we see, momentum helps accelerate SGD in the relevant direction and dampens oscillations by include past gradients in the update. Now, the size of the step at each update depends on how consistent between gradients at each time step, this is how the name momentum derived, it is similar as velocity and force, in which cumulative gradient is considered as force that generate acceleration push the particles (the objective function) down the hill (learning curve) faster.

There are many variants of the momentum strategy, such as Nesterov accelerated gradient (1983) introduces in Nesterov [14], in which the authors designed a strategy to address the issue of momentum on blindly updating parameters base on previous directions. Another considerable strategy is Adaptive Moment Estimation (ADAM), it computes adaptive learning rates for each parameter which illustrated in Kingma and Ba [15] but also keeps an exponentially decaying average of past gradients. In this thesis, we use ADAM as our optimization algorithm for all the model.

Chapter 3

Methods

My work is based on the idea in Ding et al. [1, 16], which was introduced as an empirical analysis of events embedding method. In this method, each event is presented as a feature vector in a compacted form to solve the problem about sparsity of event vectors. Then, machine training process will "learn" from these vector to predict the movement of the stock. The detail of my method will be explained in following sections.

3.1 Data Preprocessing

One of the problems in NLP is the noisy and huge data. To overcome this, unlike previous work of Ding et al. [1], we only choose the most relevant news about the financial market and a specific company to conduct our experiment. This will make the model focus more on the relevant information.

However, with the constrains above, we realize that using the headline of news like previous works Ding et al. [1] will limits the training data. This brings us to the idea of using the body of the news - which is used by newspaper as the summary of the relevant news. This step will make the raw data become more informative than using the headline of the news only.

3.2 Information Extraction

We extract the structured events from the raw text data using one of the most reliable Open Information Extraction technology - $OLLIE$, introduced in *Schmitzet al.* [17]. Given a sentences from the raw text data, $OLLIE$ will extract events from the sentence and output the result as list of events $E_i = (O_1, P, O_2), i \in \{1, 2, 3, ...N\}$ where, O_1 and

O_2 are the objects that P, presented as the action in the sentence, is performed on. Then, I add the published time T to the events for the training step.

The process can be explained in the following example. The sentence "Yahoo fell 0.6 percent to $18.27 in late trading" in $T =$ "$2010-04-20$" is transformed to: ("Yahoo", "fell 0.6 percent in", "late trading", "2010-04-20")

3.3 Embedding Vector of Information

As the result from above step, the events are mostly unique and represent a fraction of information. Therefore, it is necessary to embed the events into a dense vector but still preserves that information. Follow the idea in Ding et al. [1], we use a neuron network model with the same architecture as they described in their paper to present events by vectors of feature extracted by the model.

3.3.1 Model Architecturre

The goal of their model is using machine learning to learn distributed representations of the events. The model's architecture is described in the figure 3.1 below.

