

CLUSTERING ITEMS IN MULTIPLE DATABASES INDUCED BY STABILITY

Jhimli Adhikari

Department of Computer Science, Narayan Zantye
College, Bicholim, Goa – 403 529, India
jhimli_adhikari@yahoo.co.in

P R Rao

Department of Computer Science and Tecnology, Goa
University, Goa - 403 206, India
pralhaad@rediffmail.com

ABSTRACT

Many multi-branch companies transact from different branches. Each branch of such a company maintains a separate database over time. Little work has been reported on multiple transactional time-stamped databases. Thus, it is necessary to study data mining on multiple transactional time-stamped databases. In this paper, we introduce the notion of stability of an item. Stable items are useful in making many strategic decisions for the company. We propose the notion of degree of stability of an item. Based on the degree of stability of an item, we design an algorithm for clustering items in multiple databases. Experimental results are provided on two transactional databases.

KEY WORDS

Clustering, data mining, dispersion, multiple databases, stability

1. Introduction

Due to liberalization of government policies across the globe, the number of multi-branch companies is increasing over time. Many multi-branch companies deal with multiple databases. Thus, the study of data mining on multiple databases is an important issue. Data mining and knowledge discovery on multiple databases has been recently recognized [1], [2] as an important area of research in data mining.

Many of these multi-branch companies also deal with transactional time-stamped data. Transactional data collected over time at no particular frequency is called *transactional time-stamped data* [3]. Some examples of transactional time-stamped data are point of sales data, inventory data, and trading data. Little work has been reported on the area of mining multiple transactional time-stamped databases. Many important and useful applications might involve transactional time-stamped data. In the next section, we mention a few examples of such applications.

All the transactions in a branch are stored locally. A transaction could be viewed as a collection of items with a unique identifier. An interesting characteristic of an item is its variation of sales over time. The items having less variation of sales over time are useful in devising strategies for a company. Thus, it is important to study such items.

Rest of the paper is organized as follows. In Section 2, we propose a model of mining multiple transactional time-stamped databases. We state our problem in Section 3. In Section 4, we design an algorithm for clustering of items in multiple databases. We discuss

related work in Section 5. Experimental results are provided in Section 6.

2. Mining multiple transactional time-stamped databases

Consider a multi-branch company that has n branches. Let D_i be the transactional time-stamped database corresponding to i -th branch, for $i = 1, 2, \dots, n$. Web sites and transactional databases contain a large amount of time-stamped data related to an organization's suppliers and / or customers over time. Mining these types of time-stamped data could help business leaders make better decisions by listening to their suppliers or customers via their transactions collected over time [3]. We propose a model of mining global patterns in multi-databases over time.

The proposed model has a set of interfaces and a set of layers. Each interface is a set of operations that produces dataset(s) (or, knowledge) based on the lower layer dataset(s). There are five distinct interfaces of the proposed model of synthesizing global patterns from local patterns. The function of each interface is described below. Interface 2/1 cleans / transforms / integrates / reduces data at the lowest layer. By applying these procedures we get processed database from the original database. In addition, interface 2/1 applies a filtering algorithm on each database for separating relevant data from outlier data. E.g., if we are interested in studying the durable items then the transactions containing only non-durable items could be treated as outlier transactions. Also, it loads data into the respective data warehouse. At interface 3/2, each processed database PD_i is partitioned into k time databases DT_{ij} , where DT_{ij} is the processed database (if available) for the j -th time slot at the i -th branch, for $j = 1, 2, \dots, k$, and $i = 1, 2, \dots, n$. The j -th time databases of all branches are merged into a single time database DT_j , for $j = 1, 2, \dots, k$. A traditional data mining technique could be applied on database DT_j at the interface 5/4, for $j = 1, 2, \dots, k$. Let PB_j be pattern base corresponding to the time database DT_j , for $j = 1, 2, \dots, k$. Finally, all the pattern bases are processed for synthesizing knowledge or, making decision at the interface 6/5. Undirected lines in Figure 1 are assumed to be directed from bottom to top. The proposed model of mining global patterns over time is efficient, since we get the exact global patterns in multiple databases over time.

