Information and Software Technology 51 (2009) 1140–1149

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A stock recommendation system exploiting rule discovery in stock databases

You-Min Ha [a], Sanghyun Park [a], Sang-Wook Kim [b,*], Jung-Im Won [b], Jee-Hee Yoon [c]

[a] *Department of Computer Science, Yonsei University, Republic of Korea*
[b] *College of Information and Communications, Hanyang University, 17 Haeng-Dang, Seong-Dong, Seoul 133-791, Republic of Korea*
[c] *Division of Information Engineering and Telecommunications, Hallym University, Republic of Korea*

ARTICLE INFO

Article history:
Received 27 December 2005
Received in revised form 30 May 2008
Accepted 6 June 2008
Available online 13 March 2009

Keywords:
Stock databases
Rule discovery
Rule matching

ABSTRACT

This paper addresses an approach that recommends investment types to stock investors by discovering useful rules from past changing patterns of stock prices in databases. First, we define a new rule model for recommending stock investment types. For a frequent pattern of stock prices, if its subsequent stock prices are matched to a condition of an investor, the model recommends a corresponding investment type for this stock. The frequent pattern is regarded as a rule head, and the subsequent part a rule body. We observed that the conditions on rule bodies are quite different depending on dispositions of investors while rule heads are independent of characteristics of investors in most cases. With this observation, we propose a new method that discovers and stores only the rule heads rather than the whole rules in a rule discovery process. This allows investors to impose various conditions on rule bodies flexibly, and also improves the performance of a rule discovery process by reducing the number of rules to be discovered. For efficient discovery and matching of rules, we propose methods for discovering frequent patterns, constructing a frequent pattern base, and its indexing. We also suggest a method that finds the rules matched to a query from a frequent pattern base, and a method that recommends an investment type by using the rules. Finally, we verify the effectiveness and the efficiency of our approach through extensive experiments with real-life stock data.

1. Introduction

Around us, there are a variety of objects such as stock prices, temperature values, and money exchange rates whose values change as time goes by. The list of changing values sampled at a fixed time interval is called *time-series data* [1,12,14,15]. Time-series data reflect the status changes of objects as time passes in applications. In many applications, an element value in time-series data is affected by its preceding values accumulated [6]. Thus, by analyzing past element values in time-series data, we can find the regularities and also build their model, thereby predicting the values to appear in the future.

Stock price sequences are a typical example of time-series data [3,9]. Since the goal of stock investors is to maximize their earnings, it would help them achieve successful investments to recommend proper buying and selling points via analysis of the stock price sequences.

Time-series analysis [5] has been a well-known method for predicting stock prices in the future. It is classified into two categories: time domain analysis and frequency domain analysis. *Time domain analysis* is based on the regression model, which assumes that a current value is determined by the regression of its preceding values [5]. *Frequency domain analysis* is primarily used in analyzing stationary time-series data for predicting macroscopic tendencies of months, seasons, or years. However, they cannot reflect the conditions for investments, which could be dynamically changed by investors, and also have a problem of not being appropriate for short-term predictions.

From a machine learning perspective, there have been some methods proposed for predicting future values by analyzing past values with the neural network [19,17,18,20]. For reflecting the various conditions of investors, however, these methods should construct a neural network for each condition, therefore, incur a large storage overhead in main memory. Also, they have a difficulty in applying themselves to a huge database environment due to their inherent scalability problem.

In a database perspective, there have been research efforts on rule discovery and matching in time-series data [8,16]. Ref. [8] proposed a method that transforms time-series sequences into symbol sequences and then discovers rules from them. For transformation, it extracts a number of *windows* of a fixed length from each time-series sequence, and classifies all the windows into multiple groups by using a clustering technique [10]. It assigns a symbol to each window group, and converts every window in a time-series sequence into its corresponding symbol. Finally, all the time-series sequence into its corresponding symbol. Finally, all the time-series

* Corresponding author.
E-mail addresses: ymha@cs.yonsei.ac.kr (Y.-M. Ha), sanghyun@cs.yonsei.ac.kr (S. Park), wook@hanyang.ac.kr (S.-W. Kim), jiwon@hanyang.ac.kr (J.-I. Won), jhyoon@hallym.ac.kr (J.-H. Yoon).

sequences become a set of symbol sequences. Ref. [16] introduced a concept of *elastic rules*, and also proposed a method that discovers the elastic rules effectively by using the *suffix tree*. For suffix tree construction, this method uses the *TAH-tree* [7] in order to convert time-series sequences into symbol sequences. Also, there have been several methods which try to correlate stock values to external entities. Ref. [21] proposed a neuro-genetic stock prediction system based on financial correlation between companies, and Ref. [22] proposed a novel stock prediction system based on the correlation between the web sentiments and the stock values.

A rule consists of a *rule head* and a *rule body*. A common feature of the prior methods proposed in a database community is to find both the heads and the bodies of rules in a rule discovery process. This implies that the conditions on the heads and the bodies should be defined earlier than the beginning of the rule discovery process. As mentioned in Section 2, however, such conditions used for recommending stock investment types are highly dependent on dispositions of investors. Thus, the prior methods that discover whole rules inherently have two problems: (1) They produce a large number of rules and (2) they cannot reflect the conditions newly defined by users on the set of rules obtained from the previous rule discovery process.

In this paper, we propose a novel approach for discovering and matching of rules that solves these problems, and discuss how to apply it to stock investment. Unlike the prior methods that discover both heads and bodies of rules in advance, our approach discovers only the rule heads and stores them in a *frequent pattern base*. If a user raises a query on a stock of his/her interest, it finds a rule head matched to a query from a frequent pattern base, and then discovers its rule body at this point. With this strategy, the proposed approach allows users to define various conditions on rule bodies at any time. Our contributions are summarized as follows.

(1) We define a new flexible rule model to recommend stock investment types. This allows investors to define various conditions on rule bodies flexibly at querying time.
(2) We propose a method to discover frequent stock patterns, each of which corresponds to a rule head. Unlike the prior methods that discover both heads and bodies of rules in a rule discovery process, our method discovers only the rule heads and stores them as frequent patterns. It processes rule bodies individually at querying time. This improves the performance of a rule discovery process significantly by reducing the number of rules to be discovered.
(3) We propose a method that constructs a frequent pattern base by using the frequent patterns thus discovered and builds its index for efficient rule matching. We could construct only a single pattern base for all frequent patterns. For more efficient rule matching, we classify frequent patterns according to their length and then build a separate index for each class.
(4) We propose a method for efficient rule matching in the frequent pattern base, thereby recommending investment types to users through the matching result. Since the proposed method enables stock investors to easily adapt the query model to suit their needs or application environments, it provides a fundamental framework for adaptive recommendation systems for stock investment.
(5) To show the effectiveness and the efficiency of the proposed approach, we conduct performance evaluation via extensive experiments.

The paper is organized as follows. Section 2 presents the problem we are going to solve, and briefly provides the overview of the proposed approach. Section 3 presents our methods for building and indexing of the frequent pattern base. Section 4 describes our methods for the rule matching and the investment type recommendation. Section 5 shows the results of performance evaluation. Finally, Section 6 summarizes and concludes the paper.

2. Overview

This section briefly presents the basic concept of the proposed approach. First, Section 2.1 presents a motivating example, and Section 2.2 explains the rule model used in our approach. Section 2.3 discusses main features of the proposed approach.

2.1. Motivating example

Fig. 1 shows the changing patterns of ten sequences of stock prices, $\text{Stock}_i (0 \leqslant i < 10)$. The vertical axis represents a stock price, and the horizontal axis does a relative time when each pattern occurs. $RH - Start_i$ and $RH - End_i$ mean the starting and ending points of the time range named by a rule head. Similarly, $RB - Start_i$ and $RB - End_i$ mean the starting and ending points of the time range named by a rule body. In this figure, we see that all the sequences show exactly the same pattern in the time range of the rule head.

