Stock Recommendation System

Yuechen Qin, Bowen Dang, Zheng Fang, Guangyang Zhang
Electrical Engineering
Columbia University
Email: zf2150@columbia.edu, gz2202@columbia.com, yq2158@columbia.edu, bd2384@columbia.edu

***Abstract*—Predicting stock market movement is always difficult due to its chaotic nature. Conventionally, thumb rules and intuition seems to be the major indicator. Noticing that it might be easier to predict a stock A's movement based on stock B's movement, our project proposes a general framework for multiple events to predict two stocks' dependency. By encoding a time series as a string, we can measure dependence effectively based on string distances. We apply this technique with a default criterion of maximum price follows the opening price movement for three days in succession in our project, which, can be further customised.**

The stock recommendation system is a web application which recommends stocks with highest similarities to clients' favourites. Client view the realtime info of a specific stock and select their best-likes through drop down buttons and search bar. The System crawls Nasdaq stock infos from Yahoo Finance in a real-time manner and calculates similarities using Jaro Winkler Algorithm. The system recommends 5 stocks for every client favourite and presents the information of these stocks in a table in UI.

***Keywords*-components: stock; similarity; recommendation; big data; Jaro Winkler Algorithm; web application;**

I. Introduction

The stock market, based on real time transactions, is a new and important component of the contemporary capitalist financial system[8]. Recently, the number of shares trades on the stock exchange market is growing tremendously and expectations point to an even faster growth[9]. Considering the scenario that the move of stock market plays an important role for companies to obtain investments, take strategic decisions and expand its activities on a global level, many specialists are motivated to take advantage of the observed behavior of shares to help avoid instability within a company, or further, to predict its behavior in the near future. Though it is possible to use the historical data of stock market to do prediction for investment[6], accurate predictions are still very hard to make due to the chaotic nature of stock markets, which might depend on external events like governmental actions, company crisis and many other factors.

Our project suggested a way that, instead of trying to predict one particular stock's movement with respect to the whole market, it might be easier to predict a stock A's movement based on stock B's movement, which turned out to be the main motivation of our stock recommendation system[3]. Based on a customer's specific choices, our system is able to recommend a number of similar stocks (both short-term results and long-term results are provided). Besides, stock trend figures and real-time stock value are all available on our system.

The rest of the report is structured as follows. Section II reviews the related work and presents the background to follow the report. Section III describes the system structure of our project. Section IV shows in detail how the Jaro Winkler Algorithm works in our calculation of similarity. Section V gives a overview of the software package to be open sourced. Section VI shows the experimental results obtained by our recommendation and Section VII presents the conclusions.

II. Related Works

Much research efforts has gone into the field of similarity-based approach for stock prediction. In stock market analysis, the behavior of share pricing is usually described by discrete-time series, where each term of the series is a set of attributes regarding the price variations[8]. The indicators, Price Ratio and Price Comparison do compare two stocks' prices, but this is for finding strong stocks rather than dependencies between stocks. People have used clustering techniques[5], mining association rules from database of transactions[11] and geometric properties[7][2] to find stock similarity. The metric distances and methods based on topological properties and random matrix theory have also been used for the purpose.

The representation of time series as an univariate linear stochastic process is the development basis of the Autoregressive Model (AR) and the Moving Average Model (MA), both members of the Autoregressive Integrated Moving Average (ARIMA) model[10], which has been one of the most popular approaches to time series forecasting. The ARIMA model takes advantage of both weak-stationary and non-stationary properties of time series and proposes to integrate autoregressive and moving averages, offering ways to identify seasonalities observed at the series micro-level.

It is based on three parameters: lags of difference, the difference order term and the lags of forecast errors.

When non-linearities are present in the data, non-linear methods including the Autoregressive Conditional Heteroskedasticity (ARCH) method and its variations are adopted to model share price behavior[1]. These models have some drawbacks, such as the lack of scalability of the HMM technique and the limitation of the method of Analogues to handle long periods of time and high-dimensionality series.

Soft-computing approaches based on artificial intelligence methods have suggested new ways to forecast time series outcomes[4]. These methods usually aim to predict trends using a classifier that gives an interpretation of summarized data. Among these approaches, Artificial Neural Networks (ANN) have been one of the most popular tools used in recent works regarding the financial market.

III. System Overview

1) *System Structure*
We designed a stock recommendation system which can recommend stocks with highest similarities to clients? favourites. The system uses a J2EE framework, it has one UI page, several serve lets, a recommendation algorithm called "Jaro Winkler" algorithm and a mysql database. Figure 1 shows an overview of project structure. The project components are explained in detail in the following:

UI and servlets:
The UI helps clients choose their favourite stocks through drop-down buttons and search bar. They can view the realtime trend of every stock by clicking the **plot** button. The drop down buttons helps them make a multiple selections based on that suits their requirement. After client makes their own choices, the UI will send client favourites to the back-end and present recommendation results from back-end in a table. Our UI is written in html and is decorated with CSS and Bootstrap. It has a group of javascript functions to interact with clients in a dynamic manner.
Servlets connect UI and java code block of the project. It passes the input from UI to a java program and hands out the return data from java program to the UI. The project contains four servlets, which is used to calcite similarity, calculate distance between two stocks and extract stock information from Yahoo Finance.

