Scalable System for Textual Analysis of Stock Market Prediction

Roy Guanyu Lin
Department of Computer Science
National Taiwan University
Taipei, Taiwan
yesimroy@gmail.com

Tzu-Chieh Tsai
Department of Computer Science
National Chengchi University
Taipei, Taiwan
ttsai@cs.nccu.edu.tw

Abstract—**Stock Market Prediction is a problem that people deal with when they want to predict market trend. For short-term investment, news is one of the most important factors that has influence on stock price. Based on this idea, our target issue is to build a scalable stock market prediction system, which can process Chinese news articles in order to produce a prediction model. With this system, we can speed up the model training process and take into account more training source, e.g., posts from China's microblog service, Sina Weibo. Also, with the emergence of cloud computing, a scalable system can lease more resources from cloud to serve the growing work. Our solution about building this system is using mature open source project, such as Hadoop for parallel computing, Mahout for scalable machine learning, and Jieba for Chinese text segmentation. We provide a basic algorithm for stock trend prediction, build the software stack, collect the news in Taiwan during March 2009 to May 2014 and also run some experiments to evaluate scalability of this system. The result shows that in this application, Jieba Chinese text Segmentation tool can scale well with multiprocessing, namely, 80 percent faster with four parallel processes compared to sequential mode. However, Mahout does not show significant speedup in this scenario.**

Keywords-distributed system; scalability; stock market prediction

I. INTRODUCTION

Stock Market Prediction is a hot topic. There are several ways to deal with this issue. Some examples are fundamental analysis, technical analysis, hybrid analysis, and textual-based analysis. For short-term investments, news can dramatically affect stock price. One of the most famous example is the fake twitter post that Barack Obama had been injured in an explosion, causing the S&P 500 to decline 0.9%. The existing related works about textual analysis targeted the issues for chasing the prediction accuracy [1][2][3]. They use history textual information to train a prediction model, providing an algorithm to get better prediction accuracy. However, our goal is different; we focus on the scalability of Stock Market Prediction System not on the accuracy of prediction model. Because the amount of data has been exploding, we need a scalable platform to deal with large data sets to meet analysis requirement [11]. Textual analysis based stock market trend prediction needs a system with text processing function and machine learning function. However, the traditional tools are not scalable, e.g., CKIP service for Chinese text segmentation broadly used in Taiwan [22] is hard to scale. Therefore, we would like to build a scalable system using mature open source project. There are some benefits of it. First of all, this kind of system saves cost without paying any licensing fees. Second, with scalable system, we can extend the capacity of the system to deal with bigger data set for meeting user requirement, e.g. job completion within given deadline. Third, we can use cloud resource on demand to extend the capacity in pay-as-you-go manner [4].

With the emergence of cloud computing, we can bundle our scalable application into a VM image which is stored on cloud, and launch the instances from the image on demand to start the service. Also, we can adapt application capacity by configuring the amount of cloud resources leased according to workload. In this way, imagine that you have just an old laptop and an access to internet, you can still easily process a big amount of computation by using cloud resources. The amount of cloud resource you need depends on the data input. The way you pay is as you go. You do not have to buy a computer just for some temporary computations. This may save you money. However, deploying scalable system on cloud involves some issues, like how to extend the capacity from cloud, how to save cost when using the cloud resources [5][6]. Same questions appear with our platform, but these questions should be asked after the completion of system and the modeling of system performance [7].

The progress we have made is as follows. First, we create crawlers to collect the data used for stock market prediction, e.g., history news articles and history stock quotes in Taiwan. Second, we design the scalable system for stock market prediction, and build the system based on basic textual analysis based prediction algorithm. Third, we evaluate our main function components, Chinese segmentation tool and machine learning tool. The plan for system performance modeling would be our next goal after finishing the implementation of the algorithm.

The rest of this paper is organized as follows. Section 2 provides an overview description of system workflow, system software stack, and scalability issues. In Section 3, we talk about the basic algorithm of the stock market prediction. In Section 4, we show the preliminary scalability evaluation for Chinese text segmentation and classification model training. Finally, in Section 5, we illustrate future directions and make a conclusion.