Let us examine the price changing patterns of ten stock sequences in the time range of the rule body, which follows the rule head after the time interval of t. In every sequence, we observe that the average stock price in the rule body gets more than 20% higher in comparison with the last stock price in the rule head. By this observation, we can expect that, when a new sequence shows the same pattern as those in the rule head, it would increase more than 20% after t time interval since then.

Suppose that an investor knows this tendency in advance. If his/her stock of interest shows the same pattern as those in the rule head, the investor expects its price will grow significantly after time interval t. Thus, this investor buys that stock, thereby having a chance to get a high return. The goal of this work is to develop a system that automatically recommends stock investment types by analyzing the changing patterns of past stock sequences.

2.2. Rule model

In this paper, we use the following form of a rule to express the trend of changing stock prices. Here, H and B denote a rule head and a rule body, respectively. This rule represents that B happens after time t since H has occurred.

$$H \xrightarrow{t} B(s, c)$$

In stock applications, H is an event corresponding to an appearance of a pattern P within the time range of the rule head as shown in Fig. 1. Also, B is an event that corresponds to the characteristics of stock prices within the time range of the rule body as shown in Fig. 1. In Fig. 1, for example, B can be represented as 'INCREASE' because the average prices in the rule body get 20% higher. Like this, investors specify their own conditions, which are related to the investment types for recommendations, on the characteristics of stock prices within the time range of the rule body. Such conditions are called *rule body conditions*, which decide when the characteristics of stock prices are regarded as 'INCREASE'. In the prior example, investors set the rule body condition as "the average stock price in the time range of the rule body increases more than 20% in comparison with the last stock price in the time range of the rule head". In this case, the changing patterns of stock prices in Fig. 1 are discovered as a meaningful rule. We note that these rule body conditions vary depending on dispositions of investors.

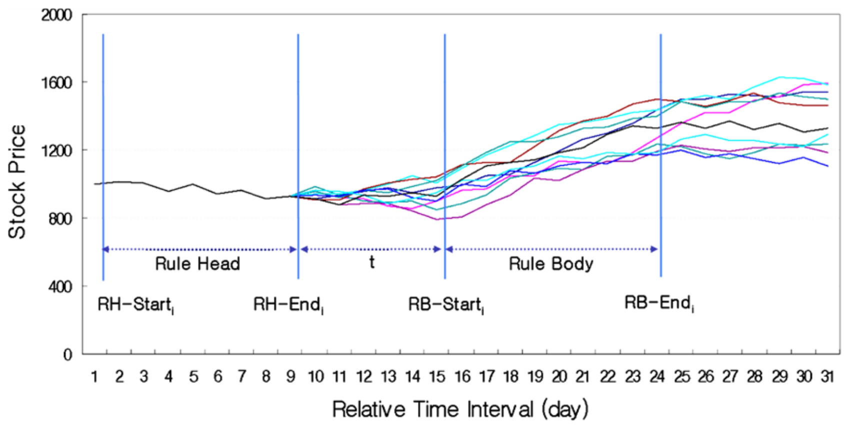

Fig. 1. Changing patterns of stock prices.

Next, we discuss (s, c) in the rule. A changing pattern can be formed as a rule only when a sufficient number of stock sequences support the pattern. s defined in the following is called a *support* which means how many times the pattern P corresponding to H appeared in past stock sequences.

s = *support* (H) = (number of occurrences of a pattern P corresponding to H) $\times$ 100 $\div$ (number of occurrences of patterns whose lengths are same as that of the pattern P corresponding to H)

Also, for becoming a meaningful rule, a set of sequences that satisfy the above support should show a similar tendency in the time range of the rule body. c defined below is a *confidence*, which represents how many stock sequences matched to H satisfy the condition on B together.

c = *confidence* (H, B) = (number of occurrences of patterns that are matched to H and also satisfied with the condition on B) $\times$ 100 $\div$ (number of occurrences of patterns that are matched to H)

Our approach discovers those rules whose support and confidence are both larger than predetermined thresholds during analyzing the past stock sequences. If a recent changing pattern of an investor's stock of interest is matched to some H, it recommends an investment type by referring to its B. Possible investment types to be recommended are 'BUY', 'SELL', 'HOLD', and 'NO RECOMMENDATION'. They are decided by conditions for B, which are highly dependent on propensities of investors.

2.3. Discussions

Previous approaches discover both the head and body of rules in a rule discovery process. Thus, investors have to define all their conditions for B before a rule discovery process starts. As mentioned earlier, however, these conditions vary according to investors. For example, an investor wants the system to recommend 'BUY' when the *average price* in the time range of the rule body gets more than 20% higher against the last price in the time range of the rule head. On the other hand, another investor wants the system to recommend 'BUY' when the *lowest price* in the time range of the rule body gets more than 10% higher against the last price in the time range of the rule head. Moreover, we note that such conditions even for the same investor are changeable according to investing points.

In such environment, previous approaches [5,19,17,18] that discover both rule heads and bodies at the same time suffer from the following problems: (1) There are a large number of rules to be discovered since they find all the rules that satisfy various conditions specified by different investors. This makes a rule discovery process take long time, and also incurs serious storage overhead for maintaining large amount of rules. (2) After a rule discovery process is finished, new investors cannot define their conditions for B. Also, existing investors cannot add or redefine those conditions. Therefore, they hardly provide flexibility in rule discovery. For reflecting new conditions, they have to perform a rule discovery process again, thus incurring a large number of redundant steps.

Another possible approach is to perform rule discovery and matching together for a given query at the time the query is issued. That is, without discovering all the rules for a whole database, it discovers only the rules matched to a given query at the querying time, examines their bodies, and finally recommends a corresponding investment type. This approach solves the two problems mentioned above, but deteriorates the performance of query processing because it should analyze a whole database every time each query is issued.

For solving the problems of these approaches, we observed the followings: (1) conditions for B vary according to the propensities of investors; (2) H is hardly affected by the propensities of investors. Thus, in this paper, we propose a new approach that discovers and stores only the rule heads rather than whole rules in a rule discovery process. We call a set of the rule heads stored a *frequent pattern base*. When an investor raises a query for his/her stock of interest, the approach first finds rule heads matched to the query from a frequent pattern base, and then examines whether the event, which satisfies the conditions on B defined by him/her, occurred in the history. As a result, the proposed approach allows investors to define the conditions on B flexibly, and also resolves the performance problem in a rule discovery process.

What deserves discussing is a target from which we discover rules. One way is to discover rules from a whole database containing all the stock sequences. Another way is to discover rules from each stock sequence. In the former case, we understand the tendency of a price movement for whole stocks. In the latter case, we grasp the tendency of a price movement for a given stock. The proposed approach is applicable to both cases. In performance study in Section 5 to examine the satisfaction ratio and query processing performance, we focus our attention on the latter case in order to identify the characteristics of each stock.

3. Construction of a frequent pattern base

In this section, we propose an efficient approach to discover frequent patterns from a stock database. We also explain how to orga-

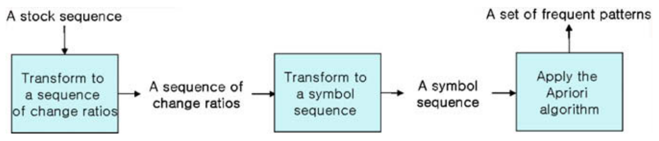
A stock sequence
Transform to a sequence of change ratios
A sequence of change ratios
Transform to a symbol sequence
A symbol sequence
Apply the Apriori algorithm
A set of frequent patterns

Fig. 2. Brief process flow for finding frequent patterns.

nize frequent patterns into a frequent pattern base and how to index them.