Yahoo Finance API:
We crawls Nasdaq stock information from the Yahoo Finance API. The URL of this API is "http://finance.yahoo.com/d/quotes.csv?s=". We select bid, 50 day moving average, day high and day low as our attributes and store these informations into our database.

java blocks:
The system has two java packages, which are: data model and similarity calculation model.
The data model are the classes we used to store and process the stock information locally. For each clients' best-like stock, the system builds up a CustomerStock object to store its symbol and a list which contains 5 recommended stocks with the highest similarities to it. Every recommended stock is stored in the recommendation list as a RecStock object. It contains fields such as its similarity to clients' favourites, its bid, moving average, day high and day low.

The similarity calculation model are the java programs we used to calculate similarity.

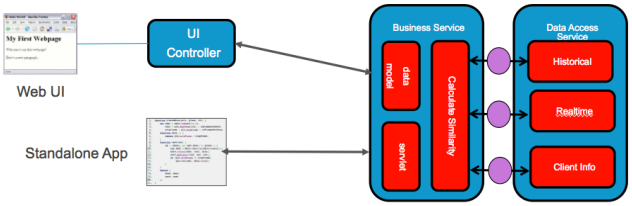

Figure 1. System Structure

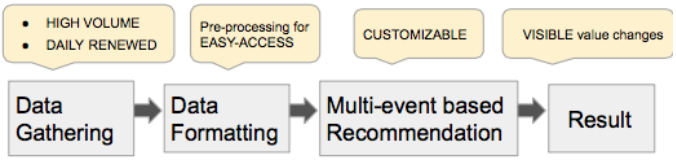

Figure 2. Flow map

2) *Dataset*

- We use NASDAQ Stock Exchange Data as our dataset. It contains 2959 stocks in total. As mentioned above, we use Yahoo Finance dataset to acquire historical prices of each stock. The dataset is daily renewed with most recent value via Yahoo API.

We choose MYSQL as our database because it is the most popular open source database and it is most deployed database in the web and cloud(with 80% deployments integrated)

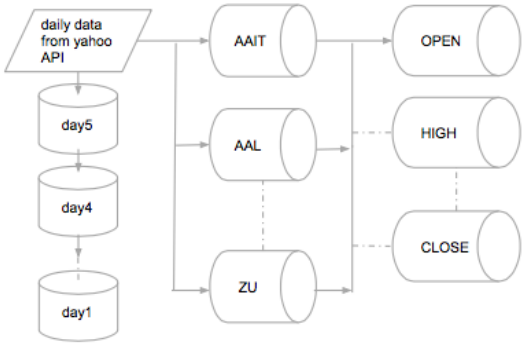

Figure 3. Dataset

IV. Algorithm (you can change this section name)

Manilla et. al. introduced the idea of an event for time series in [2]. An event is an occurrence of a particular type which has a time stamp attached to it. Given a set $\overline{E}$ of event types, an event α is a pair of $(e,t) \in \overline{E}$, where e is an event tye and t is an integer. the occurrence time of the event.Mathematically, a time series can be seen as an ordered sequence of events $\{\alpha_i\} = \{e_i, t_i\}$ where e_i is an occurrence of a particular type and t_i is the time stamp.For stock time series, the set of event types can be (daily opening price, daily closing price, daily trading volume, daily high price and daily low price) and the time series can be described as any of these four event types by giving the value associated with the date.

An episode is a sequence of events.Manilla et. al. defines an episode as a triple $(V, <, g)$where V is a set of nodes and $<$ is a partial order of V, and $g : V \to \overline{E}$ is a mapping associating each node with an event type.
Note that stock history can be seen as a series of events, we are able to define an episode as a predefined sequence of events. For example, considering "daily volume" as an specific event,we can define an episode "three successive days of volume increase"($T_i < T_{i+1} < T_{i+2}$, T_i = Trading Volume value on i^{th} day). If we can define stock move in a time wondow as a single episode, we are able to analyze its property more easily.

However, we notice that an episode is only defined by a specific event type and its ordering.However, it is far from enough for a complex market. For example, we are not able to define "high price follows the move of open price"($(O_{i+1} - O_i) * (H_{i+1} - H_i) > 0$, O_i) = Trading open value on i^{th} day and H_i) = Trading high value on i^{th} day) as an episode using the above definition.This problem is because that an episode depends on a single event. To solve this problem, we now extend the space of a single event to a multi-event set.We now define our new event set as follows:
$E = \{(e_1, t_1), ..., (e_i, t_i), (e_l, t_l)\}$, $\forall i = (1, 2, ..., l)$
$e_i = \{s_1, ...s_k, q : s_m \leftrightarrow s_n\}$, $\forall m, n = (1, 2, ..., k)$
where q is a set of mapping among events in event set e_i, which implies a complex relationship amoung stock events such like open price and volume.