In layer 4, we have collection of all time databases. If any one of these databases is too large to apply a traditional data mining technique then this data mining model would fail. In this situation, we could apply an

appropriate sampling technique to reduce the size of a database. Thus, we get approximate patterns over time.

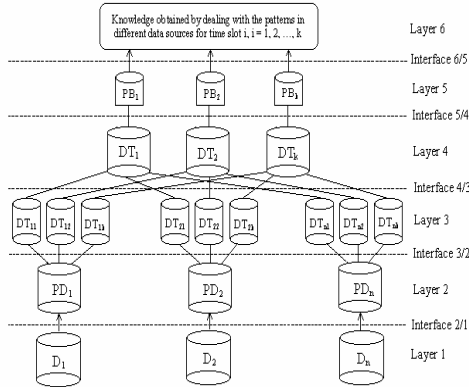


Figure 1. A model of mining global patterns in multiple transactional time-stamped databases

3. Problem statement

With reference to Figure 1, let DT_j be the database corresponding to the j -th year, for $j = 1, 2, \dots, k$. Each of these databases corresponds to a specific period of time. Thus we could call them as time databases. Each of these time databases is mined using a traditional data mining technique [4], [5]. For the specific requirement of this problem, we need to mine only items in time databases. Let I be the set of all items. Each itemset X in a database D is associated with a statistical measure, called support [6], denoted by $\text{supp}(X, D)$. The *support* of an itemset is defined as the fraction of transactions containing the itemset. The variation of sales of an item over the time is an important issue in determining stability of the item. Stable items are useful in many applications, eg. stable items could be useful to promote sales of other items. Modeling with stable item is more justified than modeling with unstable item.

Let $\mu_{s(x)}(t)$ be the mean support of item x in the database DT_1, DT_2, \dots, DT_t . Thus, $\mu_{s(x)}(t)$ is obtained by the following formula:

$$\mu_{s(x)}(t) = \left(\sum_{i=1}^t \text{supp}(x, DT_i) \times \text{size}(DT_i) \right) / \sum_{i=1}^t \text{size}(DT_i), \quad \text{for } t = 1, 2, \dots, k \quad \dots (1)$$

Let $\sigma(\mu_{s(x)})$ be the standard deviation of $\mu_{s(x)}(t)$, for $t = 1, 2, \dots, k$. We call $\sigma(\mu_{s(x)})$ as the *variation of means* corresponding to support of x . Let $\gamma_{s(x)}(t, t+h)$ be the autocovariance of $\text{supp}(x, DT_t)$ at lag h , for $t = 1, 2, \dots, k-1$. Thus, $\gamma_{s(x)}(t, t+h)$ is obtained by the following formula: $\gamma_{s(x)}(t, t+h) =$

$$\frac{1}{k} \sum_{t=1}^{k-h} \left(\text{supp}(x, DT_t) - \mu_{s(x)}(k) \right) \left(\text{supp}(x, DT_{t+h}) - \mu_{s(x)}(k) \right) \quad \dots (2)$$

$\sigma(\gamma_{s(x)}(t, t+h))$ be the standard deviation of $\gamma_{s(x)}(t, t+h)$, for $h = 1, 2, \dots, k-1$. We call this $\sigma(\gamma_{s(x)}(t, t+h))$ as *variation of autocovariances* corresponding to support of x . We have chosen standard deviation as a measure of dispersion [7]. Standard deviation and mean deviation about mean are relevant measures of dispersion. These measures take into account of variation due to each support unlike the measure range. Skewness, being a descriptive measure of dispersion is

not suitable in this context. Before we define stability of an item, we study the following time series of supports corresponding to an item. In the following example, we consider a few sample time series of supports corresponding to different items.

Example 1. Let i -th series be the time series of supports corresponding to item x_i , for $i = 1, 2, 3, 4, 5$.