3.1. Discovery of frequent patterns

A patten is called a *frequent pattern* when it has a support larger than a predefined threshold called a *minimum support*. As explained in Section 2.2, only the patterns which are frequent can be used as a rule head in the proposed rule model. Fig. 2 shows a process to discover frequent patterns from each sequence stored in a stock database.

Preprocessing step: The value ranges of elements may differ from one stock sequence to another. In addition, even within a single stock sequence, the element values may fluctuate a lot in response to dynamic economic situations or changes in business revenue. Therefore, it is rarely possible for a stock sequence with raw element values to contain frequent patterns. To solve this problem, we take the approach to transform a raw stock sequence into a sequence of *change ratios* and then into a symbol sequence. More specifically, each element $s[i](0 \leqslant i < n)$ of a raw stock sequence S is first transformed into $s'[i](0 \leqslant i < n-1)$ by the expression shown below:

$$s'[i] = \left(\frac{s[i+1] - s[i]}{s[i]}\right) \times 100$$

For example, a raw sequence $S = \langle 5000, 5100, 4900, 5400\rangle$ is transformed into a sequence of change ratios $S' = \langle 2.00\%, -3.92\%, 10.20\%\rangle$. A sequence of change ratios S' is then transformed into a symbol sequence S'' via *categorization*. Categorization is an operation which divides a value range of elements into a set of non-overlapping categories. Via categorization, each element of S' is converted to a symbol of the category to which the element belongs. Note that two elements whose values are different from each other may be represented by the same symbol. As a result, the probability for a stock sequence to contain frequent patterns becomes higher. The equi-width method [11] could be employed to categorize the value ranges of elements. However, in case of the Korean stock market, the daily price change limit is ±15 percent of the previous day's closing price, and the change ratios between adjacent elements are distributed quite nonuniformly over the range, much more around 0% and very few near -15% or +15%. Therefore, we employ the equi-depth categorization method [11] where each category contains almost same number of elements.

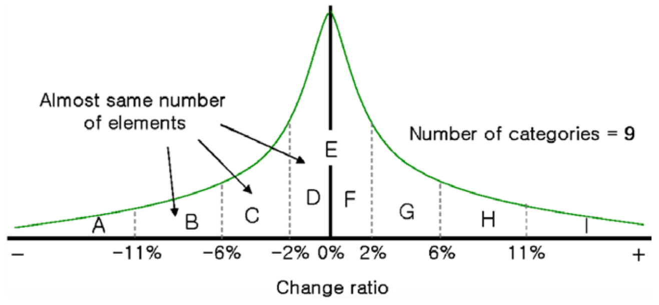

Fig. 3. A set of equi-depth categories.

Fig. 3 shows nine categories obtained by the application of the equi-depth categorization method to a sample stock sequence. According to these categories, for example, the sequence of change ratios $S' = \langle 2.00\%, -3.92\%, 10.20\%\rangle$ is converted to the corresponding symbol sequence $S'' = \langle F, C, H\rangle$. Our algorithm to discover frequent patterns begins with each of the symbol sequences obtained through a series of aforementioned transformations.

Algorithm to discover frequent patterns: Frequent patterns embedded in a given symbol sequence S'' can be easily discovered by extracting every subsequence α from S'', computing α's support Sup_α, and comparing Sup_α with a predefined minimum support MinSup. Although this method is simple to implement, it has to scan the whole sequence in disk whenever computing the support of each subsequence. Therefore, this method incurs large CPU and disk I/O overheads.

To overcome this problem, Refs. [3,2,4] proposed the *Apriori* algorithm where the concept of candidate patterns was exploited. A pattern is called a *candidate pattern* when it is not judged as infrequent. For example, let us suppose that only $\langle A\rangle$ and $\langle B\rangle$ are frequent among three patterns $\langle A\rangle, \langle B\rangle$, and $\langle C\rangle$ of length 1. Patterns $\langle AB\rangle$ and $\langle BA\rangle$ are possible to be frequent while patterns $\langle AC\rangle$ and $\langle BC\rangle$ are not. This is because patterns $\langle AC\rangle$ and $\langle BC\rangle$ include an infrequent subpattern $\langle C\rangle$. That is, a pattern is possible to be frequent only when all of its subpatterns are frequent.

To discover frequent patterns from a symbol sequence S'', we utilize the Apriori algorithm as follows. We first discover a set of frequent patterns F_1 of length 1 by scanning S''. We then perform the *self-join* of $F_{k-1}(2 \leqslant k \leqslant Len(S''))$ to produce a set of candidate patterns C_k of length k. For each pattern α in C_k, we scan S'' to decide whether the support of α is greater than or equal to a predefined minimum support MinSup. If α's support is greater than or equal to MinSup, then α is inserted into F_k. We repeat the above process with incrementing k until no candidates in C_k are judged as frequent and thus there are no elements in F_k. If there are no elements in F_k, we cannot generate candidate patterns any more. Therefore, the algorithm terminates its execution.

Construction of frequent pattern bases: Frequent patterns discovered in the previous step are stored in a *frequent pattern base*. A frequent pattern base must be organized to support an efficient rule matching. For achieving this goal, we construct a frequent pattern base in a B-tree structure. The entries to be stored in a B-tree are pairs of ⟨frequent pattern, list of positions at which the frequent pattern occurs⟩. That is, the string representing a frequent pattern is used as a key and the pointer to the list of its occurrence positions is stored at a leaf node. Each occurrence position of a frequent pattern α is denoted as a pair of ⟨SID, offset⟩ where SID is the identifier of a stock sequence within which α occurs and offset is the starting point of the stock sequence from which α appears.

We could construct only a single pattern base for all frequent patterns from a stock sequence. For more efficient rule matching, however, we classify frequent patterns according to their length and then build a separate index for each class. That is, when k is the length of the longest frequent patterns, we build k B-trees. The physical locations of all B-trees are stored in an index table. An index table contains k entries and its ith entry points to the root node of the B-tree that has been built from the frequent patterns of length i. Since the size of an index table is not large in most cases, it can be resident in main memory.

Fig. 4 illustrates the proposed frequent pattern base constructed from a sample stock database. This example assumes that the length of the longest frequent patterns is k. Let us consider a frequent pattern DBF in this example. Since it is of length 3, it is stored in the B-tree which is pointed by the third entry of the index table. Also, it occurs three times in the stock sequence identified by SID_1 and twice in the stock sequence identified by SID_2. By organizing a frequent pattern base in this manner, we are able to locate

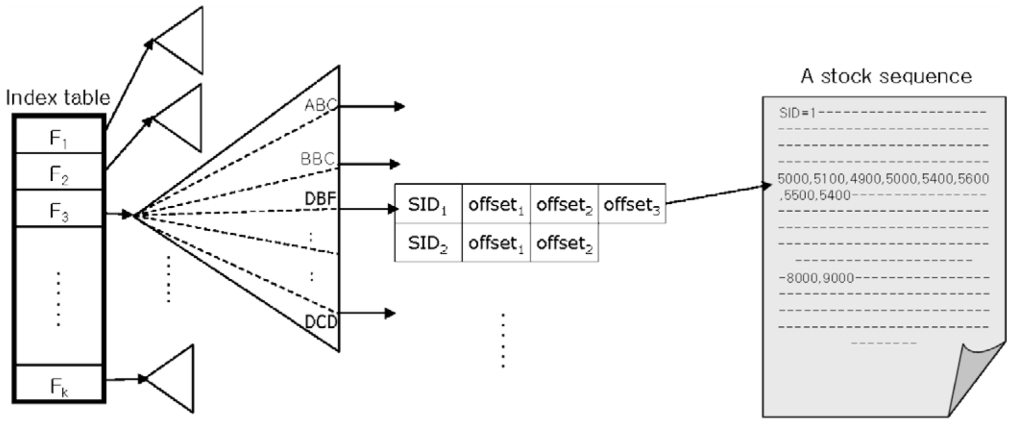

Fig. 4. Overall index organization for a set of frequent patterns whose maximum length is *k*.

the occurrence positions of the frequent patterns matched to a given query pattern.