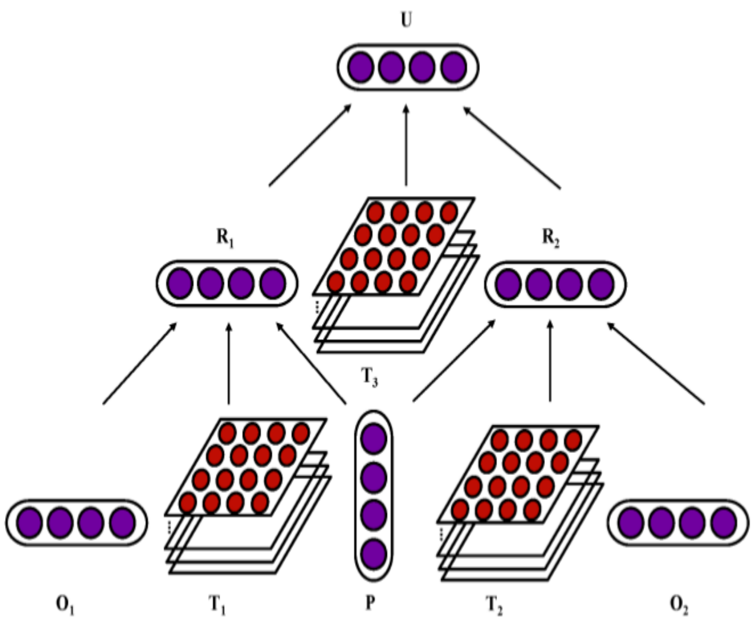


FIGURE 3.1: Model architecture

The input of model is word vectors and the output is an event vector. Since, all the event arguments, objects and actions are lists of words, so they represent an object or an action as the average of its words embeding, $O_i \in \mathbb{R}^d$ and $P \in \mathbb{R}^d$. As illustrated in figure 3.1, the roles of O_1 and O_2 are learned by two tensors, T_1 and T_2 independently, $O_1 T_1 P$ and $P T_2 O_2$ will present as the role-dependent embeddings R_1 and R_2, then, the third layer T_3 is used to semantic compositionality over R_1 and R_2 and generate a complete structured U of the event. From the figure 3.1, $R_1 \in \mathbb{R}^d$ is calculated by:

$$R_1 = f(O_1^T T_1^{[1:k]} P + W \begin{bmatrix} O_1 \\ P \end{bmatrix} + b) \tag{3.1}$$

where $T_1^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ and the bilinear product $O_1^T T_1^{[1:k]} P$ is a vector $r \in \mathbb{R}^k$. Other parameter W, and b is a simple feed-forward network, in which $W \in \mathbb{R}^{k \times 2d}$ and $b \in \mathbb{R}^k$ and f is tanh function. The same architecture described above for R_1 calculation is used to build up the complex architecture as illustrated in figure 3.1.

3.3.2 Training Process for Events Embedding

The model uses more than 5 thousands events from Bloomberg financial news. These events are only about Apple company from the year 2005 to 2009 and used as the training data for modeling the events. The training data is smaller than previous work in Ding et al. [1] but much more specific about Apple company, and this time frame is chosen because it is when Apple released its first IPhone, which strongly impacts the company in later years. The purpose is to model the events in this time frame and see how it affects Apple stock.

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 $EELM'$

```
random replace the event argument and got the corrupted event tuple
ℰ^r ← (E_1^r, E_2^r, ⋯, E_n^r)
while ℰ ≠ [ ] do
    loss ← max(0, 1 − f(E_i) + f(E_i^r) + λ‖Φ‖_2^2
    if loss > 0 then
        Update(Φ)
    else
        ℰ ← ℰ/{E_i}
return EELM
```

The training algorithm is described as above, they assume that to modeling the events, the model has to be able to classify the true events E with corrupted one E^r. The corrupted event is created by replacing one of the objects in true events with a no meaning objects randomly selected from the vocabulary of training data. Then, the objective of the model is to minimize the loss function below:

$$loss(E, E^r) = max(0, 1 - f(E) + f(E^r)) + \lambda \times \|\theta_{exclude\ b}\|_2^2 \tag{3.2}$$

where $\theta = (T_1, T_2, T_3, W, b)$ is the set of parameters that need to update in every iterations to achieve the minimum of the objective function. The standard L_2 regularization on parameters θ except b is added by hyper-parameters λ. The algorithm will repeatedly loop over the training data. At each loop, the model will calculate the loss function between the sample (one event tuple) and the corrupted one; if the loss is larger than 0 then the model will update the parameters to minimize the loss, using the ADAM optimization algorithm Kingma and Ba [15] which bases on the standard back-propagation algorithm. If loss equals 0, then skip the sample and continue to the next sample.

3.4 Stock Price Movement Prediction Model

After the learning process to represent the events by vectors, we model the effect of the events over the past week on stock prices based on the recurrent neural network (RNN).

The input sample is a sequence of daily event vectors in five days, which the daily vectors is the average of all events in that day. The output of model is the binary class represents the movement of stock price where 1 is "increasing" and 0 is "decreasing".