- (1) .03, .20, .31, .11, .07, .35, .82, .62, .44, .13
- (2) .19, .20, .18, .21, .20, .20, .19, .18, .21, .20
- (3) .05, .11, .07, .20, .16, .12, .13, .08, .17, .10
- (4) .03, .04, .03, .07, .08, .12, .09, .15, .17, .12
- (5) .04, .04, .03, .05, .04, .06, .04, .05, .06, .05

The variations of means and autocovariances of above series are given as follows: (1) $\sigma(\mu) = 0.09342$, $\sigma(\gamma) = 0.01234$, (2) $\sigma(\mu) = 0.00230$, $\sigma(\gamma) = 0.00002$, (3) $\sigma(\mu) = 0.02351$, $\sigma(\gamma) = 0.00039$, (4) $\sigma(\mu) = 0.02114$, $\sigma(\gamma) = 0.00076$, (5) $\sigma(\mu) = 0.0027986$, $\sigma(\gamma) = 0.0000124$. •

We define stable items based on the concept of stationary time series data [3], [8]. In finding $\sigma(\mu)$, we first compute a set of means of support values. Then we compute standard deviation of these mean values. Thus, we find standard deviation of a set of fractions. In finding $\sigma(\gamma)$, we first compute a set of autocovariances of support values. Then we compute standard deviation of these autocovariances. An autocovariance of supports is an average of a set of squared fractions. Thus, we find standard deviation of a set of squared fractions. So, $\sigma(\mu) \geq \sigma(\gamma)$. In fact, $\sigma(\gamma)$ is close to 0. Thus, we define our first measure of stability $stable_1$ as follows.

Definition 1. An item x is stable if $\sigma(\mu_{s(x)}) \leq \delta$, where δ is user defined *maximum threshold*. •

More strictly, we may wish to impose restrictions on both $\sigma(\mu)$ and $\sigma(\gamma)$. Thus we define our second measure of stability $stable_2$ as follows.

Definition 2. An item x is stable if $\sigma(\mu_{s(x)}) + \sigma(\gamma_{s(x)}) \leq \delta$, where δ is user defined maximum threshold. •

In definition 2, the expression $\sigma(\mu) + \sigma(\gamma)$ is the determining factor of stability of an item. We define $\sigma(\mu) + \sigma(\gamma)$ as *degree of variation (degOfVar)* of the item. Higher value of $\sigma(\mu) + \sigma(\gamma)$ implies lower degree of stability of the item. Based on above discussion, we state our problem as follows.

Let D_i and DT_j be the databases corresponding to i -th branch and j -th year of a multi-branch company as depicted in Figure 1 respectively, for $i = 1, 2, \dots, n$, and $j = 1, 2, \dots, k$. Each of the time (year) databases has been mined using a traditional data mining technique. Based on the mining results, degree of variation of each item has been computed as discussed above. Find the best non-trivial partition (if it exists) of the items in D_1, D_2, \dots, D_n based on degree of variation of an item.

A partition [9] is a specific type of clustering. Formal definition of non-trivial partition is given in Section 4.

4. Clustering items

The proposed clustering technique is based on the notion of degree of stability of an item. Again the degree of stability is based on the variations of means and autocovariances. The clustering technique requires computing the degree of variation for each item in the

databases. Let I be the set of items in the databases. Given a set of yearly databases, the difference in variations between every pair of items could be expressed by a square matrix called *difference in variation* ($diffInVar$). We construct $diffInVar$ as follows.

$$diffInVar(i, j) = |degOfVar(x_i) - degOfVar(x_j)|, \quad \dots (3)$$

for $x_i, x_j \in I$.
Example 2. We continue here the discussion of Example 1. Matrix $diffInVar$ is given as follows.

$$diffInVar = \begin{bmatrix} 0 & 0.103 & 0.082 & 0.084 & 0.102 \\ 0.103 & 0 & 0.021 & 0.019 & 0.001 \\ 0.082 & 0.021 & 0 & 0.002 & 0.020 \\ 0.084 & 0.019 & 0.002 & 0 & 0.018 \\ 0.102 & 0.001 & 0.020 & 0.018 & 0 \end{bmatrix} \bullet$$

Intuitively, if the difference in variations between two items is close to zero then they may be put in the same class. Before clustering the items, we define a class as follows.