4. Rule matching and recommendation

In this section, we propose a method for searching a frequent pattern base for the rule heads matched to a condition of a stock investor, and a method for recommending an investment type from the rule bodies coming after the matched rule heads. More specifically, we explain the basic idea of the proposed method in Section 4.1 and the detailed procedure of rule matching and investment recommendation in Section 4.2.

4.1. Basic idea

Our method mainly consists of two steps. The first step is for rule matching where we search a frequent pattern base for the rule heads matched to a condition given by a stock investor. The second step is for recommending the most promising an investment type after analyzing the rule bodies following the matched rule heads. Queries for requesting recommendation types are specified by the following expression through which a stock investor indicates the stock item of interest and the conditions determining his/her stock trading activities.

$$Q = (\mathsf{item}, \mathsf{QP}, \mathsf{t}, \mathsf{bodyLen}, [\mathsf{minHold}, \mathsf{maxHold}], \mathsf{minConfidence})$$

In the above expression, item is the stock item of interest and $\mathsf{QP}(= \langle q[0], q[1], \ldots, q[Len(QP) - 1]\rangle)$ is a query pattern representing the most recent changing pattern of item. As shown in Fig. 1, t denotes the time interval between the end of rule heads and the beginning of rule bodies[1], bodyLen denotes the length of rule bodies. For determining investment types, we inspect each of the rule bodies of length bodyLen beginning after t time units from the end of the rule heads matched to QP. [minHold, maxHold] denotes the range of average increase ratio for retaining the current stock item. Given a rule head *H* and a rule body *B*, their *average increase ratio* is defined by the following expression:

$$AverageIncreaseRatio(H, B) = \frac{\text{average price of } B - \text{last price of } H}{\text{last price of } H} \times 100$$

One of such investment types as ‘BUY’, ‘SELL’, ‘HOLD’, and ‘NO RECOMMENDATION’ is chosen in response to the value of average increase ratio. That is, we recommend ‘BUY’ when the value of average increase ratio is larger than maxHold, ‘HOLD’ when it is between minHold and maxHold, and ‘SELL’ when it is smaller than minHold.

As an example, let us suppose that the rule head matched to QP is ⟨5000, 5100, 4900, 5000⟩ and the three elements coming after t time units from the end of this rule head are 5600, 5500, and 5400. Let us also suppose that bodyLen of a user query is 3. Since the value of the last element of the rule head is 5000 and the average value of the elements in the rule body is 5500, the value of average increase ratio is 10%. If [minHold, maxHold] of a user query is [−5%, 5%], then we would recommend ‘BUY’ to an investor.

For a rule to be meaningful, the changing patterns of its bodies must be similar. The *confidence* of a rule represents how similar the changing patterns of its bodies are. If the confidence of a rule is less than minConfidence specified in a user query, then the rule is not considered meaningful. ‘NO RECOMMENDATION’ is delivered to a user in such a case. minConfidence must be set to at least 50% to prevent more than one investment type from being recommended.

Note that the query model explained above is just an illustration. That is, the above query model is for helping readers to understand the basic concept of the proposed method easily. This query model can be adapted in accordance with specific needs of users. For example, according to the dispositions of users, different values can be assigned to query parameters such as t, bodyLen, minHold, maxHold, and minConfidence. Moreover, a new metric rather than the average increase ratio can be employed as a basis for determining the investment type. For example, conservative investors may want to use the *minimum increase ratio* rather than the average increase ratio.

$$MinimumIncreaseRatio(H, B) = \frac{\text{lowest price of } B - \text{last price of } H}{\text{last price of } H} \times 100$$

Since the proposed method enables stock investors to easily adapt the query model to suit their needs or application environments, it provides a fundamental framework for adaptive recommendation systems for stock investment.

4.2. Procedure for rule matching and recommendation

This section discusses the detailed procedure of rule matching and investment recommendation. The overall procedure consists of seven steps shown below.

[1] t is the time interval that does belong to neither the rule head nor the rule body, i.e., a kind of a *don't care* interval. So, stock prices in this interval are not considered in the process of formulating rules. In this paper, we introduce the notion of t in order to generalize our rules to be discovered. In Section 5, we presented the experimental results that examine the tendency of rules according to different values of t.

Step 1: Symbolization of a query pattern: To symbolize a query pattern QP$(=\langle q[0], q[1], \ldots, q[Len(QP)-1]\rangle)$, we use the same method as the one explained in Section 3.1. That is, we first transform a query pattern QP into a sequence of change ratio QP$'(=\langle q'[0], q'[1], \ldots, q'[Len(QP)-2]\rangle)$ and then transform QP$'$ into a symbol sequence QP$''(=\langle q''[0], q''[1], \ldots, q''[Len(QP)-2]\rangle)$ via categorization.

Step 2: Searching for matched rule head: We search a frequent pattern base for the rule head RH matched to QP$''$. For this step, we first look up the index table to find out the location of the B-tree corresponding to the length of QP$''$. We then search this B-tree for the leaf node matched to QP$''$. The leaf node has a pointer to the list of the occurrence positions ⟨SID, offset⟩ of the patterns matched to QP$''$. Here, SID is the identifier of a stock sequence within which a pattern matched to QP$''$ occurs and offset is the offset from the beginning of the stock sequence identified by SID.

Step 3: Retrieval of actual element values of rule heads and bodies: Using the list of the occurrence positions obtained in Step 2, we retrieve from a stock database the actual element values of the corresponding rule head RH and body RB. If there are n instances of the rule head matched to QP$''$, n pairs of $\langle RH_i, RB_i\rangle (0 \leqslant i < n)$ are retrieved in this step.

Step 4: Calculation of average increase ratio: Suppose that a user wants to employ the average increase ratio as a basis for recommending investment types. Then, for each pair of $\langle RH_i, RB_i\rangle (0 \leqslant i < n)$, we calculate its average increase ratio by comparing the end price of RH_i to the average price of RB_i.

Step 5: Calculation of confidence: For each pair of $\langle RH_i, RB_i\rangle$, we first compare its average increase ratio with [minHold, maxHold] to choose one from the three possible change patterns, 'INCREASE', 'DECREASE', and 'UNCHANGED'. More specifically, we choose 'DECREASE' when the average increase ratio is less than minHold, 'INCREASE' when it is larger than maxHold, and 'UNCHANGED' when it is between minHold and maxHold. We then compute how many pairs of $\langle RH_i, RB_i\rangle$ support 'DECREASE', 'INCREASE', and 'UNCHANGED', respectively. We finally compute the confidences of 'DECREASE', 'INCREASE', and 'UNCHANGED'. The confidence of 'DECREASE' is defined as the percentage of the pairs of $\langle RH_i, RB_i\rangle$ supporting 'DECREASE' over all pairs of $\langle RH_i, RB_i\rangle$. The confidences of 'INCREASE' and 'UNCHANGED' are defined analogously.

Step 6: Rule generation: We choose from 'INCREASE', 'DECREASE', and 'UNCHANGED' the one whose confidence is larger than or equal to minConfidence. We then generate a rule with the chosen one as a rule body. Again, note that minConfidence is required to be at least 50% to avoid more than one investment type from being recommended. We do not generate a rule when all three confidences are less than minConfidence. Such a case indicates that the patterns after t time interval from the rule head do not show consistent changing patterns.

