University of Oxford

Dissertation

Using Financial Reports to Predict Stock Market Trends With Machine Learning Techniques

Author:
Yao Zhang

Supervisor:
Dr Ani Calinescu

A thesis submitted for the degree of
Master of Computer Science
Trinity 2014
August 31, 2015

Acknowledgements

I owe my deepest gratitude to my supervisor Dr. Ani Calinescu, who are dedicated to guide me during the entire process of this dissertation. Whenever I need a help, she is always there. I acknowledge the dissertation could not gain this level of satisfaction without her encouragement, advice and patient help. It was an honor to work with her and follow her suggestions.

I also would like to thank my parents who support me during the study in the University of Oxford.

Abstract

Stock markets as a fundamental component of financial markets play an important role in the countries' economies. The factors that affect the price of stocks include the political situations, company performance, economics activities, and some other unpredicted events. The traditional prediction approach is based on historical numerical data such as the previous trend, trading volume, earnings surprise and some other numerical information. This thesis reviews the literature on the application of information retrieval and machine learning techniques and proposes a framework that uses financial reports to predict the movement of stock prices with machine learning techniques. Deep learning or unsupervised feature learning algorithms have recently been attracting enormous attention in the machine learning communities, but literature on the application of deep learning in stock prediction is still limited.

In this thesis, three deep learning models were adapted into the system to investigate their applicability. They are the Stacked Denoising Autoencoders (SDA), Deep Belief Network (DBN) and Recurrent Neural Networks-Restricted Boltzmann Machine (RNNRBM). The first two are the classic deep learning models and the last one has the potential ability to handle the temporal effects of sequential data. Two other state-of-the-art supervised learning models Random Forests (RF) and Support Vector Machine (SVM) were also adapted to the problem to compare with the deep learning models in the evaluation stage. Through the analysis on experiment results, the proposed models exhibit the importance of predicting stock price trend with containing textual information. The comparison between the performance results of the five models shows that deep learning models have a potential applicability in the context of stock trend prediction, especially for DBN and RNNRBM.

List of Abbreviations

- ANN: Artificial Neural Network
- AT: Algorithm Trading
- DBN: Deep Belief Network
- DP: Down-trend Precision Rate
- DR: Down-trend Recall Rate
- downF_1: Down-trend F_1 Score
- EMA: Exponential Moving Average
- MACD: Moving Average Convergence/Divergence
- PA:Prediction Accuracy
- RBM: Restricted Boltzmann Machine
- RF: Random Forest
- RNN: Recurrent Neural Network
- RNN-RBM: Recurrent Neural Networks-Restricted Boltzmann Machine
- RSI: Relative Strength Index
- SDA: Stacked Denoising Autoencoder
- SMA: Simple Moving Average
- SVM: Support Vector Machine
- Tf-idf: Term frequency-inverse document frequency
- UP: Up-trend Precision Rate
- UR: Up-trend Recall Rate
- upF_1: Up-trend F_1 Score

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Financial markets are nonlinear dynamic and complex systems which contain a significant level of noise, but they do not behave randomly [21, 50]. Stock markets as a fundamental component of financial markets play an important role in the countries' economies. From the process of stock market trading, companies can raise funds to invest in their technology and infrastructure for their development, and the stockholders can obtain extra assets from dividends, as additional income [59]. There are many additional factors that cause fluctuations of stock price movement, such as political situations, company performance, economic activities and other unexpected events, which make the stock trends nonlinear, uncertain, and non-stationary [40]. Hence, it is quite difficult to predict stock market prices and their direction, so the investors must usually monitor the behaviour of the stock prices and pay attention to recent news in order to avoid buying risky stocks with an overrated price and to make correct stock trading decisions. To manage this difficulty, data mining and machine learning techniques may be used to find patterns in the price time series, which can then be used to predict the movement of the current stock prices [59].

Algorithmic trading (AT) refers to using sophisticated algorithms to perform all or some parts of the trade cycle automatically[76]. The trading algorithm can be assumed to access historical and current data. Stock market prediction is a core component of the algorithm trading research area, which mainly focuses on the stock trend prediction [76]. In addition, machine learning is an active research area that attracts increased interest, and which has been applied to stock prediction with some degree of success. A large number of applications have shown supervised machine learning models such as Genetic Algorithms [38, 51], Support Vector Machine [33, 34, 37, 60], Artificial Neural Network [25, 58, 69] and Random Forests [44, 81] can be useful tools to predict the movement trend of stock prices training on the time-series price data, due to their ability to handle non-linear systems. However most of them have still not given sufficiently satisfactory results with very high accuracy and stable performance on stock prediction [2]. Some recurrent versions

of neural networks with feedback, such as [32, 39, 49], have also been tried. They were applied directly on raw data rather than focusing on the feature selection step, like other papers [77]. Thus it can be seen that there is still plenty of room to improve the performance of existing stock prediction models for more accurate decisions and with lower risk of investment. If stock price prediction models could combine some additional information which affects the stock markets such as financial news articles, public sentiment on the social networks, domain knowledge of stock markets, trading volumes etc., a higher accuracy of the prediction could be achieved [2]. Previous research studies on sentiment analysis of Social media websites, such as Twitter, Facebook etc., to predict stock market behaviour have had greater and lesser degrees of success. Among them, [12, 16, 54, 71] have obtained some notable results. However, the volumes of Twitter data are so significant that it is not realistic to process them without a supercomputer.

The limited success of applying the state-of-art machine learning techniques to stock market prediction indicates that we have to add more useful information for better predictions and need more powerful models to fit such complex and high-dimensional combined data in the future [42]. When some significant events appear, they may have profound impacts on the stock markets. Then models constructed based on only numerical data are insufficiently reliable in the prediction of the stock index movement. However, this event information would not exist in the past numerical information, but in the financial articles. The importance of incorporating textual information to perform stock prediction has been shown recently [48]. A natural approach is to add the information of financial news articles to the stock market prediction models, and some efforts and contributions have already been made in this direction [24, 35, 45, 46, 55, 70]. Deep learning or unsupervised feature learning algorithms have recently been attracting enormous attention in the machine learning research communities, because they achieved great success in addressing some problems, particularly in computer vision [17, 41] and natural language processing research [13, 20, 80]. They can automatically learn useful features from a large set of data by unsupervised feature learning in the pre-training process, which is a better way to represent the nature of the data [6]. Additionally, several research studies have shown that unsupervised feature learning and deep learning have the potentially powerful ability to solve time-series modelling problems [42]. Therefore deep learning can fit the description of the challenge of stock market prediction, and provide a new valuable approach to this field. There are several deep learning models in active research. In this thesis, SDA, DBN and RNNRBM are adapted to the problem. To compare the performance of deep learning and classical machine learning models on solving this problem, two widely used and powerful models Support Vector Machine and Random Forests are adapted into solving this problem.

1.2 Project Objectives

The main objective of this project is to adapt three deep learning models (SDA, DBN and RNNRBM) into the context of the stock trend prediction problem with

textual financial report as training data, to investigate the suitability and efficiency of these models and the importance of textual analysis for stock movement prediction. The deep learning models are the Stacked Denoising Autoencoders (SDA), Deep Belief Network (DBN) and Recurrent Neural Networks-Restricted Boltzmann Machine (RNNRBM). The applicability and limitation of these three models can be investigated during the process of implementation and evaluation. The first two are the classic unsupervised feature learning models and the last one proposed in [14] has the potential ability to handle the temporal effects of significant financial events. Two other state-of-the-art supervised learning models Random Forests (RF) and Support Vector Machine (SVM) will be compared unsupervised feature learning models. This objective also can be separated into the following several sub-objectives:

- **Data processing**
  Design a pipeline of text processing for this problem and review the theory of each technique used in the pipeline. Implement the algorithms according to the reviewed theory.
- **Model Building**
  Review the theory background of three learning models, then design the theoretical details of each proposed model and build complete learning models to investigate the applicability of stock price prediction.
- **Conduct experiments and evaluate the results**
  Design some evaluation criteria for prediction performance of proposed models. First, evaluate learning models individually. All of these three deep learning models need to be compared with each other and some other classic machine learning models together. Both price and financial indicator prediction should be evaluated. Some most widely used indicators such as the Simple Moving Average (SMA), Exponential Moving Average (EMA) and Moving Average Convergence Divergence (MACD) etc [1] can be selected to evaluate the performance.

1.3 Overview

- **Chapter 2: Background**
  In this chapter, the financial market background is introduced first and then data source used are described. Following that Artificial Neural Networks are introduced which is a main structure of deep learning models. After that deep learning is introduced.
- **Chapter 3: Textual Articles Preprocessing and Analysis**
  In this chapter, several types of pre-processing steps are introduced, including natural language processing and information techniques used on financial reports and the mapping from historical data to training data.

- **Chapter 4: Build Deep Learning Models**
  In chapter 4, three deep learning models adapted to solve the problem are introduced respectively. The components of each model and detailed training algorithms are presented. The key implementation details are also described.

- **Chapter 5: Evaluation Metrics and Experiment Results**
  In chapter 5, the prediction performance indicators are defined first. The results are fully evaluated on each model. At last, the models are compared and evaluated together. The purpose of these experiments are: 1) to evaluate the importance of textual data on the trend of stock prices. 2) to compare the performance of the adapted deep learning models and other classic machine learning models such as SVM and RF.

- **Chapter 6: Conclusion**
  In this last chapter, the contributions of this project are reviewed and the corresponding theoretical analysis are briefly summarised. After that the limitations of this project are discussed. Finally, several future directions are proposed.

Chapter 2

Background

This chapter provides a brief background introduction of key concepts relevant to this thesis and theory background used in this project. Section 2.2 describes what data source we chose to use and why we chose it. Section 2.3 introduces the theory and structure of Artificial Network (ANN) and how to train it. It's multilayer learning structure is the basic structure of the deep learning models. In Section 3.2, Chi-square test is introduced, which is used to select important bag-of-words which are most related to the trend of stock prices.

2.1 Financial Markets

Financial markets are generally used to describe markets in which people trade financial securities and communities with a low transaction cost. The prices reflect the supply and demand. Stock and currency exchange are two main components of financial markets. In this project, financial markets mainly refer to the stock market. This definition of usage is quite common can be found in some research papers, such as [63, 75]. A stock market has two main components which are tradable assets and traders. The traders aim to obtain the maximum gain from the market by trading assets according to their experience. The main operation is stock exchange in the stock market.

Stock is issued by a cooperation which aims to raise money instead of having a loan from a bank. The stock is partitioned into shares which can be traded in the stock market. In return, the companies need to pay a certain amount of profit to the holders of the stock (shareholders). A shareholder has a fraction of ownership on the assets and earnings of the company. Ownership depends on the amount of shares the holder own relative to the whole shares. A cooperation may declare distinct types of shares. Each type of shares can have a specific share value or privileges. Additionally, the per-value price of a stock can change continuously. Some shareholders sell the shares when the price of stock is higher to earn money. Thus it is important to estimate the future trend of the stock price for the traders. The traders can estimate the stock price based on many criteria such as the trading vol-

ume, the previous trend of stock price, important news about the company, public opinions on the social network and so on. Share prices change according to the change of supply and demand, and there is a mechanism to associate them. This sort of trading is more risky than earning form annual profit, because the prices are highly unpredictable.

Algorithmic trading (AT) generally refer to a trading system which performs all or some parts of the trade cycle automatically in the financial market by heavily replying on complex mathematical models and sophisticated computer program [76]. It consists of six stages which are pre-trade analysis, trading signal prediction, trade operation, risk management, post-trade analysis and asset allocation. Each of them could be a research field of the AT, and in this thesis, the trading signal generation is our main research area. The traditional stock prediction is based on historical data and trading volume. In this study, financial textual articles are used to predict the trend of stock prices.

2.2 Data Sources

The amount of available textual data related to stock markets is huge, including various forms such as government and shareholder reports or financial news concerning the prospect of a company. Some examples of financial article source data are shown in Table 2.1. [19]

Table 2.1: Some examples of financial article source data [19]

Textual Source	Types	Examples	Description
Company Generated Sources	SEC Reports	8K	Reports on significant changes
		10K	Annual reports
Independently Generated Sources	Analyst Created	Recommendations	Buy/Hold/Sell assessments
		Stock Alerts	Alerts for share prices
	News Outlets	Financial Times	Financial News stories
		Wall Street Journal	Financial News stories
	News Wire	PRNewsWire	Breaking financial news articles
		Yahoo Finance	45 financial news wire sources
	Discussion Boards	The Motley Fool	Forum to share stock-related information

In this thesis, I chose 8-K financial reports which is used to notify the investors when there is a significant event happened such as change in accountants, elections of director, asset movement or bankruptcies (more information are included in Table 2.2). Lee et al. [48] publicly provide all reports of S&P 500 companies between 2002 and 2012 for researchers in the corpus: http://nlp.stanford.edu/pubs/stock-event.html. This corpus was used as the experiment data.

Material definitive agreements not made in the ordinary course of business
Bankruptcies or receiverships
Director is elected
Director departs
Asset movement: acquisition or sale
Result of operations and financial condition
Material Direct Financial obligations (bonds, debentures)
Triggering events that accelerate material obligations (defaults on a loan)
Exit or disposal plans
Material impairments
Delisting or transfer exchange notices
Unregistered equity sales
Modifications to shareholder rights
Change in accountant – and good idea to explain why
SEC investigations and internal reviews
Financial non-reliance notices
Changes in control of the company
Changes in executive management
Departure or appointment of company officers
Amendments to company Governance Policies
Trading suspension
Change in credit
Change in company status
Other events
Financial exhibits

Table 2.2: Type of events in 8-K reports from http://en.wikipedia.org/wiki/Form_8-K

The historical price data were extracted from *Yahoo! Financ* (https://uk.finance.yahoo.com). The data set consists of 10 years historical price information of the S&P500 companies. The information includes date, volume, opening price, closing price, adjusted closing price, high price, low price like the Table 2.3:

Table 2.3: Example of historical price data

Date	Open	High	Low	Close	Volume	Adj Clos
2013-02-01	45.08	45.66	44.54	45.29	2827800	45.2
2013-01-31	44.35	44.87	44.31	44.78	3383300	44.78
2013-01-30	44.58	44.89	44.17	44.40	2942700	44.4

2.3 Artificial Neural Network

Artificial Neural Networks (ANNs) [53] were inspired by the functionality of the 'human brain where nearly billions of neurons are connected to process the information and respond the input signal. Researchers have given computers some certain levels of intelligence such as language translation and pattern recognition by using artificial neural network. This section will give a brief review of ANNs.

2.3.1 Perceptron

The linear regression model consists of a linear combination of fixed non-linear functions with a set of fixed input data as below Eq 2.1 [11]:

$$y(\mathbf{x},\mathbf{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right) \tag{2.1}$$

where M is the number of inputs, f is a non-linear activation function and ϕ_j is a nonlinear basis function whose parameters can be adjusted along with the adjustment of coefficient w_j. Neural Networks use the similar form of Eq 2.1, which is composed of "neurons" that can be viewed as non-linear functions of a linear combination of the input data.

A single neuron consists of inputs which are multiplied by corresponding weights and then combined a non-linear function usually called *activation function* that generates the output. The graphical structure is shown in in Figure 2.1 .

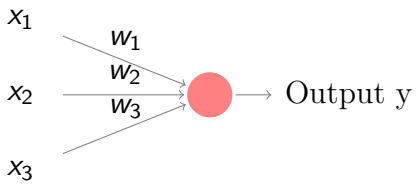

Figure 2.1: An artificial neuron

The mathematical expression of a single neuron for the j^{th} neuron is shown below in Eq 2.2. It represents a linear combination of inputs from x_1 to x_D passed to the activation function f, which is usually the *sigmoid* or *tanh* function.

$$a_j = f\left(\sum_{i=1}^{D} W_{ji}^{(1)} x_i + x_0{}^{(1)}\right) \tag{2.2}$$

where D means the number of input, j means it is the j^{th} neuron and the subscript (1) means it is the first layer, so $w_{ji}^{(1)}$ means the weight between the i^{th} input and neuron j in the first layer, and $x_0^{(1)}$ means the bias.

2.3.2 Multilayer Perceptron

Multilayer Perceptron [64] is a more sophisticated model with multiple layers, which consists of an input layer, an output layer and many hidden layers. Its graphical structure is shown in Figure 2.2. It is a type of network, whose nodes in the same layer do not connect to each other. There is also no loop in the entire model. In addition, it can be viewed as a feed-forward network.

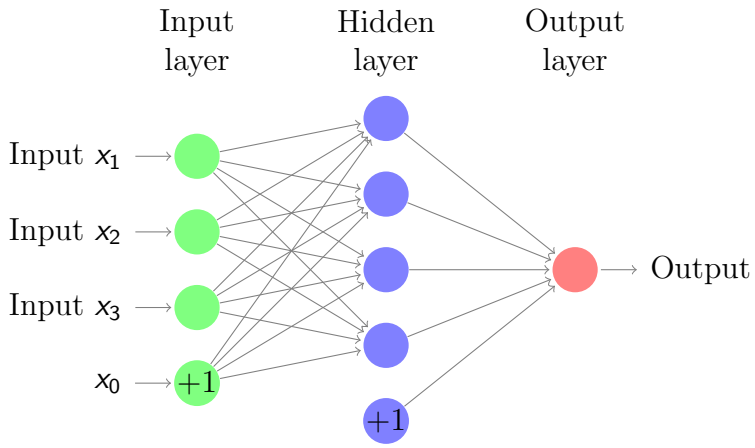

Figure 2.2: Graphical Structure of a typical Neural Network

There might be multiple hidden layers, and the output of the hidden layer cells h_j corresponds to an input of the next layer, with output vector $\mathbf{h}^{(b-1)}$ of the layer $b-1$ as the input of layer b. The output $\mathbf{h}^{(b)}$ can be computed from the weight matrix $\mathbf{W}^{(b)}$ and bias vector $\mathbf{z}^{(b)}$, using Eq 2.3.

$$\mathbf{h}^b = f(\mathbf{W}^{(b)}\mathbf{h}^{(b-1)} + \mathbf{z}^{(\mathbf{b})}) \tag{2.3}$$

The output layer is used to make a prediction $\mathbf{h}^{(k)}$ and the loss can be computed with a loss function $L(\mathbf{h}^{(k)}, y)$ combined with the actual label y. Each unit in the output layer can use a logistic sigmoid function shown in Eq 2.4 as the activation function for multiple binary classification problems.

$$\sigma(a) = \frac{1}{1 + exp(-a)} \tag{2.4}$$

The negative conditional log-likelihood (NLL) is often used to compute the loss, which is $L(\mathbf{h}^{(k)}, y) = \log P(Y = y|x)$. NLL is expected to be minimized.