Give this definition of a complex event, we are now able to find multi-event episodes from stock time series, as show in the following:

1. Read time series $S(t)$, $t = 1, ..., N$
2. Read episode definition
   $E = \{(e_1, t_1), ..., (e_i, t_i), (e_l, t_l)\}$, $\forall i = (1, 2, ..., l)$
3. episode_index = 1
   complementary_episode_index = 1
4. for every event e_i do
   if e_i is an valid event at time t_i
   push (e_i, t_i) to valid_times[j]
   end if end for
5. find combinations of $(e_i, t_i), i = 1, ..., m$ that either q or complements of q is satisfied
   If q is satisfied, push that to
   episode[episode_index] episode_index++;
   end if
   If complements of q is satisfied, push that to
   complementary_episode[complementary_episode_index]
   complementary_episode_index++;
   end if

After we have defined the multi-event episode, we are able to define a specific event set with a single symbol. For example, if we would like to check the inner relationship between two stocks that "high price and low price has a negative relationship within a week's period", we are able to define this episode as event "a" and its counterexample as event "b". The algorithm we use for encoding is as follows,

1. Read episode
   $E = \{(e_1, t_1), .., (e_l, t_l)\}$, $\forall i = (1, 2, ..., l)$ 2. $B = \emptyset$
   for each $i = 1, ...N$, N = lehgth of checked time series, do
   if t_i belongs to a member of episode description
   $B = B.append("a")$
   else if t_i belongs to a member of complementary episode description
   $B = B.append("b")$
   else
   $B = B.append("c")$
   end do
3. return B

This way, we can turn time series comparison problems into a string similarity checking.After we've got two equivalent ternary strings, similarity between these two

stocks can simply seen as measured string distance, which will be discussed later.

In our further study, we find that these multi-event question can be altered to a combination of a set of single events, so we look back to our initial definition of event set and notice their inner relationship. Now that we are using static event to do similarity evaluation, we are able to alter a complex language to a much simpler combination of equalization checking. For example, we can abstract most of the restrictions as combinations of "in a k day window, s_m and s_n has a negative/positive relation", which means that it will be more efficient for us to deal with the large amount of stock data if we have a pre-encoded event description. That way, we only have to add the mapping information every time we want to check a different property or inner relationship of the stock market. Our pre-encoded content of each event set like in the following:

1. Read episode
   $E = \{(e_1, t_1), .., (e_l, t_l)\}$, $\forall i = (1, 2, ..., l)$ and
   $e_i = \{s_1, ...s_k, q : s_m \leftrightarrow s_n\}$, $\forall m, n = (1, 2, ..., k)$
2. $A_1, ...A_k = \emptyset$
   for each $j = 1, ...k$, $k =$ length of event space, do
   for each $i = 1, ...N$, $N =$ length of checked time series, do
   if s_j belongs to a member of episode description
   $A_j = A_j.append("a")$
   else if t_i belongs to a member of complementary episode description
   $A_j = A_j.append("b")$
   else $A_j = A_j.append("c")$
   end do
   return A_j
   end do
3.Alter $q : s_m \leftrightarrow s_n$ to $Q : A_m \leftrightarrow A_n$
4. return $E_i = \{A_1, ...A_k, Q : A_m \leftrightarrow A_n\}$

In the above encoding, we use Jaro-Winkle algorithm to check whether s_j is a member of episode description or its complement. It is also used for our future comparison work. Jaro(see e.g., Winkler 1985, 1989, Winkle and Thibaudeau 1990) introduced this string comparator to measure the aprtial agreement between two strings. The string comparison algorithm accounts for length of strings and partially accounts for the types of errors typically made in alphanumeric strings by human beings. It is used to adjust exact agreement weights when two strings do not agree on a character-by-character basis.

Specifically, the Jaro string comparator is

$$d_j = \begin{cases} 0 & if \quad m = 0 \\ \frac{1}{3}(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{|s_1|}) & otherwise \end{cases}$$

where

m is the number of matching characters

t is the number of transpositions

$|s_1|$ is the length of string in first file

$|s_2|$ is the length of string in second file

Two characters are considered in common only if they are no further apart than $L/2 - 1$ where $L = \max(|s_1|, |s_2|)$.

JaroWinkler distance uses a prefix scale p which gives more favourable ratings to strings that match from the beginning for a set prefix length ℓ. Given two strings s_1 and s_2, their JaroWinkler distance d_w is:

$d_w = d_j + (\ell p(1 - d_j))$

where

ℓ is the length of common prefix at the start of the string up to a maximum of 4 characters

p is a constant scaling factor for how much the score is adjusted upwards for having common prefixes

In our preparation work, we encode the episode we want to check as a string, of which the length is the time window. we have realized Jaro_Winkle algorothm, we are able to check whether s_j is a member of episode description or its complement by doing the following:

if $s_j = string\_episode$
$A_j = A_j.append("a")$
else if
$Jaro\_Winkle.compare(t_i, string\_episode) = 0$
$A_j = A_j.append("b")$
else $A_j = A_j.append("c")$
end if

After we look deeper into our encoding process and analyze the experimental result,we notice that the condition e_i belongs to neither a member of episode description nor its complement will bring a considerable impact on checking the similarity when we increase the window length such like episode "In a 2 week period, trading volume will follow the move of open price". Also considering the property of JarO_Winkle algorithm, we do the following change to our encoding algorithm:

1. Read episode
   $E = \{(e_1, t_1), .., (e_l, t_l)\}$, $\forall i = (1, 2, ..., l)$ 2. $B = \emptyset$
   for each $i = 1, ...N$, $N =$ lehgth of checked time series, do
   if t_i belongs to a member of episode description
   $B = B.append("a")$
   else if t_i belongs to a member of complementary episode description
   $B = B.append("b")$
   else if $episode\_window < 4$,where
   $episode\_window$ is the number of days included in an episode
   $B = B.append("c")$
   end do
3. return B

1) *UI overview*

The User interface of our stock recommendation system contains our project title, main page figure, three drop-down buttons and recommendation search bar. The function of these components are explained in detail in the following.