The natural model that fits extremely well on sequence data is RNN, so we use its as our second framework for predict stock movement. In previous paper, the Convolutional Neural Network (CNN) was used for these training data, but we recognize that our data is small and using such large architecture for different time spans (monthly, weekly, daily) to model specifically about Apple is costly. Therefore, a simple Long-Short Term Memory network (LSTM) is used on weekly time span, with smaller amount of parameters. The result of our model in test data is very competitive with the previous work, 64.44% accuracy using LSTM.

Formally, given a sequence of daily event vectors $U = (U_1, U_2, U_3, U_4, U_5)$. For each time step i with input U_i and previous hidden state h_{i-1} we compute the updated hidden state $h_i, C_i = LSTM(x_i, h_{i-1}, C_{i-1})$. The last hidden state is of the LSTM layer is used in the next feed-forward layers. The output of feed-forward layer is the prediction of the model which is a vectors $y \in \mathbb{R}^2$ represent the score for 2 class increase/decrease of the Apple stock price.

Chapter 4

Experimental Setup

4.1 Data Collection

We collect news from Bloomberg news API over the periods from January 2005 to January 2010, and instead using all news titles, we searched news with the key word "Apple". The result is a list of news titles with the paragraph contains the relevant information. We collected the paragraphs and break it into sentences, then we extracted events using only those sentences. The events outputted from $OLLIE$ were then filtered by its score from $OLLIE$, events with score less than 0.7 will be filtered out. Details about our training data is shown in table 4.1 below.

	Number of sample
New paragraphs with key word "Apple"	34514
Events extracted from paragraphs	57502
Filtered Events	6857

TABLE 4.1: Table to summarize training Events

The experiments in our paper focus on predicting the Apple stock prices obtaining from Yahoo Finance. We collect the stock price data and divide into three groups for training, validating and testing in the stock trend prediction model. The summary of data set is illustrated in table 4.2.

	Number of sample (days)
Training Data	727
Validating Data	80
Testing Data	90

TABLE 4.2: Table to summarize training stocks model

4.2 Baseline stock prediction models and proposed models

The baseline models are some simple models using standard feedforward neural network on the flatten event embeddings vectors in 5 days and CNN model on sequence of event vectors in 5 days. These model were described in detail as baselines model for comparing:

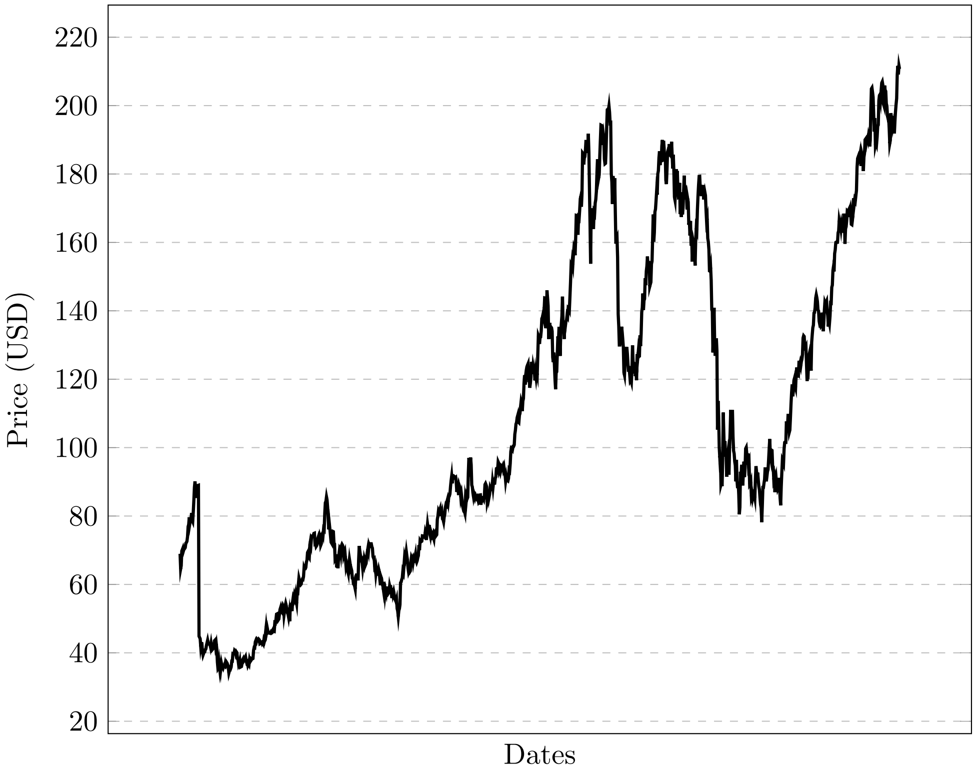
- Model 1: In this model, we use a simple fully-connected neural network, each t days samples were flatten and fully connects to one node in next hidden layer. This model contains 2 hidden layers with 512 and 1024 nodes separately, using sigmoid activation function, the output is 2-d vectors presented as the prediction. During training process, the dropout layers were added for regulation purpose.
- Model 2: The second base model was design base on CNN architecture, it contains 2 convolutional layer, each layers has 128 kernels and one max-pooling layer. The training process is the same as the first baseline model.