2.3.3 Back-propagation Algorithm

The back-propagation algorithm was introduced in [79, 65]. The weight matrices should be initialized randomly first by a Gaussian distribution as the following formula:

$$G(W_{ij}, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(W_{ij}-\mu)^2}{2\sigma^2}} \tag{2.5}$$

where μ and σ are the mean value and variance of the Gaussian distribution respectively. Then the feed-forward phase is executed to compute all the activations and output of each layer. The difference between the output and target value are calculated as cost error. Next the back-propagation phase is fired. In this process, the error will be back propagated to each layer to adjust the weights. Then feed-forward and back-propagation will be further iterated until the error converges to a certain desired level.

Assume we are given the following training data:

- $\mathcal{D} = (X, Y)$
- $X = [\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(n)}] \in R^{n*m}$ is a set of input data of the model, in which $\boldsymbol{x}^{(i)}, (i \in |\mathcal{D}|)$ is a single input instance.
- $Y = [\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, ..., \boldsymbol{y}^{(n)}] \in R^{n}$ is the label set of the training data, in which $\boldsymbol{y}^{(i)}, (i \in |\mathcal{D}|)$ is a single target value the model should output.

After the feed-forward process, we obtain the weight matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$, and $\theta = [\boldsymbol{W}, \boldsymbol{b}]$. Using these, the input data X and activation function, we can obtain the predicted values $\widehat{Y}$.

The cost between prediction output and target label can be calculated with the average value of the negative log-likelihood (NLL) of the prediction under the target distribution Y, and the regularization term which controls the magnitudes of weights to prevent from overfitting, according to Eq. 2.6, 2.7

$$cost = \frac{1}{|\mathcal{D}|}\mathcal{L}(\theta = \{\boldsymbol{W}, \boldsymbol{b}\}, \mathcal{D}) + regularization \tag{2.6}$$

$$= \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} \log(P(Y = \hat{\boldsymbol{y}}^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{W}, \boldsymbol{b})) + \frac{\lambda}{2}\sum_{i=1}^{l} |W_i|_2^2 \tag{2.7}$$

where λ is the regularization coefficient and l is the number of layers of the network. The above function can be minimized by the back-propagation algorithm. For each

layer i, we calculate the output $\boldsymbol{h}^{(i)}$ from input $\boldsymbol{h}^{(i-1)}$ as equation 2.3. Then perform back-propagation. For the output layer, compute:

$$\boldsymbol{\delta}_l = -(\boldsymbol{y_j} - \boldsymbol{h}^{(l)}) \odot f'(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \tag{2.8}$$

where $\odot$ means the element-wise multiplication. f' means the derivative of activation function f. For layer $i \in [2, l-1]$, compute the derivative:

$$\boldsymbol{\delta_i} = (\boldsymbol{W}_i\boldsymbol{\delta}_{i+1}) \odot f'(\mathbf{W}^{(i)}\mathbf{h}^{(i-1)} + \mathbf{b}^{(\mathbf{i})}) \tag{2.9}$$

Finally, given learning rate η, layer i, weight matrix and bias vector can be updated by the following formula with $\boldsymbol{\delta}_{i+1}$:

$$\boldsymbol{W}^{(i)}_{new} = \boldsymbol{W}^{(i)}_{old} - \eta(\boldsymbol{h^i} * \boldsymbol{\delta_{i+1}} + \lambda \boldsymbol{W}^{(i)}) \tag{2.10}$$

$$\boldsymbol{b}^{(i)}_{new} = \boldsymbol{b}^{(i)}_{old} - \eta\boldsymbol{\delta}_{i+1} \tag{2.11}$$

‘

2.4 Deep Learning

Nowadays, the performances of machine learning models heavily rely on the representation of data or feature selection steps rather than just the choice of machine learning algorithms. Thus much effort is applied on preprocessing pipelines such as feature selection. Even though some specific domain knowledge can be used to help design the representation of data, the motivation of Artificial Intelligence needs more powerful representations of features. Deep Learning also called unsupervised learning, is a relatively new research field of machine learning which can learn multiple levels of abstraction and representation of features directly from data. It aims at learning feature hierarchies with higher level features formed by the composition of lower ones. The multiple levels structures allow to build complex functions which take data as input and output the result directly without depending on features crafted by humans [4].

Deep learning achieved many successful results on some problems, such as image classification [17, 41], semantic parsing [13] and speech recognition [20]. Deep architecture may express the complex distributions more efficiently with better performance on challenging tasks [8, 4]. The hypothesis that the composition of additional functional levels can give more powerful modeling capacity has already been proposed for a long time [28, 66]. However, the training process of deep architecture was proven to be very difficult, until some successful approaches of [7, 30, 29] for training stacked autoencoder and DBN occurred. One key idea behind them is to train the deep architecture layer by layer by unsupervised learning, which is also called unsupervised feature learning. Each layer will generate a more abstract representation of the observed layer by doing a local optimization. Unsupervised feature learning can learn useful features automatically and directly from the data set by unsupervised learning without given specific features defined by human. The

unsupervised learned features are more natural with less information lost [6]. Some deep learning models also have a potential powerful capacity of solving time-series problems [42], which is another reason that makes deep learning suitable for stock trend prediction. Therefore deep learning can provide a new potential and powerful approach to improve stock prediction.

2.5 Building Deep Representations

Experimental results show that it is much harder to train a deep architecture than training a shallow one [8, 22]. The rapid recent growth and success of deep learning owe to a breakthrough initiated by Geoff Hinton in 2006 and quickly followed up by some papers [29, 8, 61]. Greedy layer wise unsupervised pre-training as a central idea was proposed. It means just one layer of the hierarchy is trained at one time by unsupervised feature learning to learn a new transformation of data with the previous layer output. Finally, the set of pre-trained layers is combined with a standard supervised classifier such as Support Vector Machine, Logistic Regression or as initialization for a deep learning model such as a Stacked Denoising Auto-encoder or Deep Belief Network. Experiments show that the layer-wise stacking can attain a better feature representation in most time [43]. Although it is not difficult to combine single layers pretrained by unsupervised learning into a supervised model, it is not very clear how the single layers should combine to form a better unsupervised model [5]. One approach is to stack pre-trained RBMs (section 4.2.1) into DBN (section 4.2.2).

In this chapter, we briefly introduced the relevant theoretical background. Next two chapters, we will present the design and methodology used in this project on textual article pre-processing and the deep learning models we built. We first move to the methodology used for the data pre-processing stage.

Chapter 3

Data Pre-processing and Analysis

In this project, we used financial reports with natural language processing techniques and three important deep learning models, respectively, to build stock market prediction models. For the text-mining process, two issues were considered: (1) appropriate news article with relevant textual data content and (2) an approach to associate the mining textual data with historical time-series prices [35]. First we had to find appropriate news articles. Lee et al. [48] generated and released a corpus that aligns the financial events of all S&P companies between 2001 and 2013 with changes in stock prices, which can be used as the source data for this project. The aligned data focus on the 8-K reports which listed American companies are required to file when ever a significant event happens [48]. Second we associated the textual financial events with the time series price data. In this stage, several Natural language processing and information retrieval techniques were used, for example Tf-idf and Chi-square test [52].

Natural language processing and information retrieval techniques are crucial to the success of this thesis. It is important to get the most useful text information from the data set. It is also necessary to transform the price information to the predicted. Two types of labels are used to study the performance on difference types of predictions. One is the effect of the report directly on the stock price and the other one is the effect on the stock indicators. All of them together constitute the preprocessing step. In section 3.1, the way we normalize the text in to the same standard is introduced. Section 3.2 presents the method we used to select the bag-of-words as training data. In section 3.4, we talk about how did we unifies the different articles in the same day. The normalization method we used on the training data is described in section 3.5. Finally we discuss the transformation price data processing steps to different types of training labels in section 3.6.

3.1 Normalizing Text

In order to use the information of textual articles, we need to transform them to word vectors. First we need to preprocess the raw textual data in a normalized form.

Some not useful unigrams such "<DOCUMENT>" and "FILE:" are removed from the files. After that each word should be lemmatized to be reduced to the stem and prevent from duplicated words in different forms.

Stemming and lemmatization (or lemmatization) in linguistics and information retrieval is the process of transform the inflected form to its root or stem form. For example for grammar reason, different forms of words with similar meanings such as fly, flies and flying are often used in documents. The aim is to reduce the inflectional forms and associate the various forms to the base word [52], for example:

$$\text{dog, dogs, dog'} \Rightarrow \text{dog}$$
$$\text{am, is, are} \Rightarrow \text{be}$$

For example, if the input text is : "In addition, the Board approved cancelling the Company's non-shareholder approved option plan upon the completion of the employee exchange program and shareholder approval to amend the remaining shareholder-approved executive officer stock plan so it can be used to grant options to all employees." (from 8-K report).

The result will be: "In addition , the Board approve cancel the Company non-shareholder approve option plan upon the completion of the employee exchange program and shareholder approval to amend the remain shareholder-approved executive officer stock plan so it can be use to grant option to all employee ."

Then every words in the article were translated to the lower case. The digit numbers and some stop words were removed such as I, have, a, do, his, to... which are not meaningful for the classification. The result of the example above will become to: " addition board approve cancel company non-shareholder approve option plan completion employee exchange program shareholder approval amend remain shareholder-approved executive officer stock plan use grant option employee"

3.2 Text Feature Selection

Feature selection, also known as variable selection, is the process of selecting a subset of relavant features to train the learning model [26]. For the text classification, feature selection often refers to selecting the terms occurring in the training set as features to do text classification with two main purposes [52]. First it can decrease the size the of training vocabularies, so the effectiveness of training can be improved with a much shorter training time. Second, it can eliminate some noise data which may have a negative effect on the training result to increase the accuracy and avoid overfitting. Text feature selection can be viewed as a process of replacing a complex classifier training on all words occurred with a simpler one training only on selected relatively useful words. The text feature selection in this project refers to bag-of-word selection. The basic text feature selection algorithm is shown in Algorithm 1. To select text features, we need to extract all the unique vocabularies in a set first, without duplicated words:

- {management, improve, ... , phone}

Then for each class and each term in the vocabulary set, the utility score of the term were computed for the given class. Then store the scores with term as a key in a feature dictionary (or map, table):

- {'four': 0.196, 'bookrunning': 8.478, 'increase': 10.811, ... , 'eligible': 4.236}

The feature dictionary were sorted by comparing those scores in a decreasing order. Then take the top 1000 words in the feature dictionary as the selected features, and discard the other terms not used in the classification.

- {'effectively' : 55.491, 'combine' : 50.727, 'entirely' : 48.416, ... , 'priced' : 9.057}

The function ComputeUtilityScore is a most significant part, because the bag-of-words are selected based on this returned value. Mutual information and Chi-square are two most popular algorithms in information retrieval, which can be used as ComputeUtilityScore function. In this project, Chi-square [52] was chosen as the feature selection algorithm.

Algorithm 1 FeatureSelection

```
1: procedure selectFeatures(documents, class, number)
2:     vocabulary ←extractVocabulary(documents)
3:     featureList ← {}
4:     for each term ∈ vocabulary do
5:         score_{term,class} ← ComputeUtilityScore(document, term, class)
6:         Append(featureList,{term : score_{term,class}})
7:         SortFeaturesWithValueInDecreasingOrder(featureList)
8:     end for
9:     return TopLargestFeatures(featureList,number)
10: end procedure
```

3.3 Chi-square

In this project, the Chi-square test is used to select related words which had most significant effects on the stock price for each stock. Chi-square denoted χ^2 [52] is a popular feature selection approach in information retrieval. In statistics, χ^2 test is used to test whether two events A and B are independent, which means:

$$P(AB) = P(A)P(B) \tag{3.1}$$

In the feature selection step of information retrieval, the two events are the occurrence of term and occurrence of class. Hence it means we need to calculate the

degree of relationship between a term and a document class. The value can be computed using Eq 3.2, then we can rank the unique term t by the quantity $\chi^2(\mathcal{D}, c, t)$ for all classes.

$$\chi^2(\mathcal{D}, c, t) = \sum_{z_c \in \{0,1\}} \sum_{z_t \in \{0,1\}} \frac{(O_{z_t z_c} - E_{z_t z_c})^2}{E_{z_t z_c}} \tag{3.2}$$

where $\mathcal{D}$ is a document set which contains all documents. t means a term, c means a class. $z_t = 1$ means the term occurred in a document and $z_t = 0$ means the document do not contain the term. $z_c = 1$ means the document belongs to the class c and $z_c = 0$ means the document is not in class c. $O_{z_t z_c}$ is the observed occurrence times in $\mathcal{D}$ and $E_{z_t z_c}$ is the expected frequency. For example, O_{11}, means how many documents contains t and belongs to the class c and O_{01} means how many documents do not contain t but belongs to the class c. The E_{11} can be calculated by the following formula Eq 3.4 below:

$$E_{11} = O * P(t) * P(c) \tag{3.3}$$

$$= O * \frac{O_{11} * O_{10}}{O} * \frac{O_{11} * O_{01}}{O} \tag{3.4}$$

Some implementation details are shown in section 5.3

3.4 Same Day Textual Data Compression

As mentioned before, a training example is a vector of a specific size, which consists of 0 and 1. Each attribute of data represents whether a specific important word has been occurred in the corresponding article. For a specific vector, the amount of occurred important words can be very small, which led to the training vector very spare, which means that there is many 0s with a small amount of 1s. In addition, there may be several articles in the same day, but we only can get one financial prediction for that day. Hence the word vectors in the same day need to be compressed and unified into one input vector. Boureau et al. [15] used theoretical analysis to show that feature pooling can transform the local or global 'bag of features' to more compact and usable representation in a way that retain the useful information while discarding unassociated details . Lavrenko et.al [47] shows a spatial pyramid pooling model can increase the performance by pooling rather than a plain bag of feature of a whole image [84], which also shows the importance of the spacial structure after pooling. Jarrett [36] have experimentally shown that pooling plays an very important role in the unsupervised pre-training steps if the training data is not large enough, and good results can be attained when suitable pooling apporach is used. In this project, the word vectors of a day can be viewed as an image, which need to be unified into one feature vector. We used two pooling methods which are average-pooling (Equation 3.5) and max pooling (Equation 3.6) to unify the text word vectors.

$$f_a(\mathbf{v}) = \frac{1}{n} \sum_{i=1}^{n} v_i \tag{3.5}$$

$$f_m(\mathbf{v}) = max_i v_i \tag{3.6}$$

The implementation details of this section are introduced in section 5.4.

3.5 Preprocessed Data Normalization

After pooling, we can obtain a matrix in which each row represent a feature vector related to a specific document. In order to be able to compare different documents, each row should be normalized into the same standard, which can be done with the Tf-idf weighting scheme.

3.5.1 Tf-idf

Tf-idf [72] is an abbreviation of *term frequency-inverse document frequency*, which is a weight widely applied in information retrieval tasks. The weighting scheme measures the importance of a word in a selected dictionary to a document in proportion to the number of times a term occur in a document but inversely proportional to the frequency of documents with the term in it across the whole corpus.

3.5.2 IDF: Inverse Document Frequency

IDF measures the importance of a term. For example, some word such as 'financial', '8k' may appear many times but have not much importance. Hence it is necessary to rescale the weight of each word. Each value of the word should multiply by IDF. The formula of idf can be represented as Equation 3.7:

$$idf(t, d, D) = \log \frac{|D|}{|\{d \in D | t \in d\}|} \tag{3.7}$$

where t represents the terms, d means a feature vector of a document and D is the whole document set which consists of feature vectors of all documents. $|D|$ means the cardinality of the set of whole documents. $|\{d \in D | t \in D\}|$ means the number of documents d with term t in it. Then we need to normalize each value of the feature vectors and scaled by idf via using Equation 3.8.

$$R(t, d, D) = \frac{d_t - \min_{d_i \in \mathbf{d}} d_i}{\min_{d_i \in \mathbf{d}} d_i - \max_{d_i \in \mathbf{d}} d_i} * idf(t, d, D) \tag{3.8}$$

3.6 Historical Data processing

The price change for a single day is not sufficiently representative because it contains too little information for a time series data set, hence some financial indicators

are needed. This is because, using the indicators, we can understand the trend of stock price in long or short term by calculating recent changes of corresponding indicators[18]. Moving Average Convergence Divergence (MACD) [3] for each day can be computed and then used to predict the 3next day's MACD to better evaluate the inflections of 8-k reports to the stock market. Indicators are also useful to visualize the stock trend and market information.

The indicators Implemented in this project are presented next:
Simple Moving Average (SMA) & Exponential Moving Average (EMA)

Moving Average is one of the most widely used stock indicators, because it is easy to be constructed, quantified, understood and tested. The Simple Moving Average (SMA) calculates the average price of a stock over a specific period. SMA [57] can be calculated by equation 3.9, in which i is the price index of current day, n is the length of the time period and p with index j means the price of day j.

$$SMA(i, n, \mathbf{p}) = \frac{1}{n} \sum_{j=i-n+1}^{i} \mathbf{p}_j \tag{3.9}$$

But there are two criticisms questioning its usefulness. One is that only the specific period is taken into account. Another one is that every day's price is given the same weight during the period, but some analysts think the recent price should be given a heavier weighting.