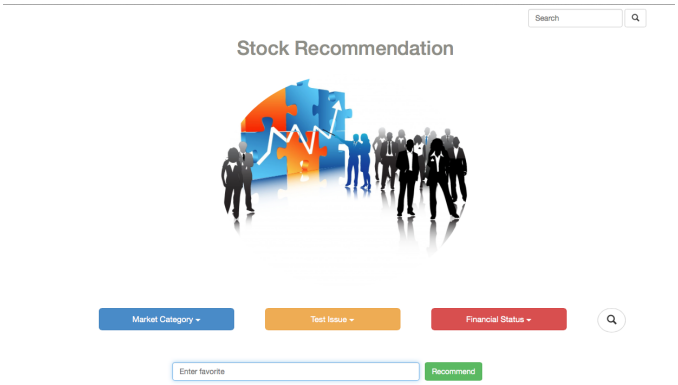

Figure 4. UI overview

2) *Drop down button*
The first step of our recommendation process is to let the client choose his favourite stocks. Clients can make selections through the three drop-down buttons, which are market category, test issue and finical status. When client chooses different options, the content of the related search bar will also change. On the right side of the drop-down buttons there is a round search button, a table is extracted from our database which presents all the stocks that fit in the prerequisites.

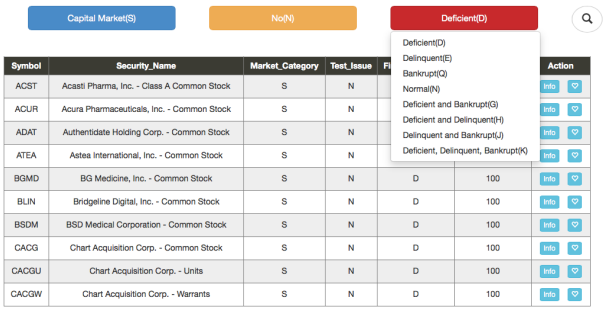

Figure 5. Drop down buttons

3) *Recommendation*
Suppose that the client has chosen two stocks as his favourites, after he clicking the **recommend** button, the system send the information to the back-end, which will calculate the similarities of all the stocks from Nasdaq stock market to the two stocks and presents the results in a table.

The system recommends 5 stocks with highest similarities for every choice. The table also presents the general information for the recommended stocks such as similarity, todays bid, moving average, day high and day low.

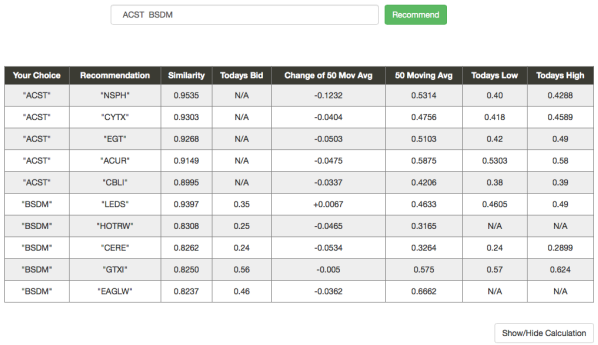

Figure 6. Recommendation results

4) *Search Button*
Additionally, we also have a search bar, which is used to search for a specific stock. If you type **AAPL** in the search bar, it will show you the stock information of apple. The format is exactly the same as the result of **info** button.

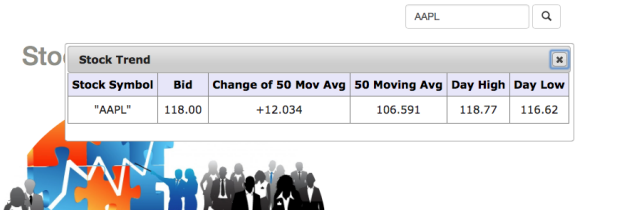

Figure 7. Search results

5) *Info and like button*
In the last cell of each row, we have two function buttons: **info** and ♡. The ♡ is clicked if the client likes the stock. When client clicks the ♡ button, the symbol of the stock will be automatically added to the recommendation search bar.
Finally, if you want to see the historical line graph of a stock, just click on the **info** button and then the line graph is presented as in Figure 1.
As is shown in Figure 2, a dialog called ”historical prices line graph” pops up when clicking the ”info” button. The horizontal coordinate stands for the dates and the vertical coordinate stands for the open prices