Definition 3. Let $I = \{i_1, i_2, \dots, i_p\}$ be the set of items. A class formed at the level of difference in variation α is defined as follows.

$$class(I, \alpha) = \begin{cases} X: X \subseteq I, |X| \geq 2, \text{ and } degOfVar(x_1, x_2) \leq \alpha, \text{ for } x_1, x_2 \in X \\ X: X \subseteq I, |X| = 1 \end{cases} \bullet$$

Based on the above definition of a class, we define a clustering as follows.

Definition 4. Let $I = \{i_1, i_2, \dots, i_p\}$ be the set of items. Let $\pi(I, \alpha)$ be a clustering of items in I at the level of difference in variation α . Then, $\pi(I, \alpha) = \{X: X \in \rho(I), \text{ and } X \text{ is a } class(I, \alpha)\}$, where $\rho(I)$ is the power set of I .

During the clustering process we may like to impose the restriction that each item belongs to at least one class. This restriction makes a clustering complete. We define a complete clustering as follows.

Definition 5. Let $I = \{i_1, i_2, \dots, i_p\}$ be the set of items. Let $\pi(I, \alpha) = \{C_1(I, \alpha), C_2(I, \alpha), \dots, C_m(I, \alpha)\}$, where $C_k(I, \alpha)$ is the k -th class of the cluster π , for $k = 1, 2, \dots, m$. π is complete, if $\cup_{k=1}^m C_k(I, \alpha) = I$.

In a complete clustering, two classes may have common items. We may be interested in finding out a cluster containing mutually exclusive classes. A mutually exclusive cluster could be defined as follows.

Definition 6. Let $I = \{i_1, i_2, \dots, i_p\}$ be the set of items. Let $\pi(I, \alpha) = \{C_1(I, \alpha), C_2(I, \alpha), \dots, C_m(I, \alpha)\}$, where $C_k(I, \alpha)$ is the k -th class of the cluster π , for $k = 1, 2, \dots, m$. π is mutually exclusive if $C_i(I, \alpha) \cap C_j(I, \alpha) = \emptyset$, for $i \neq j$, and $1 \leq i, j \leq m$.

We may be interested in finding out such a mutually exclusive and complete cluster. A partition of a set of items I is defined as follows.

Definition 7. Let $\pi(I, \alpha)$ be a mutually exclusive and complete cluster of a set of items I at the level of difference in variation α . $\pi(I, \alpha)$ is called a non-trivial partition if $1 < |\pi| < m$.

A partition is a cluster. But a cluster is not necessarily be a partition. In the next section, we find the best non-trivial partition (if it exists) of a set of items.

The items in a class are similar with respect to their variations. We are interested in the classes of a partition where the variations of items are less. The items in

these classes are useful in devising strategies for the company. Thus, we define average degree of variation adv of a class as follows.

Definition 8. Let C be a class of partition π .

$$adv(C | \pi) = \frac{1}{|C|} \sum_{x \in C} degOfVar(x) \bullet$$

4.1. Finding the best non-trivial partition

To find all the non-trivial partitions, we arrange all non-zero and distinct values of $diffInVar$ in non-decreasing order, for $1 \leq i < j \leq 5$. The arranged values of $diffInVar$ are 0.001, 0.002, 0.018, 0.019, 0.020, 0.021, 0.082, 0.084, 0.102, 0.103. We get two non-trivial partitions at $\alpha = 0.001$, and $\alpha = 0.002$. The partitions are given as follows: $\pi^{0.001} = \{\{x_1\}, \{x_2, x_5\}, \{x_3\}, \{x_4\}\}$, and $\pi^{0.002} = \{\{x_1\}, \{x_2, x_5\}, \{x_3, x_4\}\}$. We observe that at different level of α we have different partitions. We would like to find the best partition among these partitions. The best partition is based on the principle of minimizing the intra-class variation and minimizing the inter-class similarity. Intra-class variation and inter-class similarity are defined as follows.