Exponential Moving Average (EMA) [57] is similar to SMA except it addresses the two problems associated with SMA. First, a heavier weight is given to more recent price actions, which means EMA is a weighted moving average. With less importance assigned to the previous price data, it does include calculation in all the data through the whole life of the instrument . The calculation function is given by Equation 3.10.

$$\begin{aligned} EMA(i, n, \mathbf{p}) &= \mathbf{p}_i * k + EMA(i-1, n, \mathbf{p}) * (1-k) \\ k &= \frac{2}{n+1} \end{aligned} \tag{3.10}$$

Moving Average Convergence/Divergence (MACD):

MACD [3] was developed by Gerald Appel. It measures the difference between a long-term and short-term EMA with a dual moving average crossover approach. There two lines can be displayed on the screen. The faster line (MACD line) is calculated by the difference of two EMA of adjusted closing price, usually are 12 and 26 days, originally designed for buy signal. Convergence signal will occur when MACD closed to zero and divergence signal when MACD moved far way from zero. And the slower line (Signal line) can be calculated by a 9 period EMA of MACD originally designed for sell signals. However, they are usually in all instances by most users. The MACD histogram is the difference between the MACD line and signal line. The calculation formula can be displayed in the Equation 3.11

$$MACD(i) = MACD_{line}(i, \mathbf{p})) - Signal_{line}(i) \tag{3.11}$$

where

$$MACD_{line}(i, \mathbf{p}) = EMA(i, 12, \mathbf{p}) - EMA(i, 26, \mathbf{p}) \tag{3.12}$$

$$Signal_{line}(i) = EMA(i, 9, MACD_{line}(i, \mathbf{p})) \tag{3.13}$$

Relative Strength Index (RSI)

The RSI was first developed in [82]. It is a momentum oscillator that measures the seed of movement of stock prices which are scaled within 30 to 70. If the index is greater than 70, it will be considered as overbought, instead if the index is less than 30, it will be considered as oversold, and then a signal of trend reversal will be given. The actual formula is calculated as Equation 3.14:

$$\begin{aligned} RSI &= 100 - 100/(1 + RS) \\ RS &= \frac{\text{Average of x days' up closes}}{\text{Average of x days' down closes}} \end{aligned} \tag{3.14}$$

This chapter introduced the methods we used in pre-processing steps, including the textual articles pre-processing, bag-of-words selection, same day article unification, data normalization, and historical price data processing. In Chapter 4, we introduce three deep learning models we adapted to the stock price prediction problem.

Chapter 4

Deep Learning Models

Deep learning have been introduced in section 2.4 and 2.5. In this thesis, three deep learning models were adapted to the context of the stock trend prediction problem to investigate their suitability and efficiency. They are Stacked Denoising Autoencoders (SDA), Deep Belief Network (DBN) and Recurrent Neural Networks-Restricted Boltzmann Machine(RNN-RBM) [14]. During the implementation of the deep learning models, *theano* [10] was used, which is a python library allowing the user to efficiently define mathematical expressions involving multi-dimensional arrays on GPU, and prototype new machine learning models without unnecessary implementation overhead. Two other classical machine learning methods, SVM and Random Forests, are also used to build prediction models, which would be as a comparison baseline used against the deep learning models. In this chapter we introduce the theory and implementation of the deep learning models which we adapted to the stock prediction problem.

4.1 Stacked Denoising Autoencoders

4.1.1 Autoencoders

An autoencoder [4] is a network whose graphical structure is shown in Figure 4.1, which has the same dimension for both input and output. It takes an unlabeled training examples in set $X = \{\boldsymbol{x^1}, \boldsymbol{x^2}, ..., \boldsymbol{x^n}\}$ where $\boldsymbol{x^i} \in [0,1]^d$ is a single input and encodes it to the hidden layer $\boldsymbol{h} \in [0,1]^z$ by linear combination with weight matrix $\boldsymbol{W}$ and then through a non-linear activation function. It can be mathematically expressed as $\boldsymbol{h} = a(\boldsymbol{Wx} + \boldsymbol{b})$, where $\boldsymbol{b}$ is the bias vector.

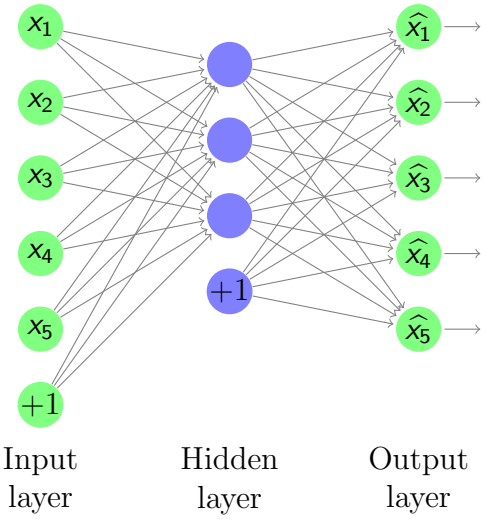

Figure 4.1: An Autoencoder

After that the hidden layer representation will be reconstructed to the output layer $\widehat{\boldsymbol{x}}$ through a decoding function, in which $\widehat{\boldsymbol{x}}$ has a same shape as $\boldsymbol{x}$. Hence the decoding function can be mathematically expressed as $\widehat{\boldsymbol{x}} = a(\boldsymbol{W'h} + \boldsymbol{b_h})$, where $\boldsymbol{W'}$ can be $\boldsymbol{W'} = \boldsymbol{W^T}$ called tried weights. In this project, tied weights were used. The aim of the model is to optimize the weight matrices, so that the reconstruction error between input and output can be minimized. It can be seen that the Autoencoder can be viewed as an unsupervised learning process of encoding-decoding: the encoder encodes the input through multi-layer encoder and then the decoder will decode it back with the lowest error [30].

To measure the reconstruction error, traditional squared error $\boldsymbol{L(x, \widehat{x}) = \|x - \widehat{x}\|}$ can be used. One of the most widely used way to measure that is the cross entropy if the input can be represented as bit vector or bit possibilities. The cross entropy error is shown in Equation 4.1:

$$\boldsymbol{L(x, \widehat{x})} = -\sum_{i=1}^{d} [\boldsymbol{x}_i log \widehat{\boldsymbol{x}}_i + (1 - \boldsymbol{x}_i) log(1 - \widehat{\boldsymbol{x}}_i)] \tag{4.1}$$

The hidden layer code $\boldsymbol{h}$ can capture the information of input examples along the main dimensions of variant coordinates via minimizing the reconstruction error. It is similar to the principle component analysis (PCA) which project data on the main component that captures the main information of the data. $\boldsymbol{h}$ can be viewed as a compression of input data with some lost, which hopefully not contain much information about the data. It is optimized to compress well the training data and have a small reconstruction error for the test data, but not for the data randomly chosen from input space.

4.1.2 Denoising Autoencoders

In order to prevent the Autoencoder from just learning the identity of the input and make the learnt representation more robust, it is better to reconstruct a cor-

rupted version of the input. The Autoencoder with a corrupted version of input is called a Denoising Autoencoder. Its structure is shown in Figure 4.2. This method was proposed in [78], and it showed an advantage of corrupting the input by comparative experiments. Hence we will use denoising autoencoders instead of classic autoencoders in this thesis.

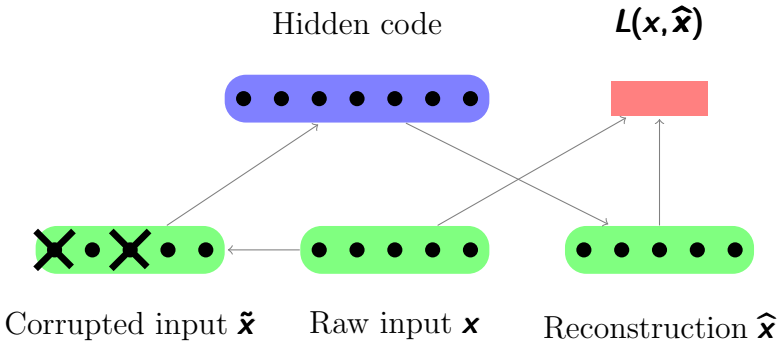

Figure 4.2: A graphical figure of Denoising Autoencoder. An input x is corrupted to $\tilde{x}$. After that the autoencoder maps it to the hidden representation h and attempts to reconstruct x.

A Denoising Autoencoder can be seen as a stochastic version with adding a stochastic corruption process to Autoencoder. For the raw inputs x, some of them will be randomly set to 0 as corrupted inputs $\tilde{x}$. Next the corrupted input $\tilde{x}$ will be encoded to the hidden code and then reconstructed to the ouput. But now $\hat{x}$ is a deterministic function of $\tilde{x}$ rather than x. As Autoencoder, the reconstruction is considered and calculated between $\hat{x}$ and x noted as $L(x, \hat{x})$. The parameters of the model are initialized randomly and then optimized by stochastic gradient descent algorithms. The difference is that the input of the encoding process is a corrupted version $\tilde{x}$, hence it forces a much more clever mapping than just the identity, which can denoise and extract useful features in a noise condition.

The training algorithm of a denoising autoencoder is summarized in Algorithm 2.

Algorithm 2 Denoising Autoencoder Training

procedure DA TRAINING(e,b,$\boldsymbol{x}$,c,l,$\boldsymbol{\theta}$)
  $\boldsymbol{x} = [x_1, x_2, ...x_n] \in R^{n*m}$ is the input matrix, in which $x_i \in [0,1]^m$ $(1 \leq i \leq m)$ is a single input data
  e is the amount of epochs to be iterated
  b is the amount of batches
  l is the learning rate
  c is the corruption level
  $\boldsymbol{\theta} = \{W, \boldsymbol{b}, \boldsymbol{b_h}\}$ where $W \in R^{n*d}$, $b \in R^d$, $b_h \in R^d$, $\boldsymbol{\theta}$ is the parameters of a DA
  for 0 to e **do**
    for 0 to b **do**
      $\tilde{\boldsymbol{x}}$ = getCorruptedInput($\boldsymbol{x}$,c), in which c is the corrupted level
      $\boldsymbol{h} = sigmoid(\tilde{\boldsymbol{x}} * W + \boldsymbol{b})$
      $\hat{\boldsymbol{x}} = sigmoid(\boldsymbol{h} * W^T + \boldsymbol{b_h})$
      $\boldsymbol{L(x, \hat{x})} = -\sum_{i=1}^{d} [\boldsymbol{x}_i log\hat{\boldsymbol{x}}_i + (1 - \boldsymbol{x}_i) log(1 - \hat{\boldsymbol{x}}_i)]$
      cost = mean($\boldsymbol{L(x, \hat{x})}$)
      $\boldsymbol{g}$ = compute the gradients of the cost with respect to $\boldsymbol{\theta}$
      for θ_i, g_i in $(\boldsymbol{\theta}, \boldsymbol{g})$ **do**
        $\theta_i = \theta_i - l * g_i$
      end for
    end for
  end for
end procedure

4.1.3 Stacked Autoencoder

Unsupervised pre-training

A Stacked Autoencoder is a multi-layer neural network which consists of Autoencoders in each layer. Each layer's input is from previous layer's output. The greedy layer wise pre-training is an unsupervised approach that trains only one layer each time. Every layer is trained as a denoising autoencoder via minimising the cross entropy in reconstruction. Once the first i^{th} layer has been trained, it can train the $(i+1)^{th}$ layer by using the previous layer's hidden representation as input. An example is shown below. Figure 4.3 shows the first step of a stacked autoencoder. It trains an autoencoder on raw input $\boldsymbol{x}$ to learn $\boldsymbol{h_1}$ by minimizing the reconstruction error $\boldsymbol{L(x, \hat{x})}$.

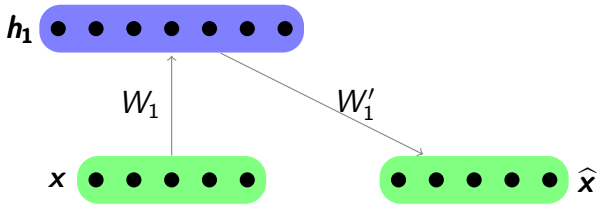

Figure 4.3: Step 1 in Stacked Autoencoders

Next step shown in Figure 4.4. The hidden representation $\boldsymbol{h_1}$ would be as "raw input" to train another autoencoder by minimizing the reconstruction error $\boldsymbol{L(h_1, \widehat{h_1})}$. Note that the error is calculated between previous latent feature representation $\boldsymbol{h_1}$ and the output $\boldsymbol{\widehat{h_1}}$. Parameters W_2 and W_2' will be optimized by the gradient descent algorithm. The new hidden representation h_2 will be the 'raw input' of the next layer.

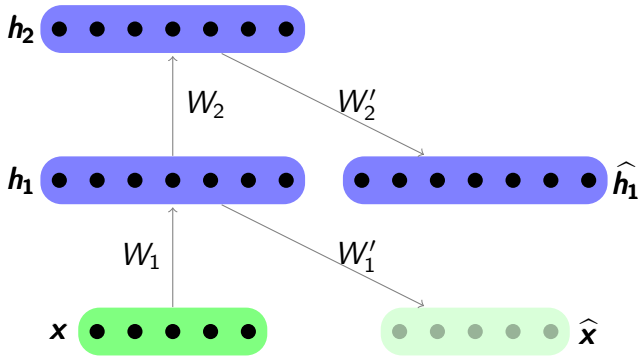

Figure 4.4: Step 2 in Stacked Autoencoders

The pre-training algorithm of stacked denoising autoencoder is summarized in algorithm 3.

Algorithm 3 Unsupervised preTrain SDA

procedure PRETRAINING(e,b,$\boldsymbol{X}$,c,l,$\boldsymbol{\theta}$)
  $\boldsymbol{X = [x_1, x_2, ...x_n]} \in R^{n*m}$ is the input matrix, in which $x_i \in [0,1]^m$ $(1 \leq i \leq m)$ is a single input data
  l is the learning rate
  $\boldsymbol{h} = [h_1, h_2, ..., h_z] \in Z^l$, where h_i is the number of hidden units in layer i and l is the number of hidden layers.
  $\boldsymbol{c} = [c_1, c_2, ..., c_z] \in (0,1)^z$, in which c_i is the corruption level of the input of the hidden layer i.
  $\boldsymbol{\Theta} = [\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, ..., \boldsymbol{\theta_z}]$, where $\boldsymbol{\theta_i} = \{W_i, \boldsymbol{b_i}, \boldsymbol{b_{hi}}\}$
  $\boldsymbol{O = [O_1, O_2, ..., O_l]}$ is the output of each hidden layer, where $\boldsymbol{O_i = [o_{i,1}, o_{i,2}, ..., o_{i,n}]} \in R^{n*h_i} (0 < i < l)$
  $\boldsymbol{\theta_1}$ = DA TRAINING(e,b,$\boldsymbol{x}$,c,l,$\boldsymbol{\theta_1}$)
  for i from 1 to n **do**
    $\boldsymbol{o_{1,i}} = sigma(\boldsymbol{x_i W_1} + \boldsymbol{b_i})$
  end for
  for j from 2 to l **do**
    $\boldsymbol{\theta_j}$ = DA TRAINING(e,b,$\boldsymbol{O_{j-1}}$,c,l,$\boldsymbol{\theta_j}$)
    for i from 0 to n **do**
      $\boldsymbol{o_{ji}} = \sigma(\boldsymbol{o_{j-1,i} W_j} + \boldsymbol{b_j})$
    end for
  end for
end procedure

Supervised fine-tuning

At last once all the layers has been pre-trained, the next step called fine-tuning is performed. A supervised predictor will be extended to the last layer, such as support vector machine or a logistic regression layer. In this project, we chose a logistic regression layer. After that the network will be trained. A sample graph is shown in Figure 4.5. It can be seen that for each layer of the network, only the encoding hidden representation $\boldsymbol{h_i}$ ($i \in N$) are considered. The fine-tuning step will train the whole network by back-propagation like training an Artificial Neural Network. A stacked denoising autoencoder is just replace each layer's autoencoder with denoising autoencoder whilst keeping other things the same.

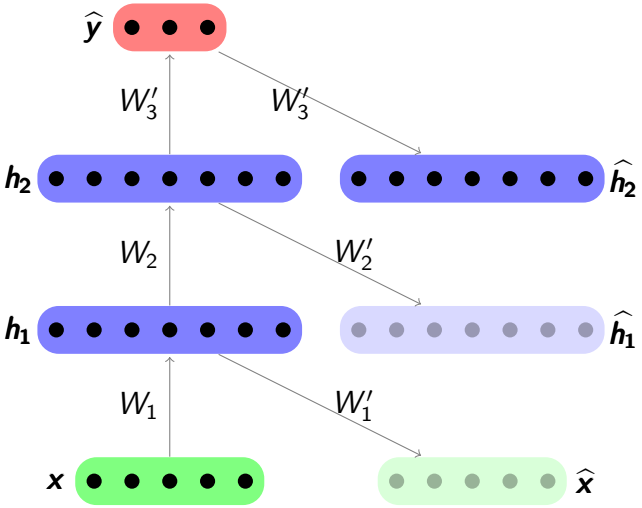

Figure 4.5: A complete architecture of stacked autoencoder

The supervised fine-tuning algorithm of stacked denoising auto-encoder is summarized in Algorithm 4.

Algorithm 4 Supervised FineTuning SDA

1: **procedure** FINETUNING(e,b,$\boldsymbol{X}$,c,l,$\boldsymbol{\theta}$)
2: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ...\boldsymbol{x_n}] \in R^{n*m}$ is the input matrix, in which $x_i \in [0,1]^m$ ($1 \leq i \leq m$) is a single input data
3: l is the learning rate
4: $\boldsymbol{h} = [h_1, h_2, ..., h_z] \in Z^l$, where h_i is the number of hidden units in layer i and l is the number of hidden layers.
5: $\boldsymbol{c} = [c_1, c_2, ..., c_z] \in (0,1)^z$, in which c_i is the corruption level of the input of the hidden layer i.
6: $\boldsymbol{\Theta} = [\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, ..., \boldsymbol{\theta_z}]$, where $\boldsymbol{\theta_i} = \{\boldsymbol{W_i}, \boldsymbol{b_i}, \boldsymbol{b_{hi}}\}$
7: $\boldsymbol{O} = [\boldsymbol{O_1}, \boldsymbol{O_2}, ..., \boldsymbol{O_l}]$ is the output of each hidden layer, where $\boldsymbol{O_i} = [\boldsymbol{o_{i,1}}, \boldsymbol{o_{i,2}}, ..., \boldsymbol{o_{i,n}}] \in R^{n*h_i} (0 < i < l)$
8: logLayer = LogisticRegression
9: **for** epoch from 0 to e **do**

10: $cost = \frac{1}{|\mathcal{D}|}\mathcal{L}(\theta = \{W, b\}, \mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{i=0}^{|\mathcal{D}|} \log(P(Y = y^{(i)}|x^{(i)}, W, b))\ell(\theta = \{W, b\}, \mathcal{D})$

11: g = compute the gradient of *cost*with respect to $\boldsymbol{\theta}$

12: **for** θ_i, g_i in $(\boldsymbol{\theta}, \boldsymbol{g})$ **do**

13: $\theta_i = \theta_i - l * g_i$

14: **end for**

15: validationLoss= $\frac{1}{n}\sum_{i=i}^{n} l(\hat{y}_i, y_i)$

16: **if** *validationLoss* $<$ *bestValidationLoss* **then**

17: *bestEpoch* = *epoch*

18: *bestPara* = $\boldsymbol{\theta}$

19: *bestValidationLoss* = *validationLoss*

20: **end if**

21: **end for**

22: **return** *bestEpoch*, *bestPara*

23: **end procedure**

4.2 Deep Belief Networks

One of the most popular unsupervised learning models is Restricted Boltzmann Machines (RBM), which has been used for many types of data including images [29], speech representation [56] and moving ratings [68]. It was another success in building a generative model for bags of words that represents documents [31], which indicate its potential capacity to build a model to predict the stock with financial documents. The most widely usage of RBM is composed to build a Deep Belief Network (DBN). The learning process of RBM is usually contrastive divergence. Experiences are needed to decide the setting of numerical parameters, such the batch size, the learning rate, the momentum, the number of epochs to iterate, the layers of hidden units, and how many units in each hidden layer, the update methods being deterministically or stochastically and the initialization of weights and bias. It also needs to be considered the way to monitor the learning process and stop criteria. This section will introduce the key components and training method of our DBN model, based on the guide in [27].