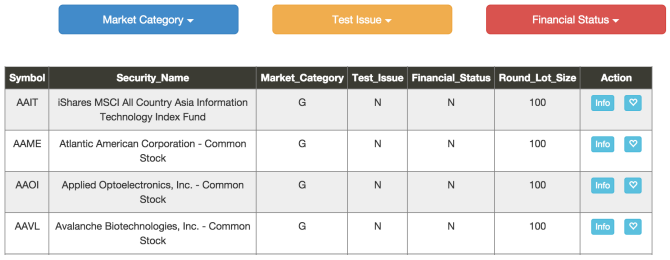

Figure 8. info button

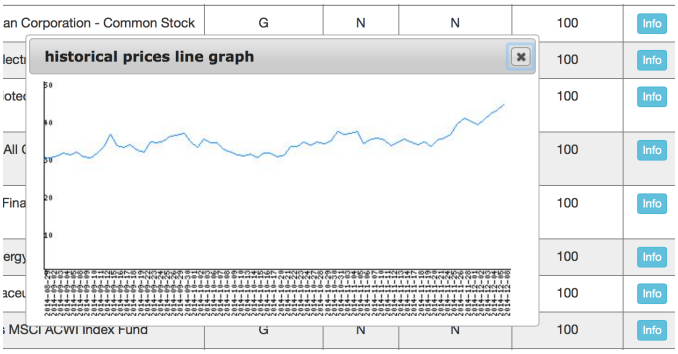

Figure 9. line chart

of the selected dates. The number of recent days' price, which shows the trend of the stock price, can be helpful for stock buyers to get a general sense of the chosen stock.

We decide to realize our purpose using the following steps:
1) to connect to the MySQL database and draw a line chart and store it using PHP;
2) to use the HTML file to invoke the PHP file and pass to it the parameters;
3) to fetch and show the line chart drawn via a HTML file.
We use PHP to connect to the MySQL database and draw the line chart using the historical stock prices. And then use html to call and execute the PHP file so that the picture can be presented.
The reason why I choose PHP to draw the picture is that it is easy to connect to the database and to draw a picture at the front end. First, I use the mysql_connect() function to connect to the database. Then I use mysql_select_db() function to choose the stock database. I use the following statements to receive the symbol transmitted from HTML to get the symbol of the stock for the query:
$symbol = $_GET['symbol'];
$sym = (string)$symbol."_NASDAQ";
Then I utilize mysql_query() function to fetch the date and open price of the given stock by its symbol. After getting the data, I use the mysql_fetch_row() function to traverse the rows of the given table and then array_slice() to get the latest given number of days' stock price data.

Having fetched the data needed, we begin to draw the line chart. In the first place, I use the associative array $price to store the data with key equal to date and value equal to the stock price. To initialize, use the imagecreatetruecolor() to create a new true color image, imagecolorallocate() to set the colors, and imagefill() to perform a flood fill starting at the given coordinate with the given color in the image. I set the total image to be 500 by 300 pixels. X-axis and Y-axis labels can be easily set by imagestringup() function with iterations. And then I can draw the X axis and Y axis using imageline() function. The following key step is to convert the $price associative array to the $point associative array in order to make the stock price correspond to the pixels of the image so that I can draw the picture. Lastly, I use imagesetpixel() to draw the dots and imageline() to connect the dots with lines. The jpeg image can be created using imagejpeg() function and stored to the designated directory.

The last thing is to make sure the PHP file is connected and can be invoked by the HTML file which creates our GUI. We use JavaScript to program the behavior of web pages. I create a function called plot() to be called when the button "info" is clicked. The symbol of the stock is stored in the "value" variable as described by the following 2 statements:

var cell=document.getElementById("t02").rows[a].cells;
var value=cell[0].innerHTML;
We use xmlhttp.open("GET","http://localhost/plot_400_300.php?symbol="+value,true) function to call the "plot_400_300.php" file and to pass the "symbol" parameter to the PHP file. To solve the time delay between the drawing and storing functions and the fetching and displaying functions, we decide to store our line chart under the web content of eclipse working directory in order to use eclipse's automatic refreshing feature.

V. Experiment Results

We are now able to use rather complex episodes for our stock market evaluation. Since it is hard for us to get access to commercial databases, we use stock information provided by yahoo finance to show that our similarity

evaluation is reasonable.

As discussed before, our system is able to recommend 5 most similar stocks for investors based on user defined episode criterion. In our following experiment, we set our basic episode "high price will have a positive relationship with open price in a k day period", and we set different k value and training data space to see the performance of our recommendation. With this criterion, similarity between stock A and stock B implies that their three day following patterns are similar. So if short trading on A is profitable, we can expect a profitable trading on B soon after.

In our experiment, we randomly pick stock JSM for further evaluation.Because of space constraint, we give a representation of the most similar 3 stocks. We first choose 50 days data as our training data and set the time window $k = 3$.The open and high price of JSM in the last 50 days is shown in Figure 11.

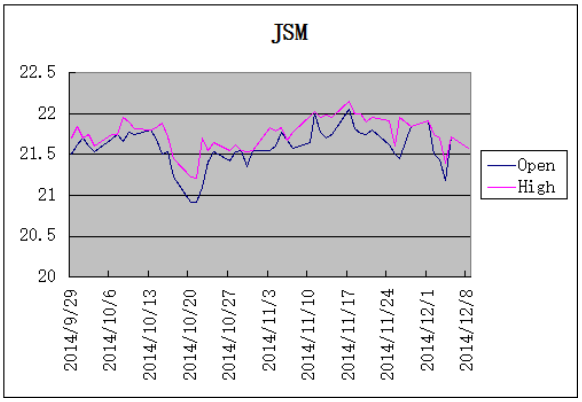

Figure 10. Stock time series snapshot for JSM

Our recommendation result are stock IEP, CTRP and BDSI, as shown in Figure 2, Figure 3 and Figure 4. As we can clearly see from the figures, stock IEP and JSM shows high cirrelation, their trend line suggests a good similarity. Also, if we look at some specific area in the figures around 2014/10/20, 2014/11/17 and 2014/12/8, we will find that the trend of our recommended stocks all have similar trend as our selected stock, which means that our recommendation is helpful in deciding similar stocks.