Definition 9. The intra-class variation $intra-var$ of a partition π at the level α is defined as follows.

$$intra-var(\pi^\alpha) = \sum_{k=1}^{|\pi|} \sum_{\substack{x_i, x_j \in C_k \\ x_i < x_j}} |degOfVar(x_i) - degOfVar(x_j)| \bullet$$

Definition 10. The inter-class similarity $inter-sim$ of a partition π at the level α is defined as follows.

$$inter-sim(\pi^\alpha) = \sum_{C_p, C_q \in \pi; p < q} \sum_{x_i \in C_p, x_j \in C_q} \text{minimum}\{degOfVar(x_i), degOfVar(x_j)\} \bullet$$

The best partition among a set of partitions is selected on the basis of goodness value of a partition. Goodness measure $goodness$ of a partition is defined as follows.

Definition 11. The goodness of a partition π at level α is defined as follows. $goodness(\pi^\alpha) = intra-var(\pi^\alpha) + inter-sim(\pi^\alpha) - |\pi|^\alpha$, where $|\pi|^\alpha$ is the number of classes of π .

We have subtracted $|\pi|^\alpha$ from the sum of intra-class variation and inter-class similarity to remove the bias of goodness value for a partition. Better partition is obtained at higher goodness value. We would like to partition the set of items using above goodness measure.

Example 3. With reference to Example 2, we calculate goodness value of each of the non-trivial partitions.

$$intra-var(\pi^{0.001}) = 0.001, inter-sim(\pi^{0.001}) = 0.081, \text{ and } |\pi^{0.001}| = 4. \text{ Thus, } goodness(\pi^{0.001}) = -3.916.$$

$$intra-var(\pi^{0.002}) = 0.003, inter-sim(\pi^{0.002}) = 0.06, \text{ and } |\pi^{0.002}| = 3. \text{ Thus, } goodness(\pi^{0.002}) = -2.937.$$

The goodness value corresponding to the partition $\pi^{0.002}$ is the maximum. Thus, the partition $\pi^{0.002}$ is the best among the non-trivial partitions. Let us return back to Example 1. There are five series of supports corresponding to five items. Based on variation among the supports in a series, we could partition the series as follows: {series 1}, {series 2, series 5}, {series 3, series 4}. Hence, we get the following partition: $\{x_1\}, \{x_2, x_5\}, \{x_3, x_4\}$. The proposed clustering technique also identifies the same partition as the best partition. Thus, it verifies the correctness of the proposed clustering

technique. $adv(\{x_1\} | \pi^{0.002}) = 0.105$, $adv(\{x_2, x_5\} | \pi^{0.002}) = 0.0025$ and $adv(\{x_3, x_4\} | \pi^{0.002}) = 0.022$. We find that the average degree of variation of $\{x_2, x_5\}$ is the least among the classes of $\pi^{0.002}$. Thus, the items x_2 and x_5 are most suitable among all the items in the databases for making strategies of the company. •