4.2.1 Restricted Boltzmann Machines

Energy based model assigns an energy value to each possible configuration of units by an energy function. The function will be trained to make it has some properties such as making the energy of desirable configuration has energy as low as possible. Boltzmann Machines (BMs) are a type of energy model in a form of Markov Random Field whose energy function is linear for its parameters. In order to make it sufficiently powerful to express complex functions, some invisible variables as hidden units are added. The Restricted Boltzmann Machines is a restricted version of BMs, which constrain the BMs without the connections between different vision

units or connections between different hidden units. A example graph is shown in Figure 4.6

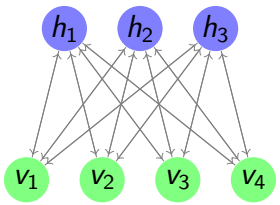

Figure 4.6: A Restricted Boltzmann Machine

The energy function is shown below:

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{j \in \text{hidden}} a_j h_j - \sum_{i \in \text{visible}} b_i v_i - \sum_{i,j} h_j v_i w_{ij} \tag{4.2}$$

where h_j v_i represent the value of the hidden and visible units. a_j is the offset of the visible layer and b_i is the offsets of the hidden layer. w_{ij} is the weights connecting the hidden and visible units. The energy model defines the probability distribution via the energy function:

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \tag{4.3}$$

where Z called partition function is a sum of all possible combinations of visible and hidden vectors:

$$Z = \sum_{\boldsymbol{v}, \boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \tag{4.4}$$

The probability of a given vector can be calculated by summing all hidden units:

$$p(\boldsymbol{v}) = \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \tag{4.5}$$

According to the free energy function of Energy-Based Models $G(\boldsymbol{v}) = -\log \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})}$, we can get the free energy of RBM model:

$$G(\boldsymbol{v}) = -\boldsymbol{b}\boldsymbol{v} - \sum_i \log \sum_{h_i} e^{h_i (a_i + W_i \boldsymbol{v})} \tag{4.6}$$

The probability assigned to a training input can be increased via adjusting W, $\boldsymbol{a}$ and $\boldsymbol{b}$ to make the energy of the training examples lower and other examples in the input space higher. Both of the visible units $\boldsymbol{v}$ and hidden units are conditionally independent of one anther, by reason of its specific structure. It can be mathematically expressed as:

$$P(h_i = 1 | \boldsymbol{v}) = sigmoid(W_i \boldsymbol{v} + a_i) \tag{4.7}$$

$$P(v_j = 1 | \boldsymbol{h}) = sigmoid(W_j \boldsymbol{h} + b_j) \tag{4.8}$$

Then the free energy of a RBM can be simplified to the following equation:

$$G(\boldsymbol{v}) = -\boldsymbol{bv} - \sum_i \log 1 + e^{a_i + W_i \boldsymbol{v}} \tag{4.9}$$

Algorithm 5 RBM training

procedure TRAINRBM(e,b,$\boldsymbol{x}$,c,l,$\boldsymbol{\theta}$,k)
    $\boldsymbol{x} = [x_1, x_2, ...x_n] \in R^{n*m}$ is the input matrix, in which $x_i \in [0,1]^m$ $(1 \leq i \leq m)$ is a single input data
    e is the amount of epochs to be iterated
    b is the amount of batches
    l is the learning rate
    k is the Gibbs steps to do
    for 1 to e **do**
        $\boldsymbol{h}$ = SAMPLEHGIVENV($\boldsymbol{x}$)
        for Scan 1 to k **do**
            $\boldsymbol{v}$ = SAMPLEVGIVENH($\boldsymbol{h}$)
            $\boldsymbol{h}$ = SAMPLEHGIVENV($\boldsymbol{v}$)
            store $\boldsymbol{v}$, $\boldsymbol{h}$ in chain of Gibbs sampling
        end for
        endChain = get the previous $\boldsymbol{v}$ in chain of Gibbs sampling
        cost = MEAN(FREEENERGY($\boldsymbol{x}$) - FREEENERGY(***endChain***))
        cost = *cost*+ the norm penalty of input
        g = compute the gradient of *cost* with respect to $\boldsymbol{\theta}$
        for θ_i, g_i in $(\boldsymbol{\theta}, \boldsymbol{g})$ **do**
            $\theta_i = \theta_i - l * g_i$
        end for
    end for
end procedure

procedure SAMPLEHGIVENV($\boldsymbol{v}$)
    $\boldsymbol{h1}$ = *sigmoid*($\boldsymbol{Wv} + \boldsymbol{a}$) (Equation 4.7)
    $\boldsymbol{sh1}$ = get a sample of hidden units given $\boldsymbol{h1}$
    return $\boldsymbol{sh1}$
end procedure

procedure SAMPLEVGIVENH($\boldsymbol{h}$)
    $\boldsymbol{v1}$ = *sigmoid*($\boldsymbol{Wh} + \boldsymbol{b}$) (Equation 4.8)
    $\boldsymbol{sv1}$ = get a sample of hidden units given $\boldsymbol{v1}$
    return $\boldsymbol{sv1}$
end procedure

procedure FREEENERGY($\boldsymbol{sv1}$)
    return $-\boldsymbol{bv} - \sum_i \log 1 + e^{a_i + \boldsymbol{W_i * sv1}}$
end procedure

4.2.2 Deep Belief Networks

Deep Belief Networks (DBNs) [30] is a greedy layer-wise form network consists of stacked RBMs, and its hierachical architecture is shown in Figure 4.7. It is also a graphical model which learns a multi-layer representation of the training examples.

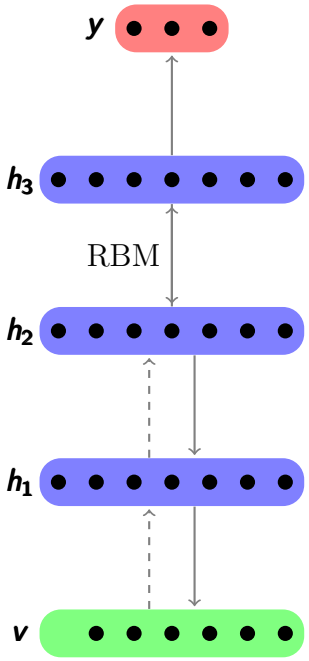

Figure 4.7: An complete architecture of stacked autoencoder

As a stacking autoencoder, an DBN is stacked with RBMs by greedy layer wise unsupervised feature learning[30, 7], and each layer is a building block trained by RBM. After the hidden layer $\boldsymbol{h^i}$ is trained, in next train step it will be as the visible layer, and the successive layer h^{i+1} will be the hidden layer. The joint distribution between input $\boldsymbol{x}$ and all hidden layers $\boldsymbol{h^i}(i \in [1, n])$ can be expressed in Equation 4.10 below. n is the number of RBM layrs. x is the first layer so it can also be viewed as $\boldsymbol{h^0}$. $P(\boldsymbol{h^{n-1}}, \boldsymbol{h^n})$ is the joint probability between visible layer $\boldsymbol{h^{n-1}}$ and hidden layer $\boldsymbol{h^n}$ which is on the top of the DBN. $P(\boldsymbol{h^{i-1}}|\boldsymbol{h^i})$ is the distribution for the visible layer $\boldsymbol{h^{i-1}}$ conditioned on layer $\boldsymbol{h^i}$ at level i.

$$P(\boldsymbol{x}, \boldsymbol{h^1}, .., \boldsymbol{h^n}) = P(\boldsymbol{h^{n-1}}, \boldsymbol{h^n}) \left(\prod_{i=1}^{n-1} P(\boldsymbol{h^{i-1}}|\boldsymbol{h^i}) \right) \tag{4.10}$$

The training process of DBN is shown in Algorithm 6

Algorithm 6 DBN Training

1: **procedure** DBNTraining
2: Set the first layer $\boldsymbol{x} = \boldsymbol{h^0}$ as visible layer prepared to train with an RBM.
3: Use the layer to obtain a hidden representation by training an RBM, which can filtered by mean activation $p(\boldsymbol{h^1} = \mathbf{1}|\boldsymbol{h^0})$.
4: Train next layer as an RBM with the input from hidden units of previous layer's RBM.
5: Iterate step 2 and 3 for a suitable times
6: Fine-tuning by supervised gradient descent.
7: **end procedure**

For the fine-tuning part in this project, a logistic regression classifier was extended to the last layer. A training label will be assigned to each input and the parameters of the DBN will be tuned by gradient descent algorithm based on the negative log likelihood cost function.

4.3 The Recurrent Neural Networks-Restricted Boltzmann Machine

The key events occurring within a company can have a profound impact on the corresponding prices of stocks. In addition, the effects on the price may be sequential and dependent. Sequential modeling is an active research area of machine learning, such as speech, words in text, music and so on. The temporal events recorded on financial reports also have the sequential feature. Recurrent neural networks (RNN) [64] have an internal memory which can store the historical sequence of training data. This property can make them suitable in modeling long-term dependencies, but the challenge is that it is not easy to train them with efficiently by gradient based optimization [9]. However, each word vector of a financial report is high-dimensional. In this case, predicting the conditional distribution of next given data of previous times is more suitable. The difficulty can be relieved by the energy based model by expressing the negative log-likelihood of a given configuration with a energy function. The RNN-RBM [14] is an energy based model which is an extension of the RTRBM [74]. The model can have a good freedom to describe the density of temporal frequency involved. It can be used to deal with the temporal effects of important financial events in a long-term period of time, which may have some degree of impacts on the stock prices. We extend the RNN-RBM by combining an DBN with two reasons. One is that component of an DBN is RBM, which makes it easy to combine DBN and RNN-RBM together. Moreover the multilayer structure can be consider to increase the power of complex function expression. This idea was introduced in [83]. In the following sections, we first had an overview of the RTRBM and then introduce the RNN-RBM. After that the RNN-RBM combined with DBN was presented.

4.3.1 The RTRBM

The RTRBM [74] is an abbreviation of the recurrent temporal restricted boltzmann machine, which is a sequence of RBMs whose parameters $\boldsymbol{a^t}, \boldsymbol{b^t}, \boldsymbol{W^t}$ are conditional on the sequence history at time t. The simple structure of the RTRBM is shown in Figure 4.8. The RTRBM can be defined as the following equation:

$$p(\{\boldsymbol{v}^t, \boldsymbol{h}^t\}) = \prod_{T}^{t=1} P(\boldsymbol{v}^t, \boldsymbol{h}^t | \{\boldsymbol{v}^k, \hat{\boldsymbol{h}}^k | k < t\}) \tag{4.11}$$

where each $P(\boldsymbol{v}^t, \boldsymbol{h}^t | \{\boldsymbol{v}^k, \hat{\boldsymbol{h}}^k | k < t\})$ is a joint probability (Equation 4.2) of the RBM at time t. $\hat{\boldsymbol{h}}^k$ is the binary mean-field value of $\boldsymbol{h}^k$ while sampling and inference, which can be transmitted to its following RBM. This property make it have a low calculation cost to inference $\boldsymbol{h'}$ Each RBM depends on previous steps, it can be considered only the biases depend on $h'^{(t-1)}$:

$$\boldsymbol{a}^t = \boldsymbol{a} + W' h'^{t-1} \tag{4.12}$$

$$\boldsymbol{b}^t = \boldsymbol{b} + W'' h'^{t-1} \tag{4.13}$$

The following definition of $\boldsymbol{h'}^t$ can be obtained by combining equations 4.7 and 4.12.

$$\boldsymbol{h'}^{(t)} = \sigma(W\boldsymbol{v}^{(t)} + a^{(t)}) = \sigma(W\boldsymbol{v}^{(t)} + W' * \boldsymbol{h'}^{(t-1)} + \boldsymbol{a}) \tag{4.14}$$

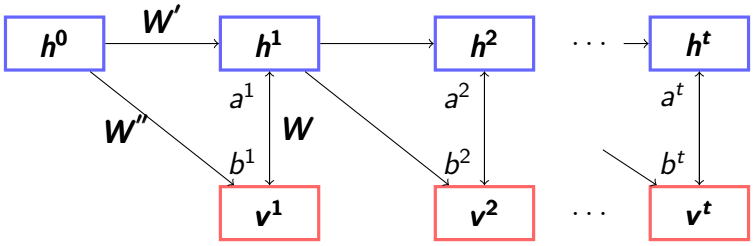

Figure 4.8: The graphical structures of the RTRBM

4.3.2 The RNN-RBM

As mentioned before, there is a restriction imposed on the RTRBM, which is that a hidden layer must describe the conditional distributions and affect the following hidden layer. However, not all the events happened would have a long-term effects. In our case, some financial reports just includes a very short summary of old information, which may not have some long-term effect on the stock movement. If we use the RTRBM to predict the stocks, every article would affect the following days. In order to lift this constraint, another model call RNN-RBM were proposed in [14] with combining a full RNN as the structure shown in Figure 4.9. The RNN-RBM is an abbreviation of Recurrent Neural Networks-Restricted Boltzmann Machine. The joint probability is still as equation 4.11, but with $\boldsymbol{h'}$ defined arbitrarily. The

parameters of each RBM are considered to be W, $\boldsymbol{at}$ and $\boldsymbol{b}^t$, which depend on the output of a single layer RNN with hidden units $\boldsymbol{h'^{t-1}}$:

$$\boldsymbol{a}^{(t)} = \boldsymbol{a} + W' h'^{t-1} \tag{4.15}$$

$$\boldsymbol{b}^{(t)} = \boldsymbol{b} + W'' h'^{t-1} \tag{4.16}$$

and the recurrent hidden units of the single-layer RNN can be obtained:

$$\boldsymbol{h'}^t = \sigma(W_2 \boldsymbol{v}^t + W_3 * \boldsymbol{h'}^{t-1} + \boldsymbol{c}) \tag{4.17}$$

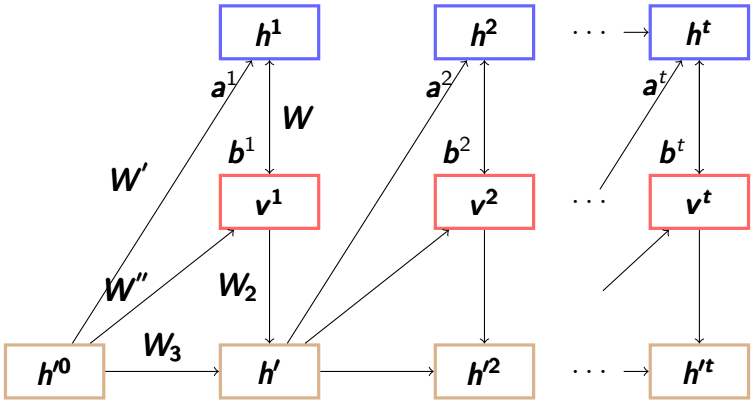

Figure 4.9: The graphical structures of the RNN-RBM

Nine parameters are given to the RNN-RBM. They are four parameters of RBM: W, $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{h'}^0$, and five parameters of RNN: W', W'' W_2, W_3, $\boldsymbol{c}$, which are used for training of this model. The general training process can be separated into several steps which are shown in Algorithm 7.

Algorithm 7 RNN-RBM Training

1: **procedure** RNN-RBMTraining
2: Use Equation 4.9 to propogate $\boldsymbol{h'}^t$ at current time t.
3: Calculate parameters of the RBM depending on $\boldsymbol{h'}^t$ by equation 4.15 and 4.16 , and then generate $\boldsymbol{v}^t$ with Gibbs sampling.
4: Use the contrastive divergence (CD) to obtain the derivative with respect to $\boldsymbol{a}^t$, $\boldsymbol{b}^t$, and W.
5: Derive at the parameters of RNN by applying Back-propagation Through Time (BPTT) algorithm [67].
6: Iterate previous steps
7: **end procedure**

Algorithm 8 describes the implementation of the RNN layer.