Next, we set a $k = 7$ time window, which means that we are going to evaluation similarity between two stocks in a longer period. First we keep the 50 date training dataset and get stock CZNC, OCLR and STRM as JSM's most similar stocks, shown in Figure 5, 6 and 7.

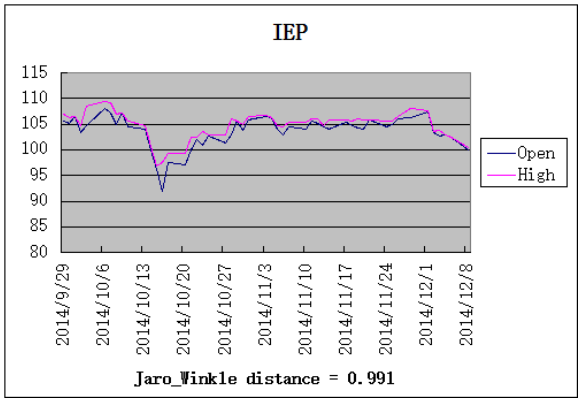

Figure 11. Stock time series snapshot for IEP

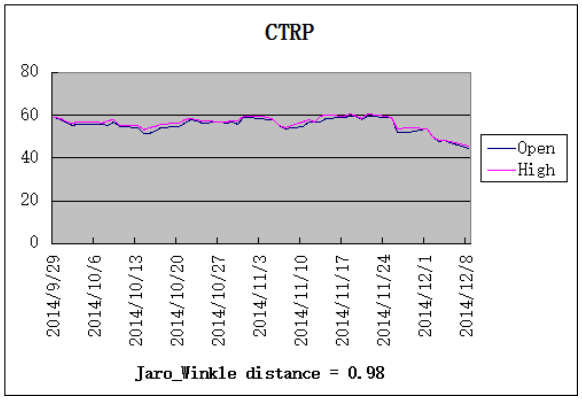

Figure 12. Stock time series snapshot for CTRP

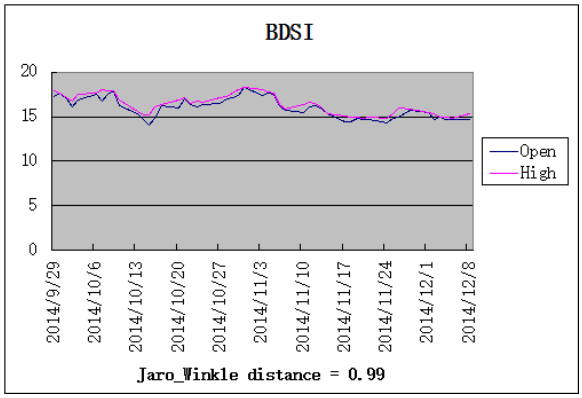

Figure 13. Stock time series snapshot for BDSI

As we can see from the figure,STRM still shows high similarity if we look at the general behavior of these two stocks, whereas CZNC and JSM shows a far apart behavior. After analyzing our algorithm, we found that for a long time window, we will need data from a longer time series for the string to be built. So we extend the training data space that the system is able to do a recommendation based