In contrast to the baselines, we use the a neural tensor network to learn event vectors as in Ding et al. [1] but then use a deep Long-Short term memory network to build the price movement prediction model. This is our proposed model, note that all model were trained using ADAM optimization process. The detail result of those baselines and proposed model are shown by the table in the next chapter.

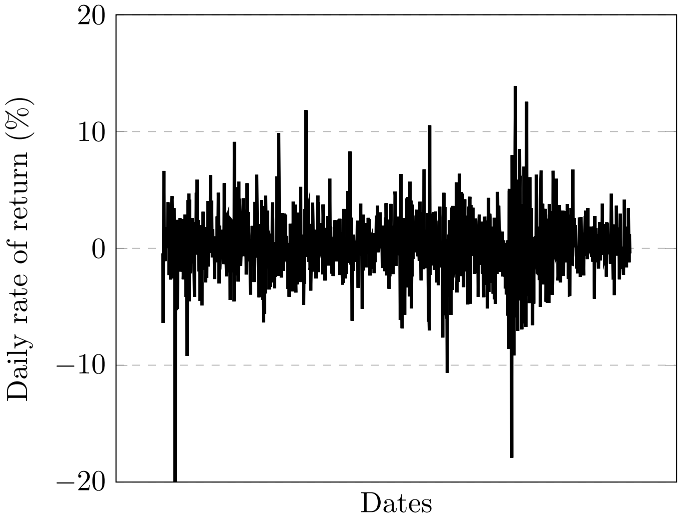
4.3 Apple Stock Prediction

Apple Inc is an American multinational technology company that designs, develops, and sells consumer electronics, computer software, and online services. The success of the company were build on their reputation, their enormous user, and amazing marketing plan. These reason are important factors that make us decided to choose AAPL stock price, because this show that AAPL is strongly impacted by news, important events and sensitive to the behaviors of their customers.

Apple Stock Price from 2005 to 2010



Apple Stock rate of return from 2005 to 2010



Chapter 5

Results, Discussion & Conclusion

5.1 Prediction Accuracy and Baseline Comparing

Model name	Accuracy(%)	Recall(%)	Precision(%)	F-measure(%)
Fully-connected neural network 5-days	56.67	60.7	62.0	60.1
Convolutional neuron network 5-days	60.0	0.5717	0.5352	0.5296
Proposed Model: LSTM 5-days	64.4	76.3	66.04	70.2

TABLE 5.1: Table to summarize training stocks model

As shown in table 5.1, the proposed model using LSTM achieved highest performance compared to other model. This shown that the LSTM is able to learn the information embedded into events vectors and takes advantaged of the time structure in data to predict the stock price movement.

In spite of small training sample, the CNN model achieved very good performance as pointed out by previous work in Ding et al. [1], the model capture the events vectors and predict with 60% accuracy. However, we have to mention that this model only use 5-days events time frame as different from their previous works, which use both 5-days and 30-days time frames as input for the model.