Algorithm 8 Build RNN-RBM

procedure BUILDRNN-RBM($\boldsymbol{x}$,v,h,r)
    $\boldsymbol{x} = [x_1, x_2, ...x_T] \in R^{n*m}$ is the input matrix, in which $x_i \in [0,1]^m$ ($1 \leq i \leq m$) is a single input data
    v is Integer, which represents the number of visible units.
    h is Integer, which represents the number of hidden units of the conditional RBMs.
    r is Integer, which represents the number of hidden units of the RNN.
    initialize W, $\boldsymbol{a}$, $\boldsymbol{b}$,W', W'', $\boldsymbol{h'^0}$ W_2, W_3, c
    $\theta = [W, \boldsymbol{a}, \boldsymbol{b}, W', W'', W_2, W_3, \boldsymbol{c}]$
    iterate call RECURRENCE($\boldsymbol{v^t}$,$\boldsymbol{h'^{t-1}}$) given $\boldsymbol{h'^0}$ $\boldsymbol{v^0} = \boldsymbol{x}$ for T times, to get all $\boldsymbol{a}^t, \boldsymbol{b}^t$ ($0 < t < T$)
    iterate call RECURRENCE($\boldsymbol{v^t}$,$\boldsymbol{h'^{t-1}}$) given $\boldsymbol{h'^0}$ $\boldsymbol{v^0}$ = *None* for T times, to do hidden sequence generation

end procedure
procedure RECURRENCE($\boldsymbol{v^t}$,$\boldsymbol{h'^{t-1}}$)
    $\boldsymbol{a}^t = \boldsymbol{a} + W' h'^{t-1}$
    $\boldsymbol{b}^t = \boldsymbol{b} + W'' h'^{t-1}$
    generate = $\boldsymbol{v^t}$ == None // that means if the input parameter $\boldsymbol{v^t}$ is None, *generate* will be true.
    if generate **then**
        TRAINRBM(ZEROS(v), $W, \boldsymbol{a}^t, \boldsymbol{b}^t, k = 25$)
    end if
    $\boldsymbol{h'}^t = \sigma(W_2 \boldsymbol{v}^t + W_3 * \boldsymbol{h'}^{t-1} + \boldsymbol{c})$
    if generate **then**
        return $[\boldsymbol{v^t}, \boldsymbol{h'}^t]$
    else
        return $[\boldsymbol{h'}^t, \boldsymbol{a}^t, \boldsymbol{b}^t]$
    end if
end procedure

4.3.3 Combination with DBN

RNN-RBM is combined with a DBN, in order to obtain a multilayer structure with expressing complex function [83]. We expect that it is a deep structure model which can depict time series temporal effect. Both of their unit components are RBMs, so it is easy to combine them. The layer setup are briefly presented in Algorithm 9. RNN-RBN was used to substitute the first RBM which is visible layer $\boldsymbol{v}$ and hidden layer $\boldsymbol{h_1}$ of DBN in Figure 4.7. Its training algorithm is similar to DBN, which was separated to pre-training and fine-tuning. In the pre-training step, the parameters are computed by pre-training algorithm of RBM and RNN-RBM which have been introduced above. In the fine-tuning step, the parameters are tuned by back-propagation.

Algorithm 9 RNNRBM-DBN Layer Setup

procedure LAYERSETUP(*hiddenLayerSize*)
    hiddenLayerSize $= [l_1, l_2, ..., l_n]$ where n is the number of hidden layers, $l_i, i \in [0, 1]$ is the number of hidden units in the $i - th$ hidden layer
    $\boldsymbol{x} = [x_1, x_2, ...x_T] \in R^{n*m}$ is the input matrix, in which $x_i \in [0, 1]^m$ $(1 \leq i \leq m)$ is a single input data
    for i in $[0, n]$ **do**
        if $i == 0$ **then**
            inputSize $= T$
            layerInput $= \boldsymbol{x}$
        else
            inputSize $=$ *hiddenLayerSize*$[i - 1]$
            layerInput = the output of the last element of *sigmoidLayers*
        end if
        if $i == 0$ **then**
            rnnrbmLayer = create an RNNRM object given input parameters *inputSize* as visible unit size, *layerInput* as as input matrix, *hiddenLayerSize*$[i]$ as the hidden layer size.
            hiddenLayer = Create a hidden layer with input *layerInput* as input matrix, *rnnrbmLayer*.$\boldsymbol{W}$ as parameter $\boldsymbol{W}$, *rnnrbmLayer*.$\boldsymbol{b^{(t)}}$ as parameter $\boldsymbol{b}$, *sigmoid* as activation function
        else
            hiddenLayer = Create a hidden layer with input *layerInput* as input matrix and default randomized $\boldsymbol{W}$ and $\boldsymbol{b}$, *sigmoid* as activation function
            rbmLayer = create an RBM object given input parameters *inputSize* as visible unit size, *layerInput* as as input matrix, *hiddenLayerSize*$[i]$ as the hidden layer size. *hiddenLayer*.$\boldsymbol{W}$ as parameter $\boldsymbol{W}$, *hiddenLayer*.$\boldsymbol{b}$ as parameter $\boldsymbol{b}$,
        end if
        Append *hiddenLayer* to *sigmoidLayers*
        Append *hiddenLayer*.$\boldsymbol{\theta}$ to params
    end for
    logLayer = create a logistic regression with the output of the last layer of *sigmoidLayers* as input matrix, the last element of *hiddenLayerSize* as hidden layer size
    Append *logLayer*.$\boldsymbol{\theta}$ to params
end procedure

Chapter 5

Implementation

In the section, we will present key implementation details of this project. Note that the there are too many implementation details to fully introduce them. Hence we only talk about some of them as examples. First, an overview of implementation is provided.

5.1 Implementation Overview

The implementation consists of several processes which are data processing, model building, model operation, model evaluation and model optimization. The pipeline flow chart is shown in Figure 5.1. The textual financial reports first are processed with some natural language processing and information retrieval techniques which have been introduced in chapter 2. For the price data, it need to be done with a price to label mapping with a mechanism introduced in section 5.3. Some financial indicators were computed to obtain a more complete analysis. For each stock, five machine learning models are used to train the models for comparisons. The five models are SDA, DBN, modified RNN-RBN. SVM and RF. The first three are deep learning models, and the last two state-of-art and classic machine learning models. By optimizing the parameters, an relatively good model can be built. Then test data can be used to test the efficiency of the models. In the model operation step, some prediction scores can be obtained. The scores can map to the corresponding labels. At the phase of model evaluation, we can compare the models we build, and do some analysis. By the the understanding of the models, we can return to the process of data processing, to change the methodology we used or optimized some parameters such as how many bag-of-words need to be selected and some other coefficients.

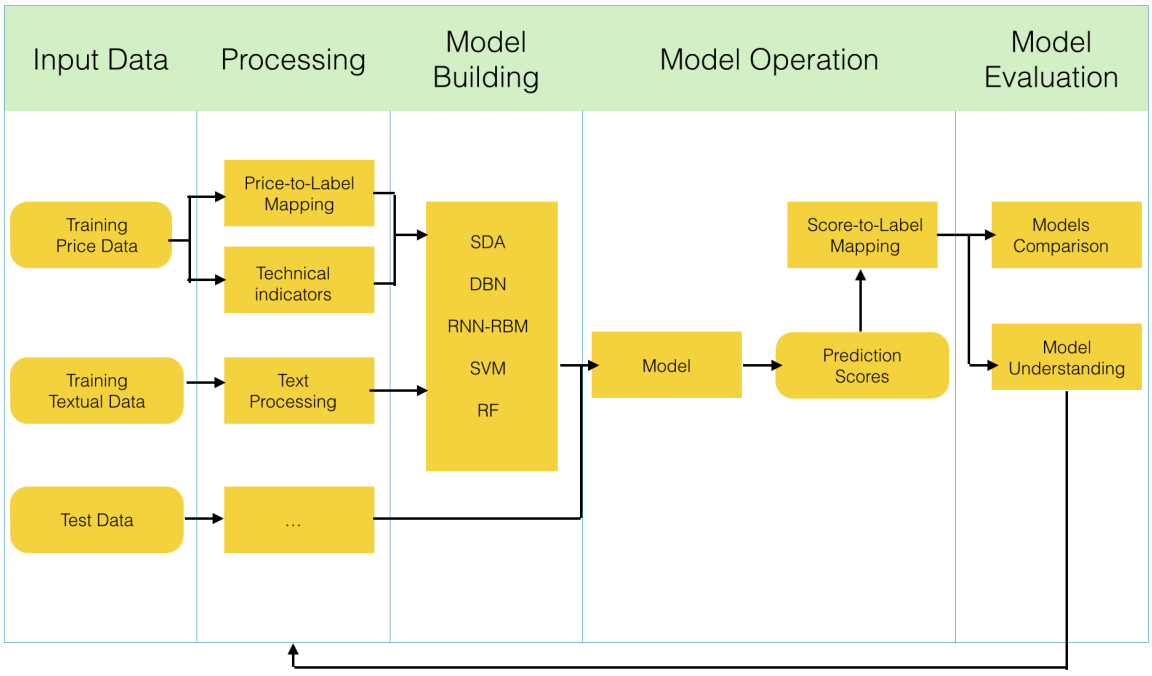

Figure 5.1: Pipeline of this project

5.2 Programming Language and External Library Used

The chosen main programming language is *python*, because it has some libraries suitable for machine learning programming. In addition, it is a script language. This feature makes it suitable to do scientific programming and experiments. Some libraries we used are not included in the python. So they are external libraries which are need to be installed. The are listed below:

- *NLTK* is a natural language process library, whose Wordnet Word Lemmatizer was used as a library to get the lemma of each words.
- *sklearn.svm* and *sklearn.ensemble* provide the library functions for using Support Vector Machine and Random Forest respectively, which can be directly used as baseline methods to compare with deep learning models.
- *theano* [10] is a python library which can allow the user efficiently define mathematical expressions involving multi-dimensional arrays on GPU, and prototype new machine learning models without unnecessary implementation overhead.
- *matplotlib* library is a plot library, which was used to draw financial indicators.
- *NumPy* is a powerful library for the fundamental scientific programming

- *progressbar* is a visualization library which can show a text progress bar to help the user to know the progress of a long running program. In this project, lots of individual program running in preprocessing steps are very long, so we used this library to monitor the progress of running.

5.3 Bag-of-word Selection

In section 3.2 and 3.3, the text feature selection and Chi-square text has been introduced. In this, we will talk about some implementation settings of feature selection process. In order to save computation time, for now, I fixed the number of terms to 1000, whose Chi-square score are the top 1000 highest. The stock change for each report are calculated by the difference between before and after the report is released. For example, if the report is released before 15:00, the price change will the difference between the day's opening price and closing price. Otherwise, the price change will be the difference between next day's opening price and the day's closing price. Each unique unigram was given three attributes, which are 'up', 'down' and 'stay'. 'up', 'down' and 'stay' means the difference between closing price and opening price on that day are greater than 1%, less than -1%, or between 1% and -1% respectively. Then I gave a label to each document by computing the stock price change on the corresponding date. Table 5.1 shows the information needed to calculate the Chi-square score, where u_a, d_a or s_a denote the number of days when the unigram occur in the 8-K reports, and the report were classified to Up, Down or Stay. Similary u_n, d_n or s_n denote the same thing but the unigram does not occur in the report. For each brand and each unigram, u_a, d_a and s_a are needed to be computed and then which are stored in the attributes 'up', 'down' and 'stay'. Other items in the table can be obtained by computing with u_a, d_a and s_a.

Table 5.1: Information needed to calculate Chi-square

	Up	Down	Stay	Sum
Appear	u_a	d_a	s_a	A
Not appear	u_n	d_n	s_n	N
Sum	U	D	S	T

Then for each brand and each term χ^2_{up} and χ^2_{down} were calculated as in Eq 5.1 and Eq 5.2. Then the greater one of χ^2_{up} and χ^2_{down} can be the utility score as presented in the Algorithm 1.

$$\chi^2_{up} = \frac{(|u_a \cdot (d_n + s_n) - u_d \cdot (d_a + s_a)| - \frac{A}{2})^2 \cdot A}{U \cdot D \cdot A \cdot N} \tag{5.1}$$

$$\chi^2_{down} = \frac{(|d_a \cdot (u_n + s_n) - d_d \cdot (u_a + s_a)| - \frac{A}{2})^2 \cdot A}{U \cdot D \cdot A \cdot N} \tag{5.2}$$

Now we have obtained the feature list with the format of bagofwords. Then for each document we need to check whether the unigram in the text feature list exists, if it exists, the value of the corresponding term will become 1, while the default

values for every text feature are zero. Then the text feature will be appended into the data structure shown in Figure 5.2. Note that every day may have not only one article, so the values corresponding to each day is a text feature matrix.

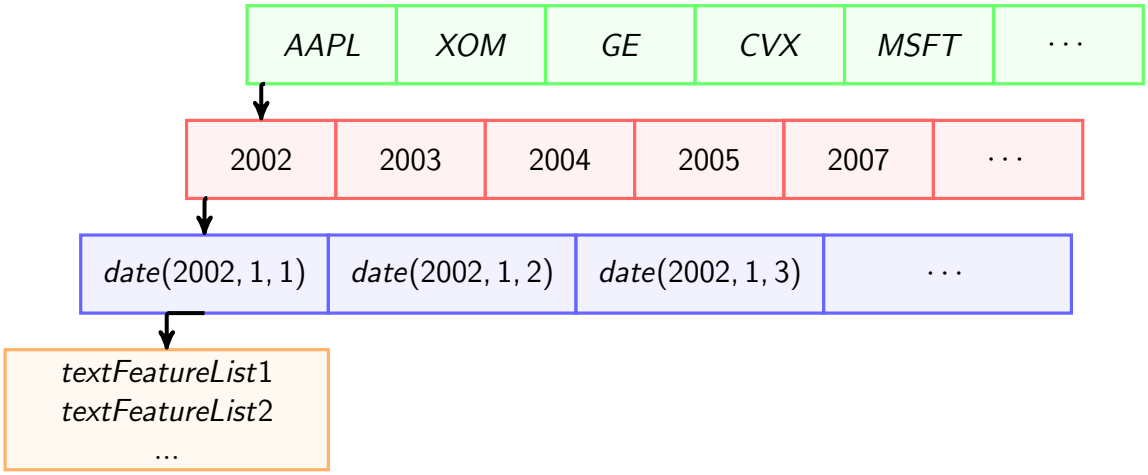

Figure 5.2: Data structure of text features

5.4 Same Day Textual Data Compression

The reasons and theory to compress the same day textual data have been introduced in section 3.4. This section, we will present several implementation details. For sparse vectors, we should train an unsupervised feature learning model to learn a more useful feature representation. In this project, both Sparse Autoencoders and RBM were tested and experimented. The model has been trained by unsupervised feature learning with training data. After that, parameters of the model were saved to a file. In the next step, we load the saved model and transform the input data to the hidden representation by the parameters of the model, e.g. $\mathbf{h} = sigmoid(\mathbf{Wx}+\mathbf{b})$. Note that for each day when the 8-k report published, there may be not only one report. In other words, there were possibly some reports published. However for each day, the label is only which is moving up or down of the stock price. Hence we have to unify the reports together from the same day. For each day, we may have a matrix which contains a set of hidden vectors $\mathbf{h_{(1)}}$ to $\mathbf{h_{(n)}}$. Then we are able to choose to use either max-pool or average-pooling to process the matrix as a new vector which can be used as an input of the next phase.

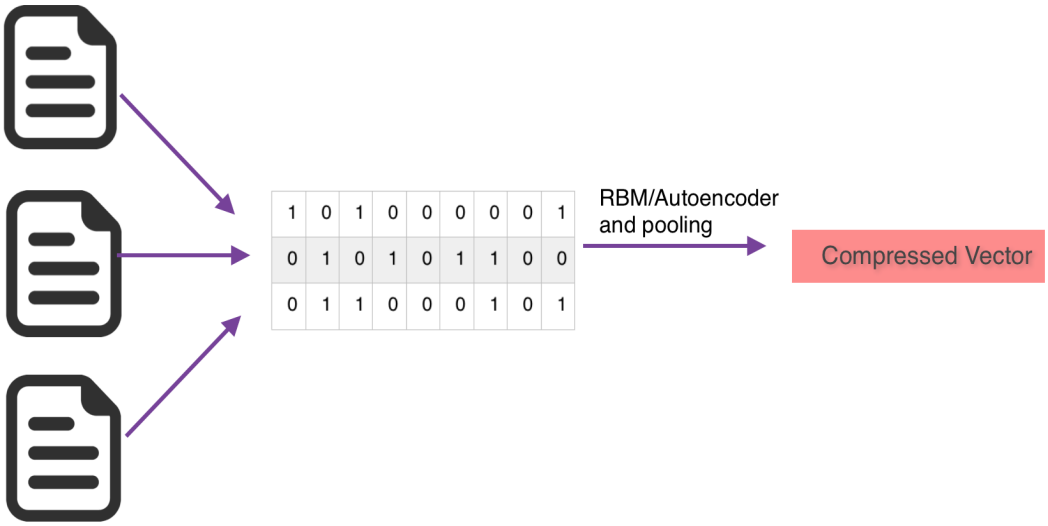

Figure 5.3: The process of same day textual data compression

5.5 Historical Price Data Processing

With the implemented functions presented in section 3.6, a program was implemented to draw a diagram including above financial indicators with the python library matplotlib. An example is shown in 5.4.

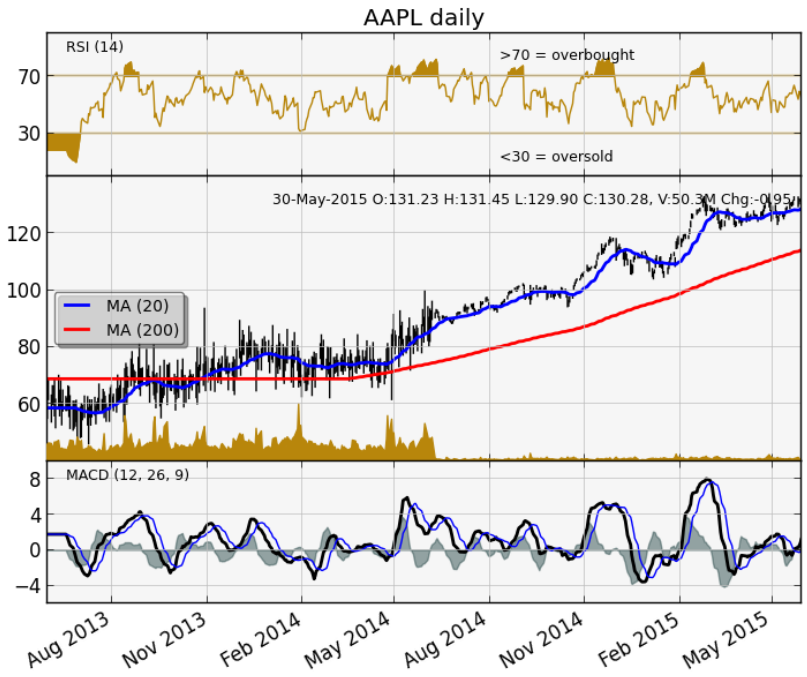

Figure 5.4: The figure contains the stock price information of Apple company from 30 May 2014 to 30 May 2015 including SMA, EMA, MACS and RS, drawn with matplotlib library

MACD were calculated for each day. The price information is stored in the data structure like Figure 5.5. Each brand is stored in a dictionary which the corresponding brand codes of the companies are as the keys. The value corresponds to each key is a dictionary whose keys are the years from 2002 to 2012. For each year, the corresponding value is another dictionary whose keys are the date objects of the year from 1^{st}, Jan to 31^{th} Dec. At last, each date object corresponds to another dictionary which contains the price information about opening price, closing price, trading volume, high price, low price, adjusted price, MACD and the next day's MACD. For example, the *AIG* stock in the figure contains a dictionary whose keys are 2002, 2003, 2004 and so on. The value corresponding to 2002 is another dictionary whose keys are $date(2002, 1, 1)$, $date(2002, 1, 8)$, $date(2002, 1, 10)$. They are the dates AIG published an 8-K report. Each value corresponding to the dates, for example $date(2002, 1, 1)$, is another dictionary whose keys are *openprice*, *high*, *low*, and other attributes shown in the figure. The values are the float numbers corresponding to the keys. The whole data structure is stored as an object in a file by python *cPickle* library.