II. OVERVIEW OF THE SYSTEM

In this section, we introduce the workflow of our system, the system software stack, and the scalability issues.

A. System workflow

We want to build a system for stock market prediction application, which we can use in Taiwan. We list some requirements for this system. First, we want to have real data to prove the feasibility of the system. To achieve this, we implemented crawlers to collect the history data, including structured data like stock prices and unstructured data like news articles. Second, because we want to make a prediction model from Chinese news articles, our system should have the ability to process Chinese Text. Similar ideas were shown in [1][2], but we deal with Chinese words, not English words. Third, we want to use machine learning technique [10] to train our prediction model, so that machines can learn how to classify a future news a good, bad, or unchanged for the company price. We will describe more details in Figure 1.

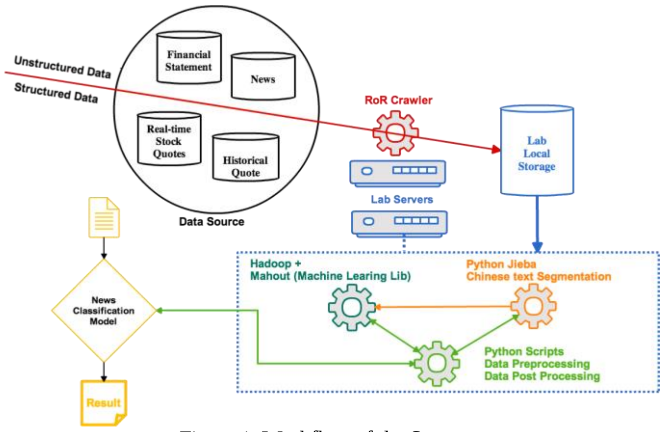
Unstructured Data
Structured Data
Financial Statement
News
Real-time Stock Quotes
Historical Quote
Data Source
RoR Crawler
Lab Servers
Lab Local Storage
Hadoop + Mahout (Machine Learing Lib)
Python Jieba Chinese text Segmentation
Python Scripts Data Preprocessing Data Post Processing
News Classification Model
Result

Figure 1. Workflow of the System.

First, we grab the history data from different sources, and collect these history information in order to evaluate our prediction model. We gathered news articles about Taiwan from the Internet. Because there is no good interface for collecting the news we need, we made automated crawlers using Ruby on Rails (RoR) [12] projects, such as Mechanize [13] and Watir [14]. We collected 566,114 news from 2010 to 2014, totally 2.3GB, and history stock quotes from Taiwan Stock Exchange Corporation (TSE). However, the history stock quote from TSE is daily based, which is too coarse-grain for us. We need finer-grain data if we want to design a more accurate model, so we wrote a program to record per-minute based stock quotes. Based on these two history information, news articles and stock quotes, we can do our prediction model training. The model training process will be described in section three. In this process, we need some tools dealing with Chinese Text Segmentation, classification model training, and also scripts for data pre-processing and post-processing. After we get a news classification model, once a news appear, the system will be triggered, and output whether the news make the company price go up, down, or stay unchanged. The next part will explain the software stack we plan to build. Then, some scalability issues will be discussed.

B. System Software Stack Design and Implementation

Our system architecture is presented in Figure 2. The orange part are the local resources, which can be physical servers or virtualized servers. The purpose of this design is to provide a better utilization of the physical servers. The blue part stands for cloud resources, and we take Amazon for example as our service provider. Amazon Web Services (AWS) [15] provides a lot of web services. Amazon EC2 [16] is one of them, which belongs to IaaS service model. Cloud computing has three service models, e.g., infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) [9]. EC2 provides lots of different specifications of virtual machine to customers. Amazon S3 [17] provides key-value storage service, which can be easily integrated with other products within Amazon. Amazon Elastic MapReduce is the service which offer better abstraction, omitting the steps of building a MapReduce runtime environment. The red part, is a famous scalable Hadoop ecosystem.

We choose Hadoop ecosystem to meet machine learning tool requirement. Mahout is the machine learning library, which is also an Apache project [18], resides on the Hadoop MapReduce stack [19]. This project has three main functions, namely, recommendation, classification, and clustering. In this paper, we use the function of classification. Last, for the green part, it means those scripts for data pre-processing and post-processing, and RoR crawlers.