Step 7: Recommendation of an investment type: We recommend an investment type in accordance with the rule generated in Step 6. That is, we recommend 'BUY' when the rule supports 'INCREASE', 'SELL' when it supports 'DECREASE', and 'HOLD' when it supports 'UNCHANGED'. We do not recommend anything when no rules are generated in Step 6. In addition to an investment type, we provide the support and confidence values of the underlying rule in order to show its reliability and usefulness.

Example Fig. 5 illustrates the proposed procedure for rule matching and investment recommendation. First, in Fig. 5a, we transform a query sequence QP = ⟨5000, 5100, 4900, 5400⟩ into a

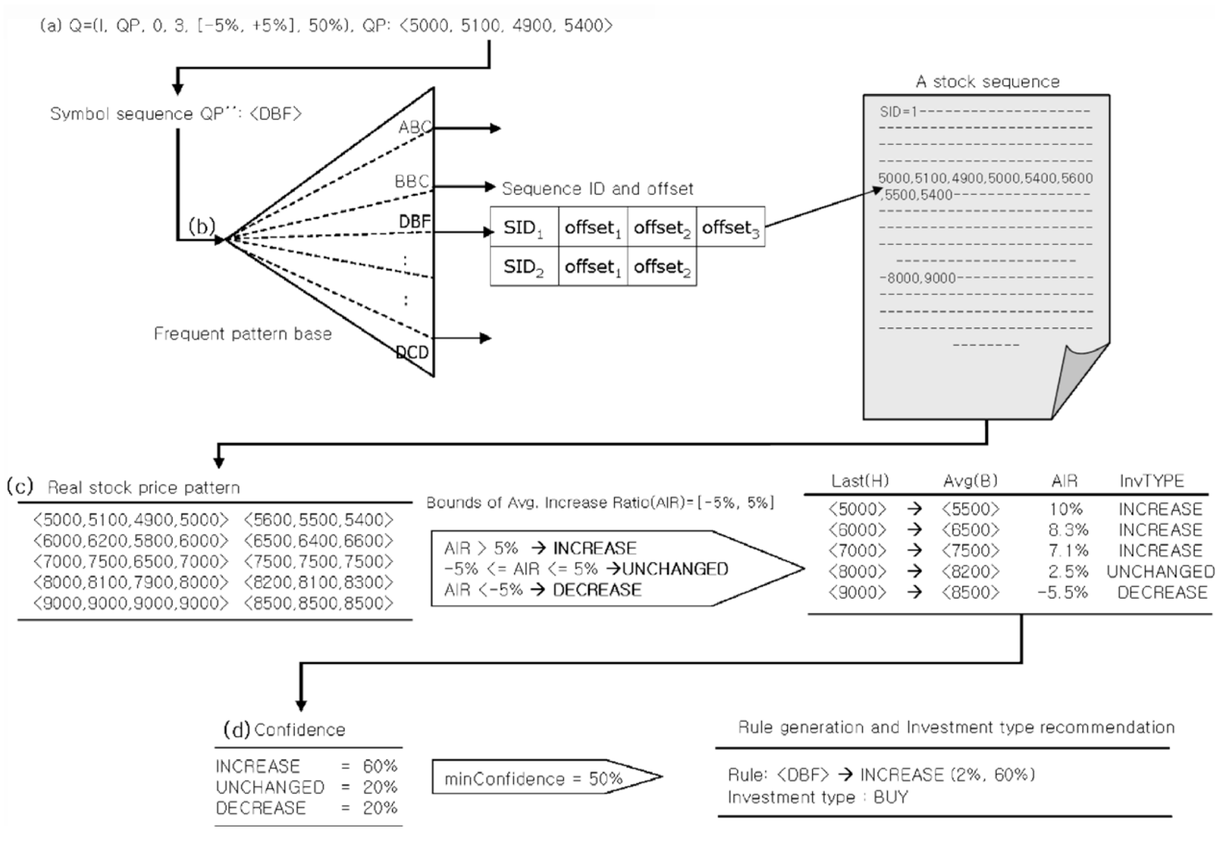
(a) Q=(I, QP, 0, 3, [-5%, +5%], 50%), QP: <5000, 5100, 4900, 5400>
Symbol sequence QP'': <DBF>
ABC
BBC
DBF
DCD
(b)
Frequent pattern base
Sequence ID and offset
SID1 offset1 offset2 offset3
SID2 offset1 offset2
A stock sequence
SID=1
5000,5100,4900,5000,5400,5600
,5500,5400
-8000,9000
(c) Real stock price pattern
<5000,5100,4900,5000> <5600,5500,5400>
<6000,6200,5800,6000> <6500,6400,6600>
<7000,7500,6500,7000> <7500,7500,7500>
<8000,8100,7900,8000> <8200,8100,8300>
<9000,9000,9000,9000> <8500,8500,8500>
Bounds of Avg. Increase Ratio(AIR)=[-5%, 5%]
AIR > 5% → INCREASE
-5% <= AIR <= 5% →UNCHANGED
AIR <-5% → DECREASE
Last(H) Avg(B) AIR InvTYPE
<5000> → <5500> 10% INCREASE
<6000> → <6500> 8.3% INCREASE
<7000> → <7500> 7.1% INCREASE
<8000> → <8200> 2.5% UNCHANGED
<9000> → <8500> -5.5% DECREASE
(d) Confidence
INCREASE = 60%
UNCHANGED = 20%
DECREASE = 20%
minConfidence = 50%
Rule generation and Investment type recommendation
Rule: <DBF> → INCREASE (2%, 60%)
Investment type : BUY

Fig. 5. An example of query processing.

sequence of change ratio, and then into a symbol sequence $QP'' = \langle DBF \rangle$ via categorization. Then, in Fig. 5b, we search a frequent pattern base for the rule head matched to QP'' and then retrieve the pairs of $\langle \text{SID}, \text{offset} \rangle$ which denote the occurrence positions of the patterns supporting the rule head matched. Next, using each pair of $\langle \text{SID}, \text{offset} \rangle$, we retrieve from the stock database the actual stock prices for the rule head and the corresponding rule body. Fig. 5c shows five such pairs $\langle RH_i, RB_i \rangle (0 \leqslant i < 5)$. For each pair of $\langle RH_i, RB_i \rangle$, we compute its average increase ratio and choose one from the three possible change patterns: 'INCREASE', 'DECREASE', and 'UNCHANGED'. In Fig. 5d, we compute the confidence of each of 'DECREASE', 'INCREASE', and 'UNCHANGED', and choose the one whose confidence is larger than or equal to minConfidnce specified in a query. In this example, the confidence of 'INCREASE' is larger than or equal to minConfidnce 50% and therefore the rule '$DBF \rightarrow$ INCREASE' is generated. Since this rule supports 'INCREASE', we recommend the investment type of 'BUY' as a final result.

5. Performance evaluation

This section shows the superiority of the proposed method by performance evaluation with extensive experiments. We first present the environment for experiments, and then analyze the results.

For performance evaluation, we performed a series of experiments on real-life stock data. We selected 10 blue-chip stock items included in the Korean Composite Stock Price Index (KOSPI) [13]. For each stock item, we collected a sequence of more than 5500 real numbers that represent the daily closing stock prices for 20 years from 28 May 1985 to 27 May 2005. In experiments, we call this data set $KOSPI-Data$. Each raw sequence was converted into a symbol sequence by categorization as described in Section 3.1.

The hardware platform was the Pentium IV 2.6 GHz PC equipped with 512 MB main memory and 80 GB hard disk of 7200 RPM. The software platform was Redhat Linux version 2.4.