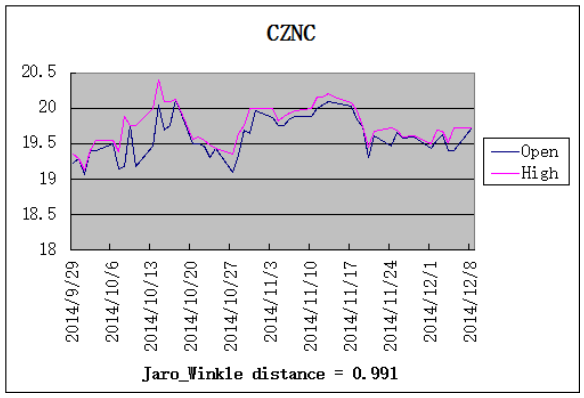

Figure 14. Stock time series snapshot for CZNC

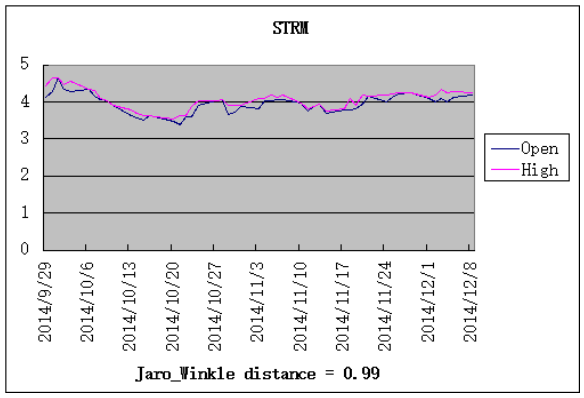

Figure 15. Stock time series snapshot for STRM

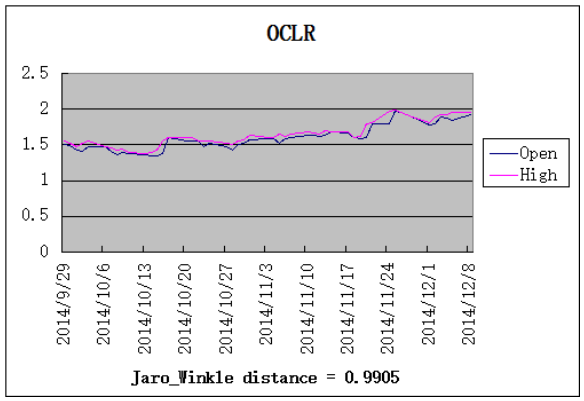

Figure 16. Stock time series snapshot for OCLR

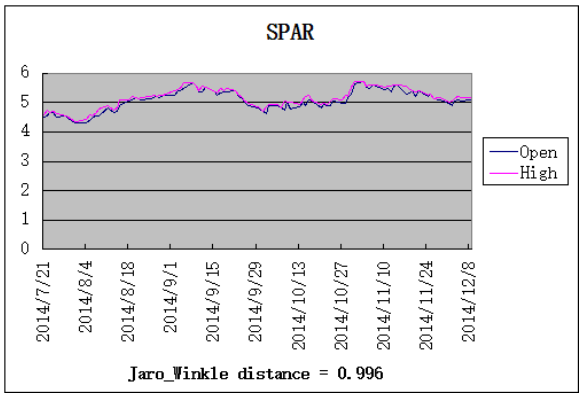

Figure 17. Stock time series snapshot for SPAR

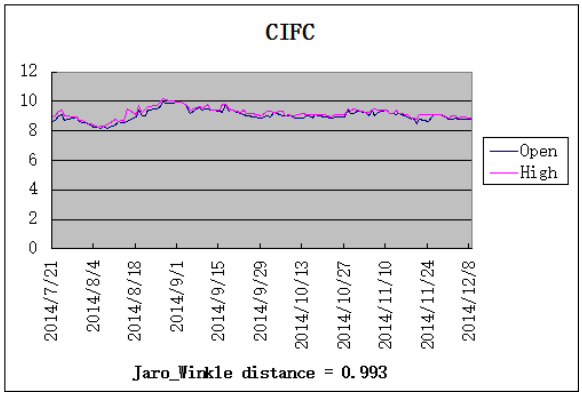

Figure 18. Stock time series snapshot for CIFC

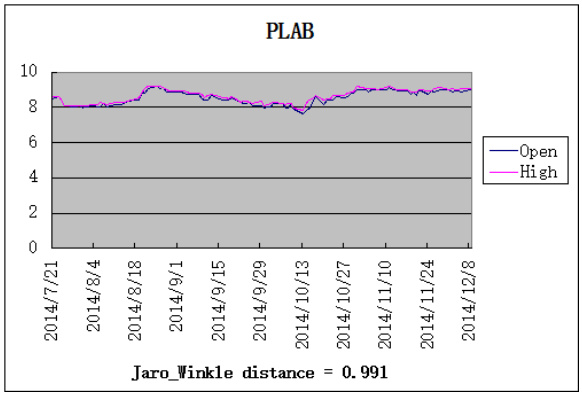

Figure 19. Stock time series snapshot for PLAB

on stock behavior in the last 1000 days. In that case, we are able to get the 3 most similar stocks:SPAR, CIFC and PLAB.

An interesting finding about the result we get is that the general behavior of these three stocks seems similar to each other, but not quite the same as JSM. However, after looking at the graph with our criterion, we noticed that in each period of 7 days, the high price do follows the move of open price for both JSM and any of the recommended stocks, which means that we have already picked the similar stocks that matches the criterion best. However, a problem with this criterion is that it doesn't

cover the behavior of JSM. Supposing that a user-selected stock doesn't show a long term behavior often, it is hard to find its similar stock correctly using such criterion.In that case, if we want recommendation to be correct enough, it is important for the user to choose a criterion that works well with the selected stock.

VI. Conclusion

Our stock recommendation system proposed a general framework for multiple events to predict two stocks' dependency. It allowed users to quickly find out their objective stocks from the screening results by three categories(market category, test issue, financial status) and check the most similar ones with their own favorite stocks. The time series of a stock is encoded as a string and stock similarity is measured based on string distances by Jaro Winkler Algorithm. The experiment results clearly reveal that the recommended stocks by our system indeed bear a similar trend with a certain given stock, thus prove the validity of our algorithm.

Future works of our project involves enable users to define different criterion to be described mathematically as set of events with restrictions on value and time. The contributions of each team member is: Yuechen Qin(25%), Guangyang Zhang(25%),Bowen Dang(25%),Zheng Fang(25%).

Acknowledgment

Our group would like to thank Prof. Lin for providing us such an opportunity to explore the amazing world of big data. We've really learned a lot during this semester and this kind of experience is quite unforgettable and rewarding. Besides, we really appreciate the efforts of all the TAs. We'd like to thank them for their patience and great help during the semester.