In the current progress, the crawler implementation have already finished, and we collect 566,114 articles of 2.3GB in size. The software stack has been setup. However, the implementation of the basic stock market prediction is still under development. Which the algorithm will be explained later.

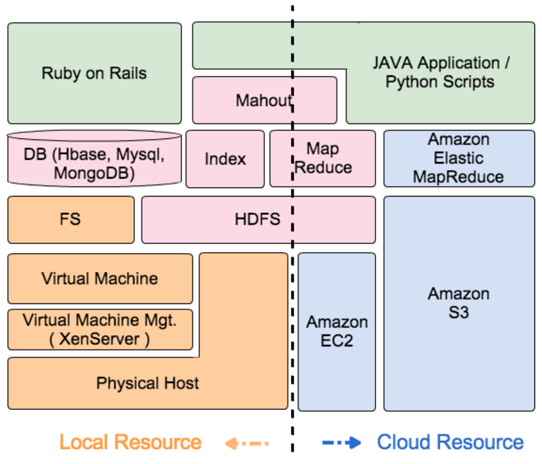

Figure 2. System Software Stack.

C. *Scalability Issues*

Scalability is the ability of a system to handle a growing amount of work. We parallel our computation tasks to get system scalability. Chinese segmentation is one of the main function components, and we discuss scalability of this part as our first step. In our system, data is stored in Hadoop Distributed File System (HDFS). HDFS provides file replication, which can maintain several copies on different machines. Also, due to the nature of data independence for each news articles, the computation can be parallelized. In short, with data replication and data independence, we can parallel Chinese segmentation jobs on those nodes with data copies.

Jieba project [20] originally provides a module to parallel Chinese segmentation process. It cuts a file into several lines, and distributes the lines to several workers in order to increase the throughput. However, when we use this module on our 566,114 Chinese news articles, sized 2.3G, the response time increases from 68 minutes to 99 minutes, shown in Figure 3. Figure 3 compares the completion time of a Chinese segmentation job between sequential mode and parallel mode with 2, 4, and 24 workers. The reason is that each news article is short, the overhead of separating lines to workers is higher than the benefit of parallel computation on multiple workers. Therefore, we change the way from line parallelism to file parallelism, distributing news article files into many workers (processes), as shown in Figure 4.

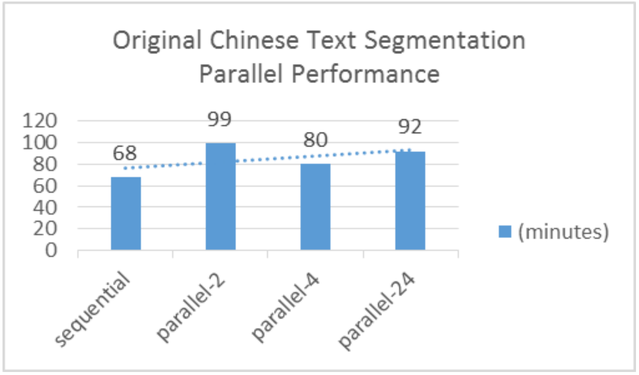

Figure 3. Performance Evaluation of Original Multi-Processes Scaling Up.

In section four, we evaluate the performance of scaling up (vertical scaling) and scaling out (horizontal scaling) of file parallelism version of Chinese text segmentation process using Jieba tool, and describe a problem we met for classification model training using Mahout.

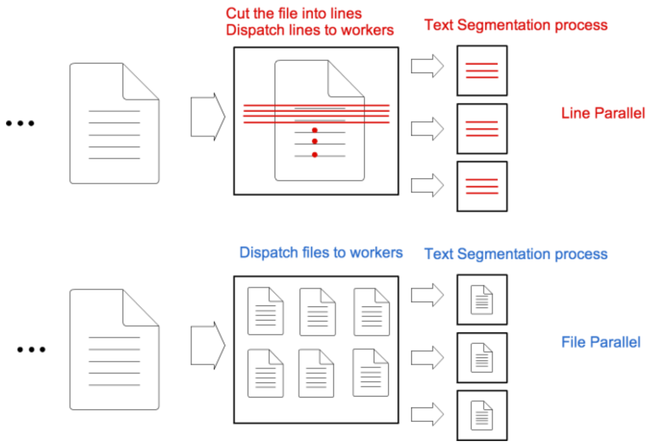
Cut the file into lines
Dispatch lines to workers
Text Segmentation process
Line Parallel
Dispatch files to workers
Text Segmentation process
File Parallel

Figure 4. Original Jieba Line Parallelism and our Inter-File Parallelism.