In Experiment 1, we evaluated the proposed rule discovery and matching model. As a performance measure, we used the satisfaction ratio and the recommendation ratio. The *satisfaction ratio* indicates how much fraction of recommendations obtained by processing queries would satisfy stock investors, and is defined in the following.

satisfaction ratio = (number of recommendations satisfying the stock investors) ÷ (total number of meaningful recommendations obtained from processing queries).

We consider only 'BUY' and 'SELL' as meaningful recommendation because 'HOLD' and 'NO RECOMMENDATION' do not lead to investors' new actions. The expected increase ratio of a stock price is a decisive factor that affects investors to choose stock items. Normally, investors choose stock items predicted to get returns higher than what they want. So, the value of the average increase ratio used in stock item selections is likely to be higher than that for the investors' satisfaction. To consider such situations, we used the concept of a *satisfaction level*. The satisfaction level denoted as $sLevel$ indicates the percentage of the average increase ratio with which investors are actually satisfied.

In order to explain the meaning of $sLevel$ more clearly, we introduce the concept of the *target price of a stock* recommended by analysts. An analyst sets the target stock price based on the result of analyzing a company of interest. Stock investors buy (or sell) stocks by referring to these target stock prices of companies. Most stock investors tend to buy stocks whose target prices are set as high as possible, however, they (even their analysts) are not sure that the upcoming prices actually do reach their target ones. Even though the actual prices do not reach the target ones, the investors are satisfied with the resulting prices if they are somewhat close to the target ones. In this paper, we measured the satisfaction ratio by referring to this concept. In our case, *minHold* and *maxHold* correspond to target prices. Also, $sLevel$ represents the degree of the investor's expectancy for the target price.

In our experiments, the satisfaction ratio for a given query is computed as follows. First, we search for the rules from a frequent pattern base by using $[minHold, maxHold]$ as a range for the average increase ratio, and obtain their corresponding investment types. Then, we compute the satisfaction ratio for recommendations by using $[minHold \times sLevel, maxHold \times sLevel]$. In case of recommendation type 'BUY', investors are satisfied with their returns when their actual increase ratio after buying is greater than $sLevel\%$ of *maxHold*, which was specified in stock item selection. In case of recommendation type 'SELL', investors are satisfied when their actual decrease ratio after selling is greater than $sLevel\%$ of $|minHold|$.

The *recommendation ratio* indicates how much fraction of users' queries produce meaningful recommendations as their results, and is defined as follows.

recommendation ratio = (number of queries that produce meaningful recommendations) ÷ (total number of queries issued by investors)

Both satisfaction ratio and recommendation ratio are affected by various parameters used in our rule discovery and matching model. They are the number of symbols defined in categorization, the minimum support used in finding frequent patterns, the starting position of rule bodies used in rule discovery, the length of rule bodies, the range of the average increase ratio in the rule body, and the minimum confidence used in rule discovery.

70% of $KOSPI-Data$ was used for constructing a frequent pattern base, and 30% of $KOSPI-Data$ was used for forming query sequences. The minimum support for a frequent pattern base was set to 2%. As meaningful investment types, 'SELL' and 'BUY' were used. Also, the basic values of parameters in all the following experiments were set as follows: the number of symbols of 3, the minimum support of 2%, the range of the average increase ratio of [−2%, 2%], the time interval between the rule head and body of 0, the rule body length of 1, the minimum confidence of 60%, and the satisfaction level of 60%.

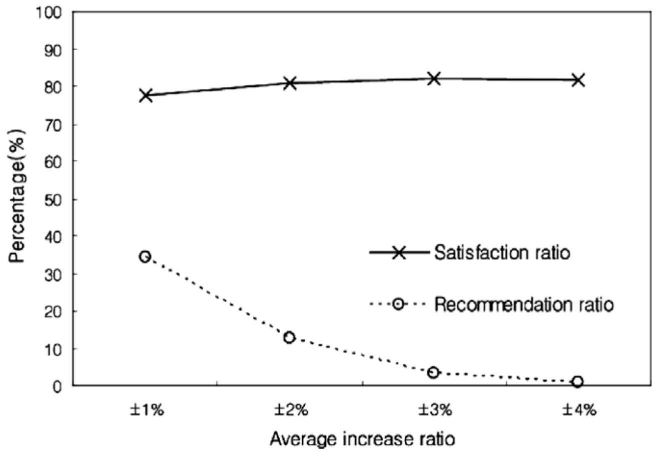

Fig. 6. Satisfaction ratio and recommendation ratio with different ranges of average increase ratio.

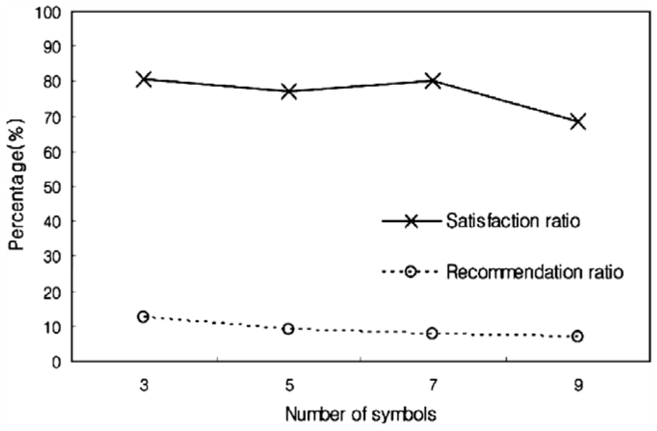

Fig. 7. Satisfaction ratio and recommendation ratio with different numbers of symbols.

Fig. 6 shows the tendency of satisfaction ratio and recommendation ratio with different average increase ratios in the rule body. We observe that satisfaction ratio is nearly constant around 80% even though the range of the average increase ratio changes. The recommendation ratio, however, decreases abruptly as the average increase ratio grows. In particular, when the range of the average increase ratio is [−3%,3%], the recommendation ratio was shown to be only 3.3%.

High recommendation ratio is not always beneficial to investors. However, a low recommendation close to 0 implies that the system is not that useful since it hardly recommends investment types. In this work, we aim at developing a stock investment system that provides reasonable recommendation ratio and high satisfaction ratio. From the results, we observe that our system recommends meaningful investment types for 13% of queries, and 80% of recommendations are satisfied by investors when the range of the average increase ratio was set to [−2%,2%].

Fig. 7 shows the tendency of satisfaction ratio and recommendation ratio with different numbers of symbols used in categorization. When the number of symbols is 3, 5, 7, and 9, satisfaction ratio is shown to be about 70–80% thereby not being influenced that much by the number of symbols. On the other hand, recommendation ratio is about 13% when the number of symbols is 3. It decreases slightly as the number of symbols increases. According to these results, the best number of symbols is 3 that shows satisfaction ratio of about 80% and recommendation ratio of 13%.

Fig. 8 shows the tendency of satisfaction ratio and recommendation ratio with different time intervals t between the rule head and rule body. As the time interval increases, both satisfaction ratio and recommendation ratio decrease. This result can be easily explained from the fact that if the time interval between the rule head and rule body increases, then the association between them diminishes. The results show that the best satisfaction ratio and recommendation ratio are obtained when t is 0.

Fig. 9 shows the tendency of satisfaction ratio and recommendation ratio according to the length of a rule body. The result reveals that both satisfaction and recommendation ratios get smaller as the length of a rule body increases. This is because, as a rule body increases, the rear values in the rule body have small associations with the rule head. The result shows that the best satisfaction and recommendation ratios are obtained when the length of a rule body is 1.