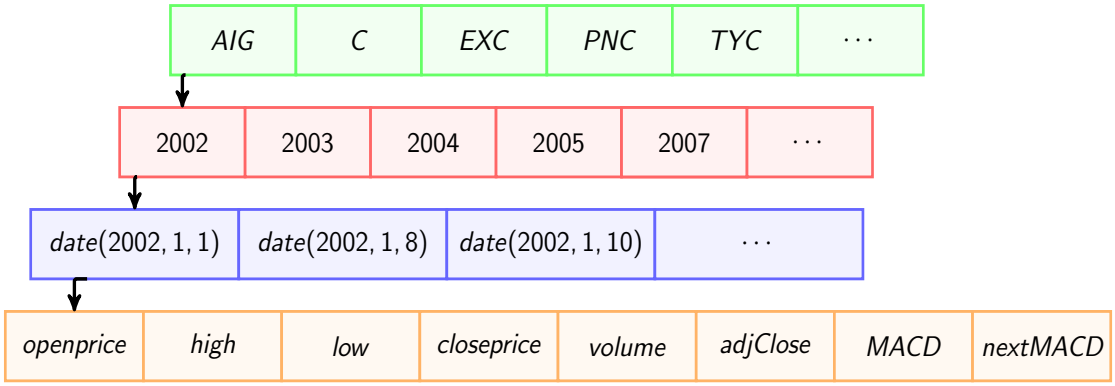

Figure 5.5: Data structure of stock price information

5.6 Model Building

The three deep learning models are implemented based on the algorithms which has been listed in Chapter 4 with python. For the model of SVM and RF, we take advantage of the library *sklearn.svm* and *sklearn2.ensemble*. The following section introduces the process of training steps.

5.6.1 Pre-training process

Each deep learning model needs two steps, one is pre-training and another is fine-tuning. The pre-training step is unsupervised layer-wise training, which aims to minimize the cost of each layer which was introduced in the previous chapter. Figure 5.6 shows an illustration of the pre-training process for different learning rate α as an example. This line chart shows the process of the pre-training step on the Citigroup Inc. (C). As the epochs increases, the cross entropy losses were going

down. Different learning rates lead to a different results. For this example, the lowest result $\alpha = 0.08$ lead to a lower cost than $\alpha = 0.001$ after 200 epochs.

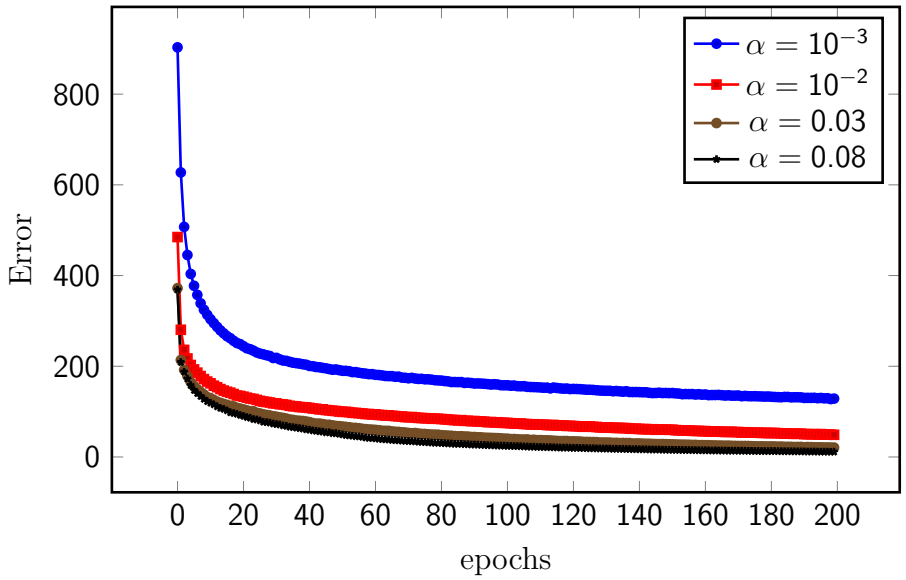

Figure 5.6: Pre-training process on the data of company Citigroup Inc. (C)

5.6.2 Fine-tuning process

Fine-tuning is another important step to learn and tune the parameters by minimizing the cost by back-propagation. Fine-tuning is a process of supervised learning, which was introduced in the previous chapter. Figure 5.7 shows an example of fine-tuning process still on the data of Citigroup Inc. (C). The fine-tune learning rate was set to 0.01, and batch size was set to 20. The brown line represents the training error trends as the epochs increased. It can be seen that the trend of the training error decreased as the epochs increased. The blue line represents the classification error of the validation set. Its trend went down first and then went up, because it would be overfitting if the training epochs are too large. The validation set was used to prevent from overfitting in the training process. The red line is the trend of the test set, which can not provide any information in the training process, just to test the performance of the trained model. The vertical dot line is the epoch the model make the validation error smallest. The model would finally choose the parameters at that epoch.

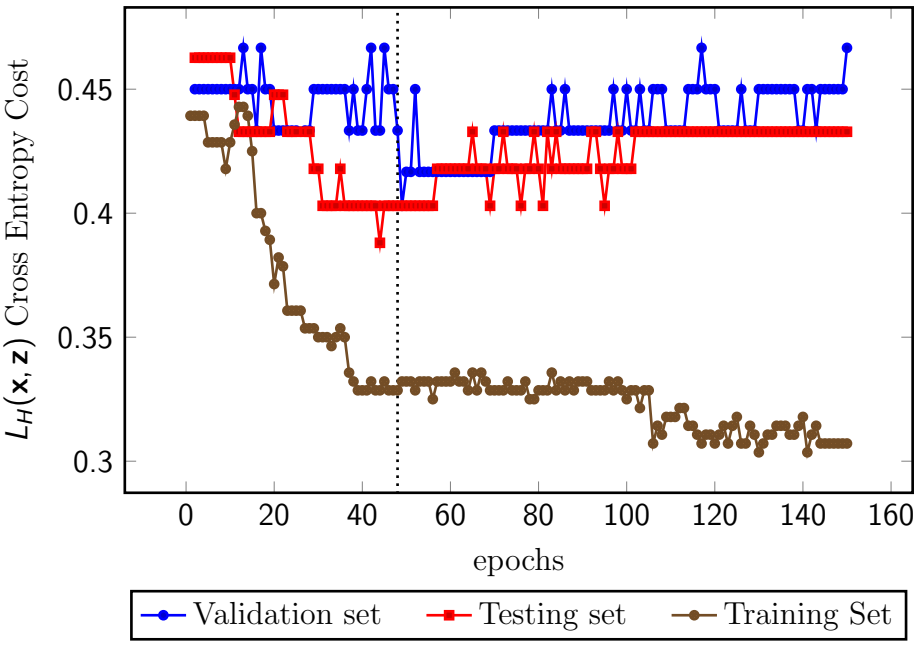

Figure 5.7: Fine-tuning process on the data of company Citigroup Inc. (C)

This chapter introduced some implementation of this project. In next chapter, we will discuss the experiments results and the analysis and evaluation of our models.

Chapter 6

Experiment Results and Analysis

This chapter describes the metrics used to evaluate the performance of each model in section 6.1. Individual results and analysis of each deep learning model are introduced in section 6.2. In section 6.3, three deep learning models and two other classic machine learning models SVM and RF are compared and evaluated together.

6.1 Evaluation Metrics

In order to evaluate the classification results on different learning algorithms, we need some measures and standards to judge the accuracy of the results. In this section, we will discuss several simple evaluation approaches to help us choose the best classification method.

6.1.1 The Confusion Matrix

The Confusion matrix [73] can be a useful validation tool to show the linear result of a classifier on a given input . Each row of the matrix represents the genuine class the input data belongs to and each column represents the output returned by the predictor. A benefit of the metric is that it can clearly show the confusing classification between two classes by a classifier. For instance, the Table 6.1 shows the example of the confusion matrix of a binary classification between class A and class B. Assume that Class A is the class we want to predict, which would be called positive, other predicted classes are called negative. There are 10 actual class A examples, and six of them are predicted correctly, which can be called true positives (TP). Four of them are predicted as class B, which can be called false negatives (FN). Three of seven actual class B samples are classified into class A called False Positive (FP) . Four of them correctly classified are called true negatives (TN). In addition the diagonal can show the accurate classifications.

Table 6.1: An example of Confusion Matrix

	Class A	Class B
Class A	6	4
Class B	3	4

6.1.2 Recall and Precision Rates

Consider a training problem on a binary classification as described above, there are only positive and negative examples with a binary classifier. True Positive (TP) / False Negative (FN) means the examples that are positive assigned to positive (true classification) / negative (false classification). Similarly, True Negative (TN) / False Positive (FP) are the positive examples assigned to negative (true classification) / positive (false classification). In this case, the most simple and obvious way to measure the performance is calculating the correction prediction rate which are the sum of TP and TN divided by the sum of FP and FN, whereas sometimes it may be a misleading measure when comparing which classifier is better. As the tables shown below in Table 6.2, classifier B is obviously better than classifier A. However the performance measure method has a problem for Table 6.3. Although the recognition of classifier B is much better than A, it may not be able to recognise positive data due to the mis-classification of all the actual positives, hence we cannot say that B is better than A. At that time some other measures can be introduced to solve that problem.

Table 6.2: Classification scenario 1

Classifier	TP	TN	FP	FN	Error Rate
A	50	50	50	50	50%
B	74	74	26	26	26%

Table 6.3: Classification scenario 2

Classifier	TP	TN	FP	FN	Error Rate
A	50	150	150	50	50%
B	0	300	0	100	25%

Recall and precision [62] are methods to measure the quality of the information retrieval process. Recall describes the completeness of the process by calculating the number of retrieved positive data divided by the number of the total positive data including the ones not retrieved. Precision describes that the accuracy of the retrieval via calculating the number of the correctly retrieved positive data divided by the number of all the correctly retrieved data including the correctly retrieved negative data. They can be used to measure the performance of the classifier. The formally mathematical expression of the calculation of precision and recall is shown

in Equation 6.1 and Equation 6.2.

$$precision = \frac{TP}{TP + FP} \tag{6.1}$$

$$recall = \frac{TP}{TP + FN} \tag{6.2}$$

6.1.3 F_α-measure

Although the recall and precision can be individually used, $F_\alpha$is a combination of these two values to measure the quality of the classifier based the which value the user are more interested in. The F_α formula is shown below in Equation 6.3.For our project, we use F_1 measures which means α is set to one shown in Equation 6.4.

$$F_\alpha = (1 + \alpha)\frac{recall * precision}{\alpha * precision + recall} \tag{6.3}$$

$$F_1 = 2\frac{recall * precision}{precision + recall} \tag{6.4}$$

6.1.4 Classification Performance

In our project, we need to predict the up-trend or down-trend in our models. Due to different aims, the retrieved positive data may refer to up-trend data, or down-trend data. If the positive data refer to the up-trend data, we defined a series of metrics called **UP**, **UR** and $\mathbf{upF_1}$ to represent the precision, recall and F_1 of the up-trend prediction. They can be mathematically expressed as:

- Up-trend Precision Rate:
  The percentage of accurately predicted uptrend data (out of all uptrend data predicted by the model):

  $$UP = \frac{\text{Number of correctly predicted uptrend data}}{\text{Number of predicted uptrend data}} \tag{6.5}$$

  This indicator reflects the confidence of the prediction of uptrend data.

- Up-trend Recall Rate:
  The percentage of up-trend data captured by the model (out of all uptrend labelled data)

  $$UR = \frac{\text{Number of correctly predicted uptrend data}}{\text{Number of data with true uptrend labels}} \tag{6.6}$$

  UR reflects the sensitivity of the model to the uptrend data.

- Up-trend F_1 Score:
  It considers both the uptrend precision and uptrend recall of the test to compute the score.

$$upF_1 = 2\frac{UP * UR}{UP + UR} \tag{6.7}$$

  upF_1 reflects the overall confidence and sensitivity of uptrend prediction.

If the user focus on the prediction of the down-trend data, a series of metrics called **DP**, **DR** and **downF$_1$** were defined as the prediction, recall and F_1 of the down-trend prediction. They can be computed with the following expression:

- Down-trend Precision Rate:
  The percentage of accurately predicted downtrend data (out of all downtrend data predicted by the model):

$$DP = \frac{\text{Number of correctly predicted downtrend data}}{\text{Number of predicted downtrend data}} \tag{6.8}$$

  This indicator reflects the confidence of the prediction of downtrend data.

- Down-trend Recall Rate:
  The percentage of up-downtrend data captured by the model (out of all down-trend labelled data)

$$DR = \frac{\text{Number of correctly predicted downtrend data}}{\text{Number of data with true downtrend labels}} \tag{6.9}$$

  DR reflects the sensitivity of the model to the downtrend data.

- Down-trend F_1 Score:
  It considers both the downtrend precision and downtrend recall of the test to compute the score.

$$downF_1 = 2\frac{DP * DR}{DP + DR} \tag{6.10}$$

  $downF_1$ reflects the overall confidence and sensitivity of downtrend prediction.

6.1.5 Prediction Accuracy

Another metric was used to measure the performance of the models is the percentage of correctly labeled vector with respect to the whole testing set, we call that prediction accuracy. It is the most commonly used performance evaluation indicator for other similar projects. It can be computed using the following equations:

$$\text{PA} = \frac{1}{n}\sum_{i=i}^{n} I(\hat{y}_i, y_i) * 100 \tag{6.11}$$

$$I(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases} \tag{6.12}$$

where n is the size of the whole testing data. $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_n]$ is the predicted label of input data. $\mathbf{y} = [y_1, y_2, ..., y_n]$ is the actual label set of the input data.

6.1.6 Stability of the Model

After defining the various evaluation metrics, the experiment results measured with the metrics can be attained to evaluate the model in different aspects. After that the stability of the model can be measured via calculating the standard deviation of each performance indicators. It is a widely used method to test the stability in statistics. The calculation formula can be expressed as Equation 6.14:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)} \tag{6.13}$$

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{6.14}$$

where $\mathbf{x} = [x_1, x_2, ..., x_n] \in R^n$ is a vector of performance of some stocks measured with a particular indicator and N is the number of stocks tested in the experiments.

6.2 Experiment Results and Analysis of Individual Deep Learning Model

This section describes some experiment results measured with the metrics defined in the previous section. The results will be analysed for the three deep learning model respectively.

6.2.1 Experiment Setup

The data set we can obtain contains data of 500 companies. Because we train each model on each company separately (not train them together as a whole model), so lots of models are built and need to find the appropriate parameters. It is unrealistic to train five model for each 500 companies. Moreover some of them do not have enough size of data to be trained. Due to the huge time consuming cost on the parameter optimization of the training process for each company, only 15 companies with largest size of data are selected as experiment data. They are Citigroup Inc. (C), American International Group, Inc. (AIG), Williams Companies, Inc. (WMB), Exelon Corporation (EXC), Tyco International Ltd. (TYC), Time Warner Inc. (TWX), The PNC Financial Services Group, Inc. (PNC), PG&E Corporation (PCG), CATCO REINSURANCE (CAT.L), Sprint Corporation (S) and Equity Residential (EQR). At first, some experiment results are shown for individual learning models. The training set is the 8-K reports from 2002 to 2008. The validation data set is the reports in 2009 and 2010. The test data set os the reports in 2011 and 2012. The number of bag-of-word features are fixed to 1000, because it is a relatively good value for a better results by experiments.

6.2.2 Experiments of Stacked Denoising Autoencoder

Each model's parameters are optimized by the experiments of pre-training and fine-tuning. After that performance indicator will be recorded. Figure 6.1 is a bar chart which shows the prediction performance of some sample companies in terms of various indicators. SAE compression is used with max-pooling to pre-process the input data. The chart includes the results in a set of specific parameters of five sample companies, which are Pacific Gas & Electric Co. Comm (PCG), Sprint Corporation (S), Tyco International Ltd. (TYC), Time Warner Inc. (TWX) and Altria Group Inc. (MO).

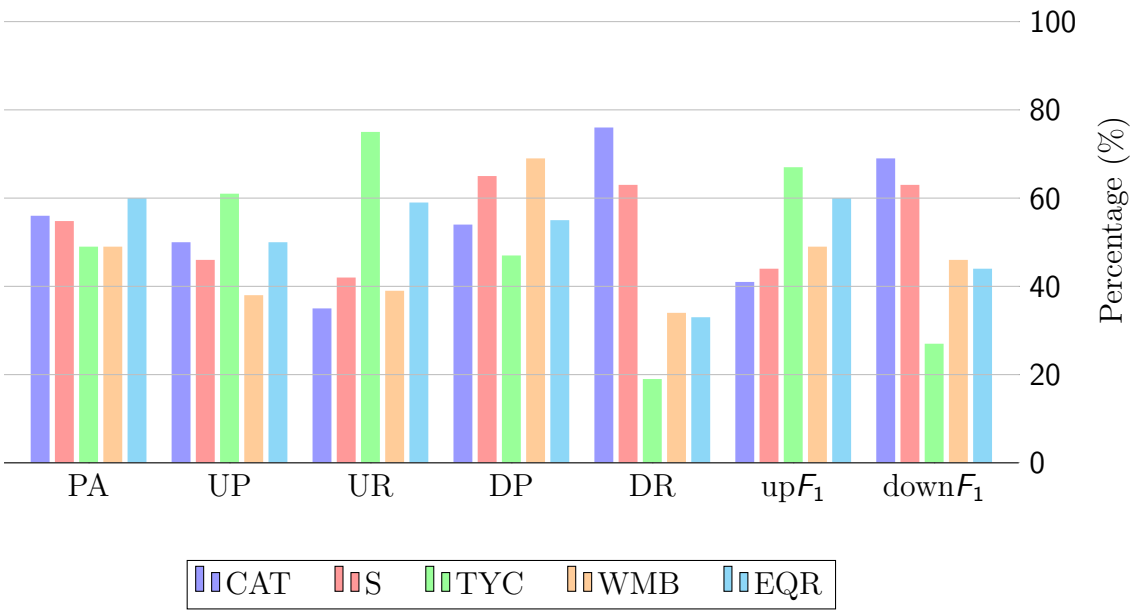

Figure 6.1: Overall prediction performance indicators of SDA on some example stocks

The first bench of the bars shows the prediction accuracies of the sample companies, which are around 57%. It can be seen that the prediction accuracy is relatively stable. Another interesting observation is that the recall rate of both up-trend and down-trend data is not so stable as the prediction rate, especially for the downtrend recall. For example, the company CAT and S has a very high downtrend recall, but the other three companies have much lower downtrend recall rate. One of the reason is that the not enough and uniform uptrend and down trend article data can make the model sensitive to different category classifications. It can also be seen that the model shows a average relatively high and stable performance on the downtrend precision, which means that model have a good confidence on the downtrend stock prediction.