III. Basic Stock Market Prediction Algorithm

In this section, we introduce our basic market prediction algorithm. Now, we use Chinese text articles to predict whether the stock price goes up, down, or stay unchanged of our target companies. In the future, we will take social media information, e.g., microblog posts, into consideration. The implementation of the algorithm is still under development. By building this algorithm we would like to prove the feasibility of our system for stock trend prediction.

The overall process of the system is as follows. The process consists of two parts. The first part is training the news classification model, and the second part is using the classification model to classify new text articles for predicting the stock trend.

For the second part, assume that we already have a classification model. When a news appears, news sensor detects the events and triggers news classification for each company to see whether this article makes the company stock price up/down, or just stay unchanged. How we get the model of news classification is described below.

The process of training classification model described in Figure 5. At the beginning, with a target company, the script automatically gets history stock quotes from database, filters dates by variation. The variation now is set to 25% variation for an hour. If the stock price goes up over 25% between time points, then we label the time interval "up", vice versa. If it is between -25% and +25%, we label the time interval "unchanged." In this way, we can get time intervals labeled "up", "down", or "unchanged". Then the script searches the news articles related to the company with time and label input, and then tags the articles with label "up", "down", or "unchanged". After labeling, the script triggers Chinese text segmentation. Also, we provide our customized Chinese dictionary to make the segmentation more accurate and do noise filtering. At last, we input the article set with featured words and labels into Mahout Naïve Bayes classification training process [21]. Before training, the script splits dataset into training set and test set. After we get a model, the script evaluates the model by test set and reports the prediction accuracy of classification model.

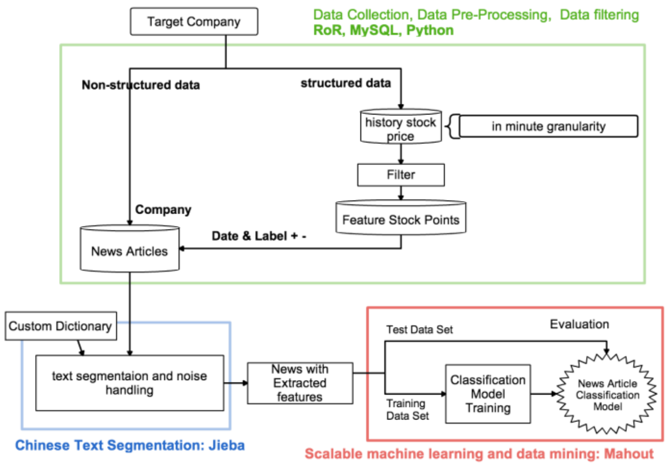
Target Company
Data Collection, Data Pre-Processing, Data filtering
RoR, MySQL, Python
Non-structured data
structured data
history stock price
in minute granularity
Filter
Company
Feature Stock Points
Date & Label + -
News Articles
Custom Dictionary
text segmentaion and noise handling
News with Extracted features
Test Data Set
Evaluation
Training Data Set
Classification Model Training
News Article Classification Model
Chinese Text Segmentation: Jieba
Scalable machine learning and data mining: Mahout

Figure 5. Model Training Process for News Classification.

IV. PRELIMINARY EXPERIMENTS

We did some experiments for evaluating the scalability of file-parallelism version of Jieba and Classification tool Mahout. The experiment environment is using Ubuntu 12.04 OS, running on one 24 cores 16G RAM server, and three 8 cores 8G RAM server.