Therefore, we conclude that although the long-term time frame events might contribute to the performance of the model, short-term time frame events still be considered as key-driven factor for the model predictions.

5.2 Conclusion

Good performance of the model on real data shown that it can be applied in the real market as auto trading robot. The output of the model can be converted into a vector of probabilities presents the movement of stock price. Then, from the most simple trading strategy to the most complex can be applied to make use of those events vectors. Applying trading strategies is out of the scope of the thesis, but the model shown good performance, hence, it is promising to make further experiments of the model as the auto trading robot.

We also realize that our experiment was conducted on a laptop, so the lack of computational power might limit the performance of the model. Therefore, for further experiments later, we will use more data and train the model on better machine to analyze the full potential of the model.

Appendix A

Detail data collecting method

:

The data was collected by using a crawler coded in *Python* and Bloomberg News API. The Bloomberg API URL format contains 5 component:

- *Main URL*: "https://www.bloomberg.com/"
- *search?query*: The query key word
- *page*: the page number that user want to see

To be more specific, the URL:

"https://www.bloomberg.com/search?query=Apple&page=2"

will link us to the page 2 of the list result when we query the key word "Apple". The API only allow about 10000 query per day using one IP address. Hence, we spent more than a week to collect all newspaper about Apple.

Then, a simple script in *Python* language is used to parse the HTML format to JSON format, which is a list of news, each element consist of three attributes:

- *Time*: the exact date that the new was published
- *Headline*: the headline of the newspaper
- *Body*: A small paragraph surrounding the matched key word "Apple". It presents the idea of the newspaper about that key word

In the last step, we sort all newspaper in chronological order. Some example of the newspaper that described in JSON format are shown below.

{
"time": "2017-03-16T11:54:57+00:00",
"headline": "Return of the Governator and Linking Brains",
"body": " (Stratechery); see also How Microsoft built its Slack competitor (the Verge)A Conversation with Apple store creator Ron Johnson (Recode)A Safer Road Map Toward Self-Driving Cars (Bloomberg)Is Your... "
},
{
"time": "2017-03-16T15:00:07+00:00",
"headline": "Your Exclusive Bloomberg View CEO Tournament Bracket",
"body": " the basketball tournament, we will divulge who made it to the second round. Will Tim Cook of Apple be able to boost his overseas cash stash faster than Eddie Lampert revives Sears? Can Mary Barra... "
},
{
"time": "2017-03-16T16:50:34+00:00",
"headline": "Your Evening Briefing",
"body": " join the EU.Swiss privacy for your wrist. Swatch Group said it2̌019s developing an alternative to Apple2̌019s iOS and Google2̌019s Android operating systems for smartwatches, as Switzerland2̌019s largest maker of... "
}

Bibliography

[1] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *IJCAI*, pages 2327–2333, 2015.

[2] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[3] Xiao-Lei Zhang and Ji Wu. Deep belief networks based voice activity detection. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(4):697–710, 2013.

[4] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[5] 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.

[6] Marc-André Mittermayer and Gerhard Knolmayer. *Text mining systems for market response to news: A survey.* Institut für Wirtschaftsinformatik der Universität Bern, 2006.

[7] Jethro Borsje, Frederik Hogenboom, and Flavius Frasincar. Semi-automatic financial events discovery based on lexico-semantic patterns. *International Journal of Web Engineering and Technology*, 6(2):115–140, 2010.

[8] Akira Yoshihara, Kazuki Fujikawa, Kazuhiro Seki, and Kuniaki Uehara. Predicting stock market trends by recurrent deep neural networks. In *Pacific Rim International Conference on Artificial Intelligence*, pages 759–769. Springer, 2014.

[9] Yuqing Dai and Yuning Zhang. Machine learning in stock price trend forecasting.

[10] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[13] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[14] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Doklady an SSSR*, volume 269, pages 543–547, 1983.

[15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *EMNLP*, pages 1415–1425, 2014.

[17] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics, 2012.