Fig. 10 reveals the tendency of satisfaction ratio and recommendation ratio with different minimum supports. We see that satisfaction ratio gradually decreases as the minimum support grows. On the other hand, recommendation ratio moves up and down locally, but tends to decrease gradually in a global point of a view as the minimum support increases. If the minimum support to find frequent patterns increases, then the frequent pattern base gets smaller, which makes satisfaction ratio and recommendation ratio become lower.

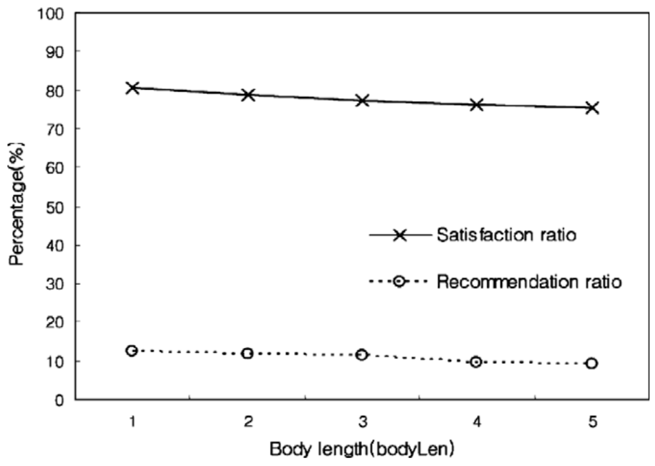

Fig. 9. Satisfaction ratio and recommendation ratio with different lengths of rule bodies.

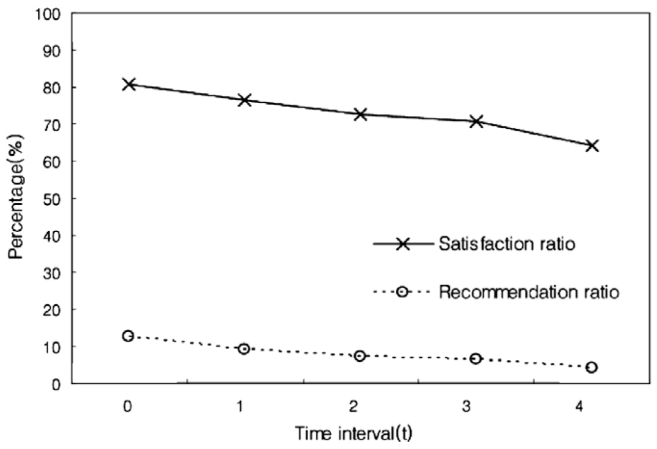

Fig. 8. Satisfaction ratio and recommendation ratio with different time intervals t.

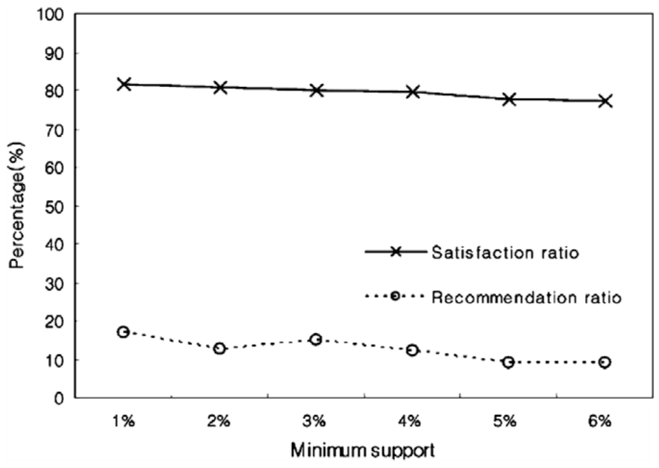

Fig. 10. Satisfaction ratio and recommendation ratio with different minimum supports.

Finally, Fig. 11 shows the tendency of satisfaction ratio with different satisfaction levels of users. If a user satisfaction level is set to 100%, the recommendations satisfy users only when their actual increase ratio belongs to the range of average increase ratio given at querying time. On the other hand, when the user satisfaction level decreases, the range of the average increase ratio used in verification also gets smaller. From the result, satisfaction ratio is inversely proportional to a user satisfaction level, but its maximum difference was only about 7% with a user satisfaction level of 100–20%. Moreover, satisfaction ratio in our all experiments appears to be larger than 70%, thereby showing the high accuracy of the proposed approach.

In Experiment 2, we analyzed the time for creating a frequent pattern base and the time for query processing in order to evaluate the efficiency of the proposed approach by using *KOSPI* – *Data*. To construct a frequent pattern base, we used the minimum support of 2%. Basic settings for parameters are as follows: the number of symbols of 3, the minimum support of 2%, the range of average increase ratio of +2%, the time interval *t* of 0, the length of a rule body of 1, and the minimum confidence of 60%.

Fig. 12 shows the time for constructing a frequent pattern base with a changing minimum support. We see that the time gets reduced as the minimum support gets larger. This is because a smaller number of frequent patterns are discovered with a larger minimum support.

Fig. 13 shows the tendency of the query processing time with a changing length of query sequences. The query processing time decreases as the length of a query sequence gets larger. This is explained as follows. Long patterns appear less frequently than short patterns. Also, in comparisons with a short one, a long frequent pattern is supported by a smaller number of actual patterns. Therefore, in case of a long query sequence, we need to process a small number of its supporting patterns, thus can enjoy a small processing time.

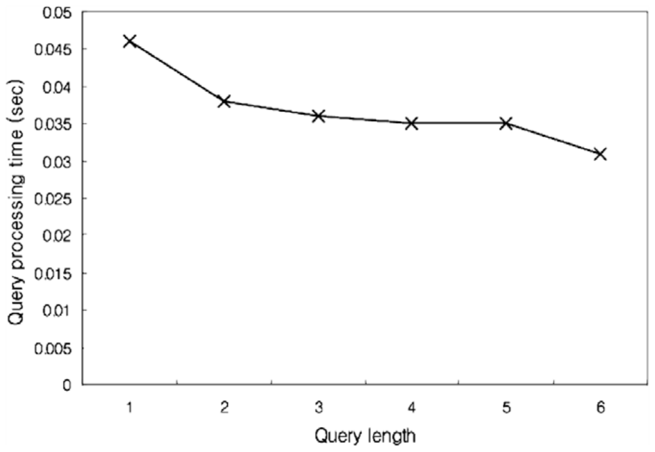

Fig. 13. Query processing time with different lengths of query sequences.

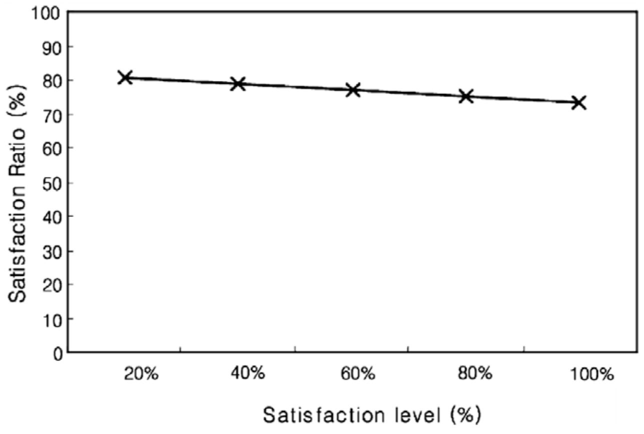

Fig. 11. Satisfaction ratio with different satisfaction levels.