We use Jieba Python version to process 2.3G Chinese news, totaled 566,114 articles. We use process pool to create many parallel processes to segment the news articles. Figure 6 shows the result of scale-up performance improvement on the 24 cores server. Compared to sequential version, four processes parallelism is 80% faster. In addition to scale-up (vertical scaling) experiments, we still made scale-out (horizontal scaling) experiments. Figure 7 shows the scale-out experiment of processing 2.3G news articles on the 8 cores 8GB server in sequential mode. We split data into two copies and three copies using scripts to three servers, and test the performance. As our expectations, the performance improvement is almost linear; for the constant module, loading time is small compared to workload computation. In the future, we think about integrating job parallelism with HDFS. HDFS default stores three copies for every chunk; so, we do not need to write one more script to deal with the data split action.

We did several experiments to test the scalability of Mahout. The experiment is conducted on one node, two nodes, and three nodes runtime environment. However, we could not get a significant performance improvement. To find out the reason, we decompose the auto script into small steps for Mahout Naïve Bayes Classification and record its latency. We found that the first step, seqdirectory, the command of which makes the files in HDFS sequential, always produces just one map task in the job. It means we cannot parallel the computation in this step. Usually, the number of map tasks is related to the number of chunks in HDFS. Our data size is more than 640MB. If it combines all the small files into a big one, it should at least have 10 chunks with default chunk size 64MB. We have not found a solutions yet. We tried to configure Hadoop several times, but failed. Now, we are still tracing Mahout source code for solving the problem. The latency of every steps in Naïve Bayes classification is depicted in Figure 8.

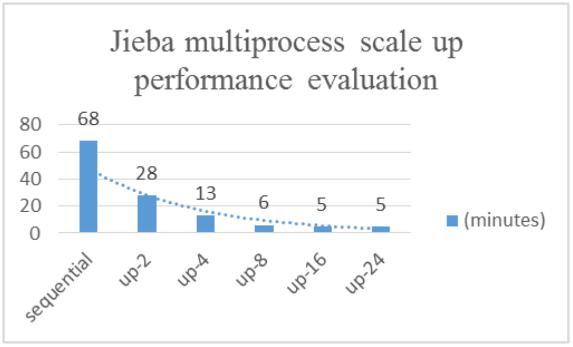

Figure 6. Performance Evaluation of File Parallelism Scaling Up.

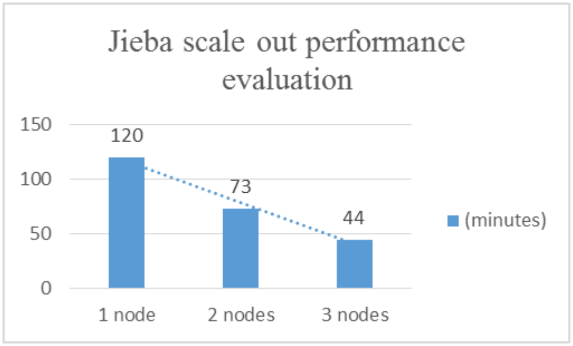

Figure 7. Performance Evaluation of File Parallelism Scaling Out.

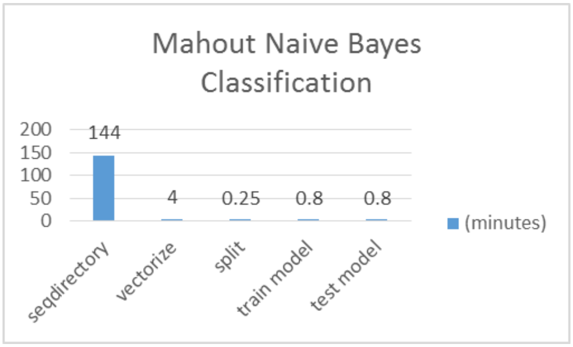

Figure 8. Performance Evaluation of Mahout Naive Bayes Classification.