The standard deviation of are given in Figure 6.2, which shows the stability of the performance with respect to the various performance indicators. It is easy to find that the value of prediction accuracy, uptrend precision and downtrend precision are relatively low which means that they are relatively stable. However the

standard deviation of uptrend recall and downtrend recall are relatively high, which means that they are not stable enough. The stability of upF_1 and downF_1 is in a middle level. According to the definition of the indicators in section 6.1.4 and 6.1.5, the average prediction accuracy are in a stable level indicates the model of different brands can have a similar performance. The performance of prediction confidences on uptrend and downtrend data both are stable. However, the relatively high standard deviation of recalls indicates the sensitivity of prediction are very dependent for different companies. The reason is possibly that different companies' reports have different selected words. Some of them are more sensitive to positives and some are sensitive to negatives.

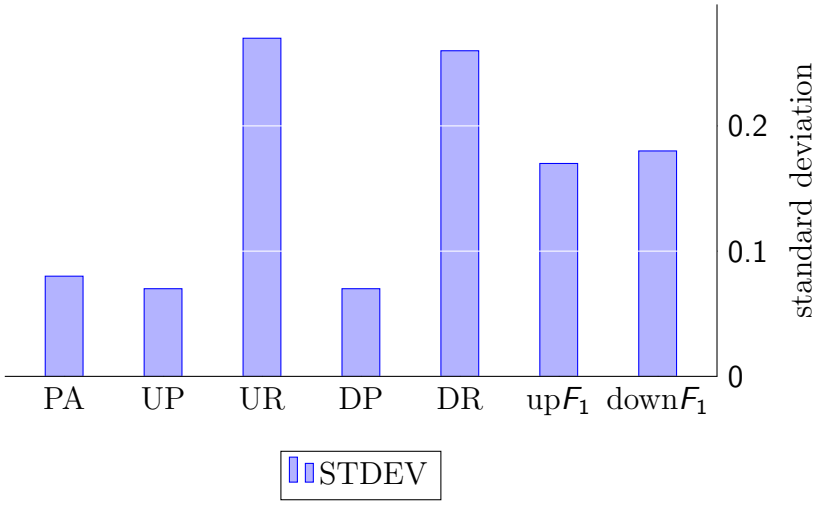

Figure 6.2: Standard Deviation Bar Chart of various performance indicator

6.2.3 Experiments of Deep Belief Networks

In Figure 6.3, the five chosen stocks are still used as examples to illustrate an overview and comparison of the prediction performances. An interesting observation can be that *down*F_1 are greater than or equal to upF_1 for the given five sample stocks, which indicates that the model has a better performance on downtrend prediction than uptrend prediction. It can also be seen that although the precisions (UP, DP) of them in the same level, their counterpart (UR, DR) have a big differences between the stocks. For example, stock CAT have a high DR, which mean it very sensitive to downtrend stocks, while the model of TYC are not very sensitive to downtrend stocks. The confidence of downtrend data is also high, because all the DP of sample stock are near 60%.

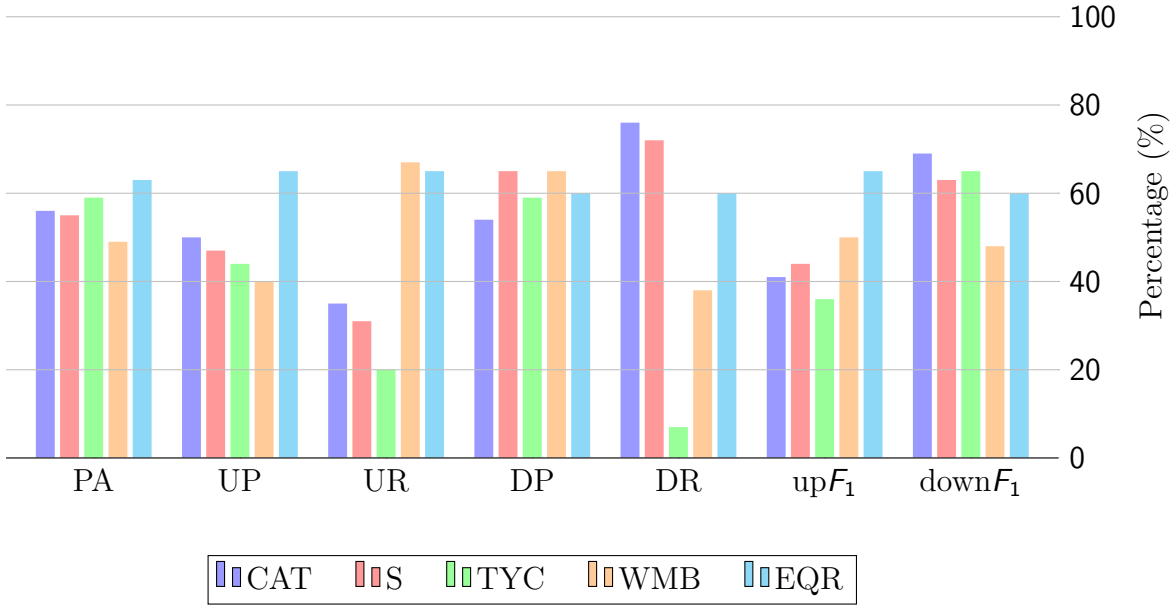

Figure 6.3: Overall prediction performance indicators of DBN on some example stocks

Figure 6.4 shows that the standard deviation of the experiments results of DBN on five example stocks with six pre-defined performance indicators. From the chart, it can be clearly seen that the performance results on prediction accuracy are most stable. Compared to the recalls including uptrend recall and downtrend recall, the precisions are more stable. According to the definition of precisions and recalls in section 6.1.4, the confidence of the prediction are stable, but the sensitivity are not stable enough. Besides the reason talked about in previous section, another dominating reason may be the financial report data for each company are not large enough. Some of them have much more uptrend data and some of them may have much more downtrend data. Hence for those companies, the model may more sensitive to one of uptrend or downtrend data. But a difference between the stability of SDA and DBN is that for DBN, the uptrend prediction are a bit more stable than the downtrend prediction, no matter in sensitivity or confidence. It indicates that the DBN can give a more stable performance for uptrend data for the given experiment data set. In terms of the stability of F_1 feature scores, they are in a middle level. The uptrend predictions are more stable than the downtrend ones.

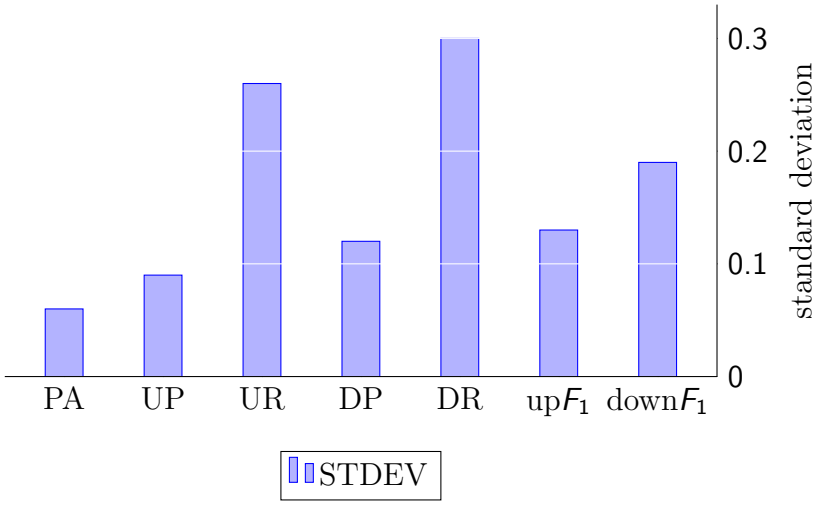

Figure 6.4: Bar Chart of Standard Deviation results for Deep Belief Networks on various performance indicators

6.2.4 Experiments of RNNRBM-DBN

Figure 6.5 shows the average performance indicators of the RNNRBM-DBN model on the the same example stocks as the previous two sections. As similar to the previous two models, precisions are more stable than recalls. Compared to previous ones, prediction accuracies are higher, but the recalls are a bit more unstable. The uptrend recall of WMB is higher than 80, but its downtrend recall is lower than 20. It also reveals that, for some stock, the model may be much more sensitive to either uptrend or downtrend prediction. This also indicates that the prediction with only financial reports is not the best solution, it is better to combine the textual data information with some other traditional financial features to improve prediction accuracy in the future work.

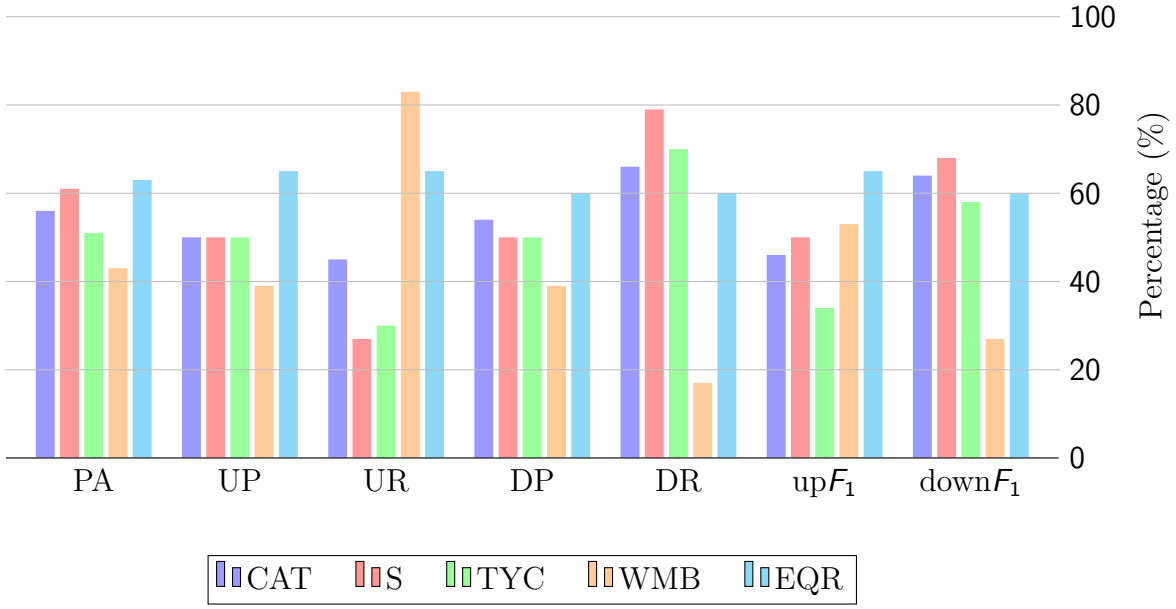

Figure 6.5: Overall prediction performance indicators of RNNRBM-DBN on some example stocks

Figure 6.6 shows the standard deviation of the experiments results of RNNRBM on five example stocks, with six pre-defined performance indicators. The shape of the diagram looks very similar to the DBN. So the stability of RNNRBM combined with DBN are similar to those of DBN. One difference is that the overall standard deviations of RNNRBM are a bit smaller than those of DBN. This may indicate slightly higher stability.

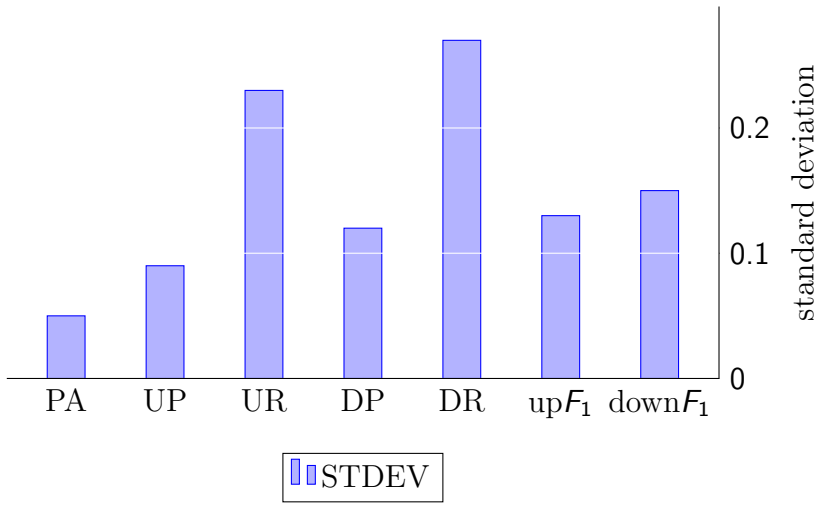

Figure 6.6: Bar Chart of Standard Deviation results for RNNRBM-DBN on various performance indicators

6.3 Empirical Results and Analysis of All Models

In this section, we will compare all the three deep learning models we built in this project, together with two other powerful machine learning models Support Vector Machine (SVM) and Random Forests (RF) as compared baseline methods. The reason we chose them is that they are two of most widely used and powerful machine learning models.

6.3.1 Evaluated with Price Prediction

Figure 6.7 shows that the average results across all the companies which are taken in the experiments with various performance indicators. From the figure, it can been seen that the RNN-RBM with adding DBN has a the most highest prediction accuracy for the selected companies. The prediction accuracy of DBN and RNNRBM-DBN are very similar, nearly in the same level and RNNRBM-DBN are just a bit higher than DBN. Compared to support vector machine and random forest, the average prediction accuracy was improved from 53.8% and 53.9% to 59.4%. All the three deep learning models have better results than traditional models. The prediction accuracy of SDA, DBN and RNNRBM are 58.3%, 59.0% and 59.4% respectively, and there is not too much difference between them. The detailed prediction accuracy results on each stock are shown in Table 6.4. The predictions including uptrend prediction and downtrend prediction of three deep learning model are in the same level, but there are some obvious differences between the recalls. Among them, SDA has the highest uptrend recall but lowest downtrend recall. Instead, RNNRBM has the highest downtrend recall and lowest uptrend recall. It indicates that SDA model are more sensitive to the uptrend prediction, and RNNRBM are more sensitive to the downtrend prediction for the given sample companies. RNNRBM having the lowest upF_1 and highest $downF_1$ means it performs best in down trend stock prediction but not good enough in uptrend data. SVM has the highest uptrend F_1 score but the lowest downtrend F_1 score among all five models, which means it perform not good on the downtrend stock prediction. For the random forest, all its performance indicators are in a stable level. Overall, all the three deep learning models obtained a relatively good results compared to the traditional machine learning models.

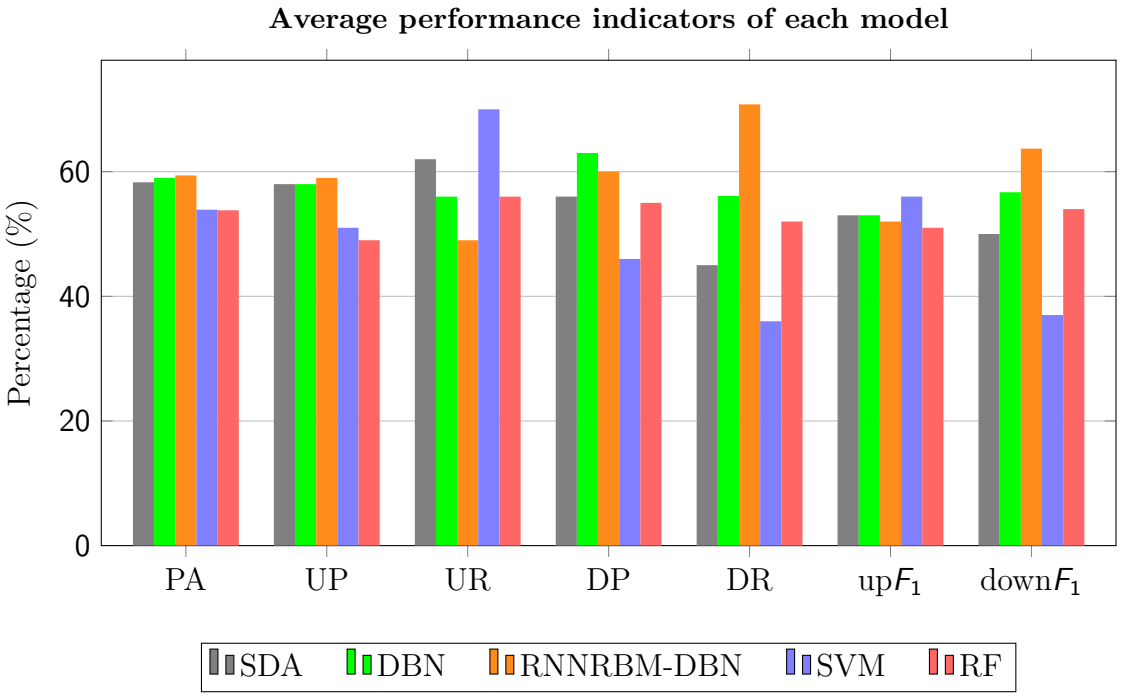

Figure 6.7: This figure shows the average performance indicators of each model. Each bar represents the average performance of the corresponding model across all stocks. For example, the first bar means the average prediction accuracy on all stocks of the models built with SDA.