Appendix

Instructions on how to implement the application

1) Project Composition

The project document has three folders: J2EE resources, jars and figures.

- The folder called J2EE resources contains all the elements we need inside the J2EE framework. Figure 9 shows how we organise these elements inside J2EE. It contains java packages, web contents and all the related structures.
- The folder called jar contains all the jars needed to be installed into the lib of the WEB-INF folder inside J2EE framework.
- The folder called figures contains the figure of the outcome of our UI.

2) Configuration steps:

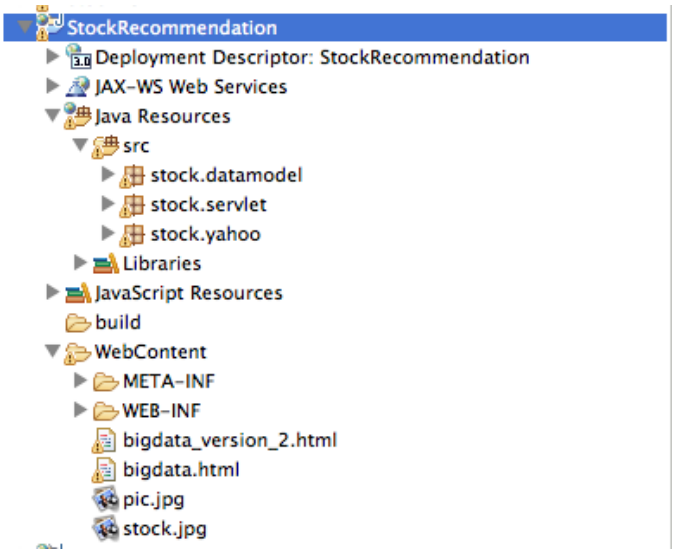

Figure 20. info and like buttons

The folder called J2EE resources contains all the elements we need inside the J2EE framework. Figure 9 shows how we organise these elements inside J2EE. The step is as follows:

- Three java code packages are put inside "Java resources/src" folder.
- $bigdata\_version\_2.html$ and $stock.jpg$ are put inside WebContent/WEB-INF folder.
- jars are put inside the lib under WEB-INF folder.

- Tomcat 7 is used as the server to run the project.

3) How to run the project:

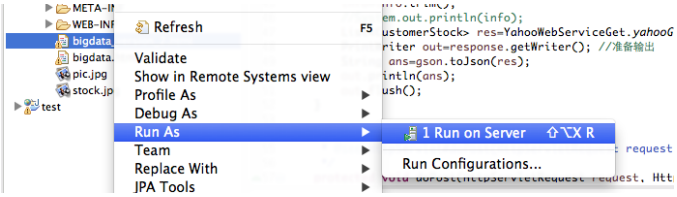

Figure 21. info and like buttons

- The project runs from $bigdata\_version\_2.html$. right click mouse, choose run as server and click finish, the UI webpage will appear in the browser at localhost:8080.
- Client can make selections through the three drop-down buttons, which are market category, test issue and finical status.
- After selection, by clicking the search button, the database will return all the stocks that fit in the prerequisites.
- If client likes the stock, he can click the like button to add the symbol of the stock to the recommendation search bar.
- Click recommend button will activate the system to calculate the similarities of all the stocks compared with clients' choice and presents the recommendation results in a table.
- The search bar on the top-right corner of the webpage is used to search for a specific stock. If you type AAPL in the search bar, it will show you the stock information of apple.

REFERENCE

[1] Tim Bollerslev, Ray Y Chou, and Kenneth F Kroner. Arch modeling in finance: A review of the theory and empirical evidence. *Journal of econometrics*, 52(1):5–59, 1992.

[2] Béla Bollobás, Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Time-series similarity problems and well-separated geometric sets. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 454–456. ACM, 1997.

[3] Abhi Dattasharma, Praveen Kumar Tripathi, and G Sridhar. Identifying stock similarity based on multi-event episodes. In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pages 153–162. Australian Computer Society, Inc., 2008.

[4] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.

[5] Martin Gavrilov, Dragomir Anguelov, Piotr Indyk, and Rajeev Motwani. Mining the stock market (extended abstract): which measure is best? In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 487–496. ACM, 2000.

[6] C.W.J. Granger. *International Journal of Forecasting*, 8:3–13, 1992.

[7] Jan Komorowski and Jan Zytkow. *Principles of data mining and knowledge discovery*. Springer, 1997.

[8] Daniel S. Kaster Caetano Traina Jr. Marcos Vinicius Naves Bedo, Davi Pereira dos Santos. *A Similarity-Based Approach for Financial Time Series Analysis and Forecasting*. Springer Berlin Heidelberg, 2013.

[9] da Hora D.N. de J.R. Palotti M.Meira W. Pappa G.L. Martinez, L.C. *From an artificial neural network to a stock market day-trading system:a case study on the bmf bovespa*. IEEE Press, 2009.

[10] Terence C Mills and Raphael N Markellos. *The econometric modelling of financial time series*. Cambridge University Press, 2008.

[11] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD Record*, volume 25, pages 1–12. ACM, 1996.