V. FUTURE DIRECTION AND CONCLUSION

In this paper, after the preliminary result, we proved that Jieba can scale well with our file parallelism version, i.e., scaling-up with four cores gets 80% faster compared to one core environment and scaling-out makes linear improvement. The main factors of scalability are the nature of data independence and data replication. Text segmentation process can be executed in a parallel way. However, the performance improvement of Mahout is limited by the first step, i.e., file sequential process. Because the bottleneck has huge impact on overall response time of classification model training process, we have to deal with it in the future. We are still solving this problem by tracing Mahout source code. It is worth mentioning that Mahout starts to move its focus on Spark [8], a new popular large scale data processing project stating faster processing speed because of in-memory computation. We will use Spark to solve the scalability issue of machine learning function in another way.

In addition to Mahout and Jieba, we will also evaluate another components in our system to prove scalability as soon as we finish building our system. Also, we will use queueing theory to build system performance modeling. With performance model, we can adapt system resource to make performance meet user requirements. Also, we will consider the issues about offloading to cloud. For example, when will we need extra resources, how to offload computations to cloud, and how to use cloud resources in a cost-aware way.

Although the work is not finished yet, we believe this is a good issue worth discussing. The era of big data is coming, a scalable system for this kind of application is needed. Because we may develop new prediction algorithm based on bigger data source, e.g., social media information, with the sharing of the experience, we believe it is helpful to give readers a hint to build a scalable system for textual analysis based stock market trend prediction.

REFERENCES

[1] Fung, Gabriel Pui Cheong, Jeffrey Xu Yu, and Wai Lam. "Stock prediction: Integrating text mining approach using real-time news." Computational Intelligence for Financial Engineering, 2003. Proceedings. 2003 IEEE International Conference on. IEEE, 2003.

[2] Schumaker Robert P., 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 (2009): 12.

[3] Lavrenko, V., Schmill, M., Lawrie, D., Ogilvie, P., Jensen, D., & Allan, J. "Mining of concurrent text and time series." KDD-2000 Workshop on Text Mining. 2000.

[4] Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee et al. "A view of cloud computing." Communications of the ACM 53.4 (2010): 50-58.

[5] Sharma, U., Shenoy, P., Sahu, S., & Shaikh, A. (2011, June). "A cost-aware elasticity provisioning system for the cloud." Distributed Computing Systems (ICDCS), 2011 31st International Conference on. IEEE, 2011.

[6] Guo, T., Sharma, U., Wood, T., Sahu, S., & Shenoy, P. J. "Seagull: intelligent cloud bursting for enterprise applications." Proceedings of the Usenix Annual Technical Conference (short paper). 2012.

[7] Ganapathi, A., Chen, Y., Fox, A., Katz, R., & Patterson, D. "Statistics-driven workload modeling for the cloud."Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on. IEEE, 2010.

[8] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. "Spark: cluster computing with working sets."Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. 2010.

[9] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

[10] Naïve Bayes Classification. (2014)Retrieved July 3 , 2014, from http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

[11] Manyika, James, et al. "Big data: The next frontier for innovation, competition, and productivity." (2011).

[12] Ruby on Rails Web Framework. (2014). Retrieved July 2, 2014, from http://rubyonrails.org/

[13] Mechanize Ruby Gem. (2014). Retrieved July 2, 2014, from https://rubygems.org/gems/mechanize

[14] Watir Ruby Gem. (2014). Retrieved July 2, 2014, from https://rubygems.org/gems/watir

[15] Amazon Web Service. (2014). Retrieved July 3, 2014, from http://aws.amazon.com/

[16] Amazon Web Service Elastic Compute Cloud (EC2). (2014). Retrieved July 3, 2014, from http://aws.amazon.com/ec2/

[17] Amazon Web Service S3. (2014). Retrieved July 3, 2014, from http://aws.amazon.com/s3/

[18] Apache Mahout Project. (2014). Retrieved July 4, 2014, from https://mahout.apache.org/

[19] Apache Hadoop Project. (2014). Retrieved July 4, 2014, from http://hadoop.apache.org/

[20] Jieba Project for Chinese Text Segmentation. (2014). Retrieved July 4, 2014, from https://github.com/fxsjy/jieba

[21] Mahout Naïve Bayes. (2014). Retrieved July 5, 2014, from https://mahout.apache.org/users/classification/bayesian.html

[22] CKIP Chinese Text Segmentation Tool. (2014). Retrieved July 3, 2014, from http://ckipsvr.iis.sinica.edu.tw/