Table 6.4: Test prediction accuracy rates for stock price prediction

Brand Code	SDA	DBN	RNNRBM +DBN	SVM	RF
C	58.2	59.7	58.2	53.7	53.7
AIG	62.7	64.4	58.3	45.7	57.6
TYC	48.7	58.5	51.2	51.2	51.2
WMB	57.4	59.5.3	66.0	57.4	70.2
TWX	64.7	64.7	64.7	64.7	58.8
EQR	64.7	62.5	62.5	53.1	52.9
PNC	61.7	61.7	61.7	61.7	53.1
PCG	54.6	57.8	64.7	65.6	51.5
CAT	55.7	55.7	55.7	44.2	50.0
S	54.8	54.8	61.2	54.8	51.6
T	57.8	52.3	54.6	52.3	47.7
EXC	54.4	55.9	54.4	52.9	39.7
JPM	58.5	60.6	58.4	51.5	50.9
MO	54.8	54.8	54.8	51.6	48.3
FCX	66.7	62.5	59.3	54.1	70.8
Average	58.3	59.0	59.4	53.9	53.8

Figure 6.8 shows the standard deviation of performance indicators of each model on the results of all companies. The bar charts compared the stability of different models on three main performance indicators. We know F_1 is calculated by precision and recall, which means it containes the overall information of precision and recall. To make emphasize the stability of the overall performance, in the graph we just look at the stability of prediction accuracy, upF_1 and downF_1. It is obvious to see that the RF has the most stable performance, which is the smallest one for all three indicators, although its performance is not better than the others. It can also be seen that all five models shows a more stable performance on uptrend predictions than downtrend predictions, especially for the SVM. The performance of three deep learning models are nearly in the same stable levels.

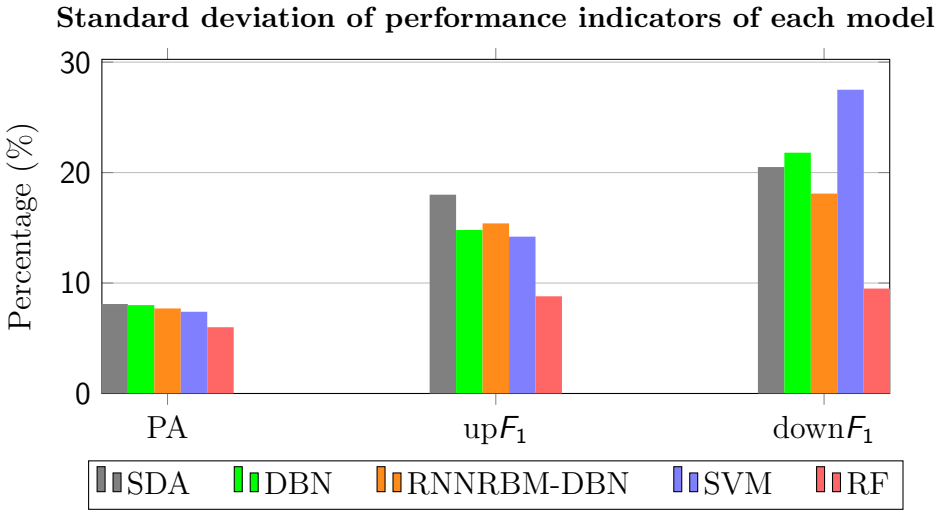

Figure 6.8: This figure shows the standard deviation of performance indicators of each model. Each bar represents the standard deviation of the specific performance indicator of the corresponding model across all stocks. For example, the first bar means the standard deviation of prediction accuracy on all stocks of the models built with SDA.

6.3.2 Evaluated with MACD Prediction

Figure 6.9 shows the MACD prediction results. MACD is a trend-following momentum indicator which was introduced in section 3.6. It also contains the information of SMA and EMA. From figure, it can been seen that the DBN performs best in MACD predictions for the tested companies, which is nearly 55%. It is a bit higher than the accuracy of RNNRBM-DBN. The performance of these two models look obviously higher than other models. All of the predication accuracy of the three deep learning models are greater than compared baseline model SVM and RF. The prediction accuracy of SDA, SVM and RF are nearly in the same level, with not too much difference to each other. Among them, SDA is a bit higher and RF is a bit lower.

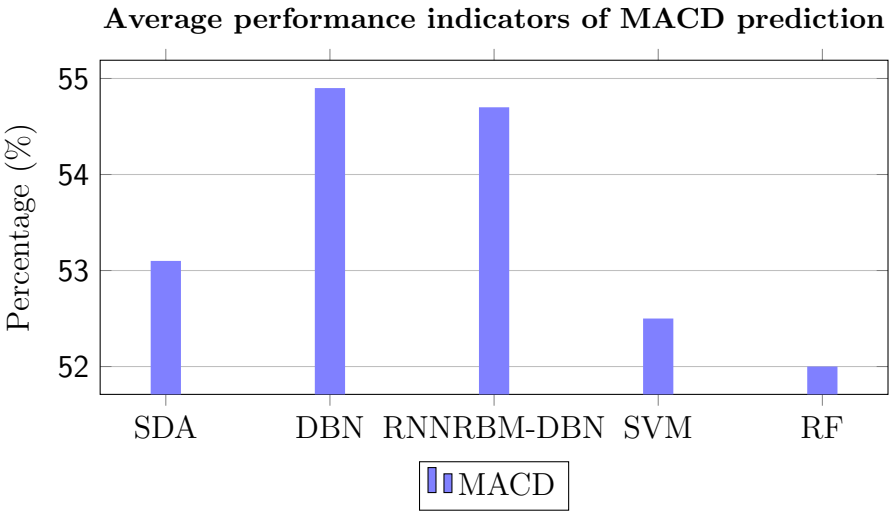

Figure 6.9: This figure shows average prediction accuracy of each model on MACD prediction. Each bar represents the prediction accuracy of the corresponding model across all stocks. For example, the first bar means the average prediction accuracy on all stocks of the models built with SDA.

Chapter 7

Conclusions and Future Works

In this last chapter, the theoretical study and analysis of this thesis are briefly concluded. The main contributions of this project are discussed. After that, the limitations of this work are presented and future work suggestions are given.

7.1 Conclusions

In this thesis, the importance of financial textual data for stock prediction was investigated with different types of machine learning models. Three deep learning models were adapted to the stock prediction problem, their results were compared with two classic machine learning models. After reviewing many natural language processing papers and comparing some information retrieval methods, the pre-processing steps were designed and implemented. The theoretical backgrounds of stacked denoising autoencoder, deep belief network and the recurrent neural networks-restricted boltzmann machine were reviewed and re-summarized. After that, technique details of the prediction models were designed. Overall, the following tasks were successfully designed and implemented:

- A series of text pre-processing steps with natural language processing techniques
- Chi-square key words selection
- Two pooling strategies to unify multiple articles of the same day.
- Tf-idf document normalization
- A set of financial indicators of prices for each textual article
- The stacked autoencoder based prediction model
- The deep belief network based prediction mode
- The RNN-RBM combined with DBN based prediction model

- A pipeline to use financial reports to predict the trend of stock prices
- A series of performance evaluation indicators
- Run experiments to determine the values of optimized parameters for each model.

During the experiments of evaluation, the designed performance indicators were used to evaluate the performance of the models in the following aspects:

- The prediction accuracy of each model for each company
- The sensitivity and confidence of up trend and down trend price prediction of each model for each company
- The stability of price prediction of each model for each company
- The comparison of three deep learning models and two other classic machine learning models (SVM and RF) with the previous defined performance indicators.

Based on the implementation process and the experiment result of evaluations, the following conclusion can be drawn:

- All the three deep learning models show a relative stable performance on prediction accuracy, uptrend precision and downtrend precision, which indicates that the confidence of the price prediction of both uptrend and downtrend data are stable. (section 6.2.2,6.2.3, and 6.2.4)
- All the three deep learning models show a relative not stable performance on uptredn recall and downtrend recall, which indicates the sensitivity of the prediction are not relative not stable enough. (section 6.2.2,6.2.3 and 6.2.4)
- The DBD model shows that the prediction accuracy is most stable among the performance indicators of DBN (section 6.2.3).
- The DBN shows that the performance on uptrend prediction is slightly more stable than that on downtrend prediction, no matter in sensitivity or confidence. It indicates that the DBN can give a more stable performance for uptrend data for the given experiment data set (section 6.2.3).
- The uptrend predictions of DBN model are overall more stable than the downtrend ones (section 6.2.3).
- The RNNRBM model have the same stability shape of the six performance indicators as the DBN model, but the RNNRBM model have a overall higher stability (section 6.2.4).
- RNNRBM show the highest prediction accuracy among the five models. The prediction accuracy of the DBN moded is nearly the same as the RNNRBM model (section 6.3.1).

- RNNRBM is most sensitive to downtrend prediction but least sensitive to uptrend prediction (section 6.3.1).
- All the three deep learning models obtained a relatively good results compared to SVM and RF on stock price trend prediction (section 6.3.1).
- The random forests model shows the most stable performance, and the overall stabilities of the three deep learning models are on the same level (section 6.3.1).
- All five models shows a more stable performance on uptrend predictions than downtrend predictions on price movement (section 6.3.1).
- For the MACD prediction, the DBN model shows the higher performance on the prediction accuracy (section 6.3.2).
- All the three deep learning models still show a relatively good results compared to SVM and RF on MACD movement prediction (section 6.3.1).

7.2 Final Remarks

Using a large number of experiments and a wide range of performance indicators, we showed that *incorporating textual information is indeed important for a more precised prediction of stock price movement*. The comparison between the performance results of the five models showed that *deep learning models have a potential applicability in the context of stock trend prediction, especially for DBN and RNN-RBM*. However there are still some limitations of this work, which are discussed below:

- Note that, to avoid the mixing the key factors that affects the movement of stock prices for different companies, we build models for each companies. As introduced in the previous chapters, we used 8-K financial reports as the input training data. However, the financial reports are not produced very frequently, which means the size of training input data for each companies is not large enough. Other types of financial articles listed in Table 2.2 can thus be used.
- Although we showed that it is very useful to incorporate financial textual analysis into the prediction of stock price movement, we do not claim that the techniques and designed pipeline in this dissertation can satisfy the basis of a viable trading strategy. Because some real-world obstacles to profitable trading has not been considered into the model. At least, a convincing trading model would overcome the restriction towards the following forms [23]:
  - *Transaction cost*, such as bid-ask spreads
  - *Slippage*, when large trading programs are executed when there may not be enough profit as expected level

– *Borrowing costs*, the charge for taking on a debt obligation when a short holding the securities predicted to go down.

Due to time constraints, the modelling of these events was beyond the topic of this dissertation.

7.3 Future Work Directions

There are several directions to extend this thesis for future improvements:

- In this thesis, we only considered the impact of textual articles to the stock price movement. To increase prediction accuracy, other traditional stock data sources can be combined. For example, recent price movements, earnings surprise, and so on.
- The optimization step for this project is based on experiments, grid search and re-factory. A more advanced optimization approach can be researched in the future.
- For each stock, five models were built. For each model, the parameters need to be optimized to a sufficiently good level. However the optimization stage to find a good set of parameters are really time consuming. Therefore, only 10 companies with largest report data were included in the experiments. The future work can try more companies to extend the generality and validity of the results.
- Due to time constraints, there are many types of technical financial indicators. In this project, we only predict the price movement and MACD indicator movement. The MACD indicators also contain the information of SMA and EMA. In the future, other technical financial indicators can be evaluated.

Bibliography

[1] Technical indicators and overlays
. *Available at: http://stockcharts.com/school/doku.php?id=chart*$_{s}$*chool* : *technical*$_{i}$*ndicators*.

[2] JG Agrawal, VS Chourasia, and AK Mittra. State-of-the-art in stock prediction techniques. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2:1360–1366, 2013.

[3] Gerald Appel. *Technical analysis: power tools for active investors*. FT Press, 2005.

[4] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

[5] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[6] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2015.

[7] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[8] Yoshua Bengio, Yann LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5), 2007.

[9] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[10] James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, et al. Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, 2011.

[11] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

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

[13] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135, 2012.

[14] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[15] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118, 2010.

[16] Alvin Chyan, Tim Hsieh, and Chris Lengerich. A stock-purchasing agent from sentiment analysis of twitter. Stanford. edu. Retrieved on September, 2012.

[17] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[18] Robert W Colby and Thomas A Meyers. *The encyclopedia of technical market indicators*. Irwin New York, 1988.

[19] Jack G Conrad and Joanne RS Claussen. Early user—system interaction for database selection in massive domain-specific online environments. *ACM Transactions on Information Systems (TOIS)*, 21(1):94–131, 2003.

[20] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

[21] Min Deng. Pattern and forecast of movement in stock price. *Available at SSRN 217048*, 2006.

[22] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on artificial intelligence and statistics*, pages 153–160, 2009.

[23] Andrea Frazzini, Ronen Israel, and Tobias J Moskowitz. Trading costs of asset pricing anomalies. *Fama-Miller Working Paper*, pages 14–05, 2012.

[24] Gyozo Gidófalvi and Charles Elkan. Using news articles to predict stock price movements. *Department of Computer Science and Engineering, University of California, San Diego*, 2001.

[25] Erkam Guresen, Gulgun Kayakutlu, and Tugrul U Daim. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397, 2011.

[26] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[27] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[28] Geoffrey E Hinton. Connectionist learning procedures. artificial intelligence, 40 1-3: 185 234, 1989. reprinted in j. carbonell, editor,". *Machine Learning: Paradigms and Methods", MIT Press*, 1990.

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

[30] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[31] Geoffrey E Hinton and Ruslan R Salakhutdinov. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pages 1607–1614, 2009.

[32] Tsung-Jung Hsieh, Hsiao-Fen Hsiao, and Wei-Chang Yeh. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied soft computing*, 11(2):2510–2525, 2011.

[33] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522, 2005.

[34] Huseyin Ince and Theodore B Trafalis. Kernel principal component analysis and support vector machines for stock price prediction. *IIE Transactions*, 39(6):629–637, 2007.

[35] Kiyoshi Izumi, Takashi Goto, and Tohgoroh Matsui. Trading tests of long-term market forecast by text mining. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 935–942. IEEE, 2010.

[36] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[37] Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, 38(5):5311–5319, 2011.

[38] Kyoung-jae Kim and Ingoo Han. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications*, 19(2):125–132, 2000.

[39] Sung-Suk Kim. Time-delay recurrent neural network for temporal correlations and prediction. *Neurocomputing*, 20(1):253–263, 1998.

[40] Manfred Krafft and Murali K Mantrala. *Retailing in the 21st Century*. Springer, 2006.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[42] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[43] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, 10:1–40, 2009.

[44] Marcelo S Lauretto, Bárbara BC Silva, and Pablo M Andrade. Evaluation of a supervised learning approach for stock market operations. *arXiv preprint arXiv:1301.4944*, 2013.

[45] Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. Language models for financial news recommendation. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 389–396. ACM, 2000.

[46] 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. Citeseer, 2000.

[47] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[48] Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, and Dan Jurafsky. On the importance of text analysis for stock price prediction. *Unpublished working paper http://www. stanford. edu/˜ jurafsky/pubs/lrec2014_stocks. pdf*, 2014.

[49] Xiaowei Lin, Zehong Yang, and Yixu Song. Short-term stock price prediction based on echo state networks. *Expert systems with applications*, 36(3):7313–7317, 2009.

[50] Andrew W Lo and Archie Craig MacKinlay. Stock market prices do not follow random walks: Evidence from a simple specification test. *Review of financial studies*, 1(1):41–66, 1988.

[51] Sam Mahfoud and Ganesh Mani. Financial forecasting using genetic algorithms. *Applied Artificial Intelligence*, 10(6):543–566, 1996.

[52] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[53] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[54] Anshul Mittal and Arpit Goel. Stock prediction using twitter sentiment analysis. *Standford University, CS229(2011 http://cs229. stanford. edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis. pdf)*, 2012.

[55] M-A Mittermayer. Forecasting intraday stock price trends with text mining techniques. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2004.

[56] Abdel-rahman Mohamed, Tara N Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E Hinton, Michael Picheny, et al. Deep belief networks using discriminative features for phone recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5060–5063. IEEE, 2011.

[57] John J Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[58] Karl Nygren. Stock prediction–a neural network approach. *Master's Thesis, Royal Institute of Technology, KTH, Stockholm*, 2004.

[59] Phichhang Ou and Hengshan Wang. Prediction of stock market index movement by ten data mining techniques. *Modern Applied Science*, 3(12):p28, 2009.

[60] Ping-Feng Pai and Chih-Sheng Lin. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005.

[61] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2006.

[62] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[63] Tobias Preis, Sebastian Golke, Wolfgang Paul, and Johannes J Schneider. Multi-agent-based order book model of financial markets. *EPL (Europhysics Letters)*, 75(3):510, 2006.

[64] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[65] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors, neurocomputing: foundations of research, 1988.

[66] David E Rumelhart and James L McClelland. The pdp research group: Parallel distributed processing: Explorations in the microstructure of cognition. *Foundations*, 1, 1986.

[67] DE Rumelhart, GE Hinton, and RJ Williams. Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986, 1986.

[68] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[69] Karsten Schierholt and Cihan H Dagli. Stock market prediction using different neural network classification architectures. In *Computational Intelligence for Financial Engineering, 1996., Proceedings of the IEEE/IAFE 1996 Conference on*, pages 72–78. IEEE, 1996.

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

[71] Jianfeng Si, Arjun Mukherjee, Bing Liu, Qing Li, Huayi Li, and Xiaotie Deng. Exploiting topic based twitter sentiment for stock prediction. In *ACL (2)*, pages 24–29, 2013.

[72] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[73] Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

[74] Ilya Sutskever, Geoffrey E Hinton, and Graham W Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems*, pages 1601–1608, 2009.

[75] Leigh Tesfatsion and Kenneth L Judd. *Handbook of computational economics: agent-based computational economics*, volume 2. Elsevier, 2006.

[76] Philip Treleaven, Michal Galas, and Vidhi Lalchand. Algorithmic trading review. *Communications of the ACM*, 56(11):76–85, 2013.

[77] Chih-Fong Tsai and Yu-Chieh Hsiao. Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, 50(1):258–269, 2010.

[78] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[79] Paul John Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, volume 1. John Wiley & Sons, 1994.

[80] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[81] Martin Wiesmeier, Frauke Barthold, Benjamin Blank, and Ingrid Kögel-Knabner. Digital mapping of soil organic matter stocks using random forest modeling in a semi-arid steppe ecosystem. *Plant and soil*, 340(1-2):7–24, 2011.

[82] J Welles Wilder. *New concepts in technical trading systems*. Trend Research Greensboro, NC, 1978.

[83] Akira Yoshihara, Kazuki Fujikawa, Kazuhiro Seki, and Kuniaki Uehara. Predicting stock market trends by recurrent deep neural networks. In *PRICAI 2014: Trends in Artificial Intelligence*, pages 759–769. Springer, 2014.

[84] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.