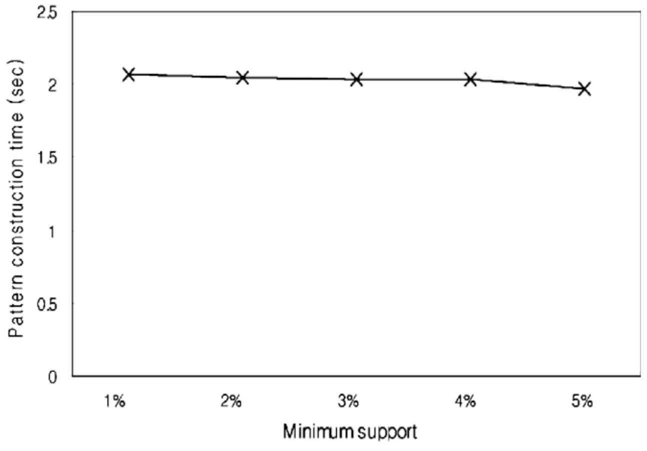

Fig. 12. Time for constructing a frequent pattern base with different minimum supports.

Fig. 14 shows the tendency of the time for constructing a frequent pattern base and the time for query processing with a changing data size. To generate large data sets, we duplicated *KOSPI* – *Data* 2, 3, and 4 times. In a horizontal axis, 1*S* denotes pure *KOSPI* – *Data*, and 2*S*, 3*S*, and 4*S* do its duplicated versions. Owing to this duplication, satisfaction ratio is entirely preserved when the same query of the minimum support and the minimum confidence is performed with different sizes of data sets. The result shows that the time for building a frequent pattern base and the time for query processing both increase in proportion to a data size. This is because the occurrences of a frequent pattern double when a data size gets twice as large as before. Consequently, the time for constructing a frequent pattern base and the time for query processing get doubled.

6. Conclusions and future study

In this paper, we aimed at devising an efficient and flexible approach that recommends appropriate investment types to stock investors by discovering useful rules from past changing patterns of stock prices stored in a database. To achieve our goal, we first

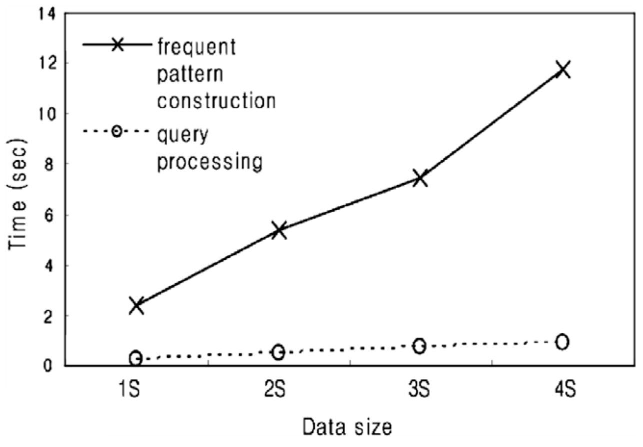

Fig. 14. Time for constructing a frequent pattern base and time for query processing with different data sizes.

proposed a new rule model for recommending stock investment types. For a frequent pattern of stock prices, if its subsequent stock prices are matched to a condition of an investor, the model recommends a corresponding investment type for this stock.

Based on the observation that the conditions on rule bodies are quite different depending on dispositions of investors while rule heads are nearly independent of characteristics of investors in most cases, we proposed a new method that discovers and stores only the rule heads rather than the whole rules in a rule discovery process. This allows investors to flexibly define various conditions on rule bodies, and also improves the performance of a rule discovery process by reducing the number of rules to be discovered. For efficient discovery and matching of rules, we proposed methods for discovering frequent patterns and organizing them into a frequent pattern base. We also suggested a method that searches a frequent pattern base for the rules matched to the conditions given by an investor, and a method that analyzes the matched rules to recommend an investment type.

To verify the superiority of the proposed approach, we performed various experiments through which we measured satisfaction ratios and response times of rule matching. Experimental results revealed that satisfaction ratios were between 70% and 80% in most cases and response times were less than a second.

As future work, we consider including diverse information related to stocks such as trade amount, the subjects of buying or selling, the frequency of appearances in the news during our analysis. Also, we are currently developing an effective method that employs continuous query processing for rapidly arriving stock data streams in our system.

Acknowledgements

This research was supported by the Korea Science and Engineering Foundation (KOSEF) Grant (No. R01-2008-000-20872-0) funded by the Korea government (MEST) and also was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program (No. IITA-2009-C1090-0902-0040) supervised by the IITA (Institute for Information Technology Advancement).

References

[1] R. Agrawal, C. Faloutsos, A. Swami, Efficient similarity search in sequence databases, in: Proceedings of the International Conference on Foundations of Data Organization and Algorithms (FODO), October 1993, pp. 69–84.
[2] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of the International Conference on Very Large Data Bases, VLDB, 1994, pp. 487–499.
[3] R. Agrawal, K. Lin, H.S. Sawhney, K. Shim, Fast similarity search in the presence of noise, scaling, and translation in time-series databases, in: Proceedings of the International Conference on Very Large Data Bases, VLDB, September 1995, pp. 490–501.
[4] R. Agrawal, R. Srikant, Mining sequential patterns, in: Proceedings of the International Conference on Data Engineering, 1995, pp. 3–14.
[5] T. Anderson, The Statistical Analysis of Time Series, Wiley, 1971.
[6] P. Bloomfield, Fourier Analysis of Time Series, Wiley, 2000.
[7] W.W. Chu, K. Chiang, Abstraction of high level concepts from numerical values in databases, in: Proceedings of the AAAI Workshop on Knowledge Discovery in Databases, 1994, pp. 133–144.
[8] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, P. Smyth, Rule discovery from time series, in: Proceedings of the International Conference on Knowledge Discovery and DataMining, 1998, pp. 16–22.
[9] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, Fast subsequence matching in time-series databases, in: Proceedings of the International Conference on Management of Data, ACM SIGMOD, May 1994, pp. 419–429.
[10] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large databases, Information Systems 26 (1) (2001) 35–58.
[11] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, 2001.
[12] S.W. Kim, S.H. Park, W.W. Chu, An index-based approach for similarity search supporting time warping in large sequence databases, in: Proceedings of the International Conference on Data Engineering, IEEE ICDE, 2001, pp. 607–614.
[13] Koscom Data Mall, <http://datamall.koscom.co.kr>, 2005.
[14] W.K. Loh, S.W. Kim, K.Y. Whang, A subsequence matching algorithm that supports normalization transform in time-series databases, Data Mining and Knowledge Discovery Journal 9 (1) (2004) 5–28.
[15] S.H. Park, W.W. Chu, J. Yoon, C. Hsu, Efficient searches for similar subsequences of difference lengths in sequence databases, in: Proceedings of the International Conference on Data Engineering, IEEE ICDE, 2000, pp. 23–32.
[16] S. Park, W.W. Chu, Discovering and matching elastic rules from sequence databases, Fundamenta Informaticae 47 (1–2) (2001) 75–90.
[17] E. Saad, D. Prokhorov, D. Wunsch II, Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks, IEEE Transactions on Neural Networks (1998) 1456–1470.
[18] B. Wah, M. Qian, Constrained formulations and algorithms for stock-price predictions using recurrent FIR neural networks, AAAI/IAAI (2002) 211–216.
[19] H. White, Economic prediction using neural networks: the case of IBM daily stock returns, in: Proceedings of the IEEE International Conference on Neural Networks, 1988, pp. 451–458.
[20] T. Fu, T. Cheung, F. Chung, C. Ng, An innovative use of historical data for neural network based stock prediction, in: Proceedings of the International Conference on Information Sciences, JCIS, October 2006.
[21] Y. Kwon, S. Choi, B. Moon, Stock prediction based on financial correlation, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, June 2005, pp. 2061–2066.
[22] V. Sehgal, C. Song, SOPS: stock prediction using web sentiment, in: Proceedings of the International Conference on Data Mining Workshop, IEEE ICDMW, October 2007, pp. 21–26.