We design an algorithm for finding best non-trivial partition of items in multiple databases. First we describe different data structures used in designing an algorithm for finding the best partition of items. For each item there are k supports corresponding to k different years. We maintain $m \times k$ supports for m items in array *supports*. The i -th row of *supports* stores supports corresponding to i -th item for k years, for $i = 1, 2, \dots, m$. Let *means* be a two dimensional array such that the i -th row stores means of supports corresponding to different years for i -th item, for $i = 1, 2, \dots, m$. Let *autocovariances* be a two dimensional array such that the i -th row stores autocovariances of supports corresponding to different lags for i -th item, for $i = 1, 2, \dots, m$. For year j , we compute mean value of supports for year 1 to j . Thus we get different mean values for different years. Let *stdDevMeans* be the standard deviation of these mean values. For year j , we also compute autocovariances of supports for year 1 to j at different lags. Thus we get different autocovariances for different lags corresponding to a year. Let *stdDevAutocovars* be the standard deviation of these autocovariances. The degrees of variation of different items are stored in array *degInVar*. Variable S is a one dimensional array containing ${}^m C_2$ variations. *adv* is a one dimensional array which stores the average degree of variation for the items in each class. The algorithm is presented below.

Algorithm 1. Best non-trivial partition (if it exists) of items in multiple databases.

procedure *BestPartition* (m , supports)

Inputs:

m : number of items

supports: array of supports of different items corresponding to different years

Outputs:

Best non-trivial partition (if it exists) of items in multiple databases

```

01: for  $i = 1$  to  $m$  do
02:   compute means( $i$ ) using formula (1) at different
      years;
03:   let stdDevMeans = standard deviation of mean
      values for different years;
04:   compute autocovariance( $i$ ) using formula (2) at
      different time lags;
05:   let stdDevAutocovars = standard deviation of
      autocovariances;
06:   compute degOfVar( $i$ ) = stdDevMeans +
      stdDevAutocovar;
07: end for
08: for row = 1 to  $m$  do
09:   for col = (row + 1) to  $m$  do
10:     compute diffInVar(row, col) using formula (3);
11:   end for
12: end for

```

```

13: sort distinct elements in the upper triangle of
      diffInVar in non-decreasing order into array  $S$ ;
14: let  $k = 1$ ; let maxGoodness = -9999;  $\pi = \emptyset$ ;
15: while ( $k \leq |S|$ ) do
16:   let curRow = 1; let curClass = 1;
17:   for  $i = 2$  to  $m$  do
18:     classLabel( $i$ ) = 0;
19:   end for
20:   let classLabel(1) = 1;
21:   let curDiffVar =  $S(k)$ ;
22:   for col = curRow + 1 to  $m$  do
23:     if (diffInVar(curRow, col)  $\leq$  curDiffVar) then
24:       if (classLabel(col) = 0) then
25:         classLabel(col) = curClass;
26:       else if (classLabel(col)  $\neq$  curClass) then
27:         partition does not exist at this level;
28:       go to line 49;
29:     end if
30:   end if
31:   end for
32:   increased curRow by 1;
33:   if (classLabel(curRow) = 0) then
34:     increased curClass by 1;
35:   classLabel(curRow) = curClass;
36:   else curclass = classLabel(curRow);
37:   end if
38:   if (curRow  $\leq m$ ) go to line 22; end if
39:   let  $j = 0$ ;
40:   while ((classLabel( $j$ )  $\neq$  0) and ( $j < m$ )) do
41:     increase  $j$  by 1;
42:   end while
43:   if ( $j = m + 1$ ) then
44:     if (maxGoodness < goodness value of current
          partition) then
45:       maxGoodness = goodness value of current
          partition;
46:       store current partition into  $\pi$ ;
47:     end if
48:   end if
49:   increase  $k$  by 1;
50: end while
51: return  $\pi$ ;
52: end procedure

```

Line 2 takes $O(k)$ time to compute means(i), for some $i = 1, 2, \dots, m$. Also, line 3 takes $O(k)$ time to compute standard deviation of mean values. To compute formula (2), we require $O(k)$ time. Thus, line 4 takes $O(k^2)$ time. In line 5, we compute standard deviation of $k-1$ autocovariance values. Thus, line 5 takes $O(k^2)$ time. The for-loop in lines 1-7 repeat m times. Thus, the for-loop in lines 1-7 take $O(m \times k^2)$ time. For computing *diffInVar* at a given row and column, it takes $O(1)$ time. Thus, lines 8-12 take $O(m^2)$ time. There are maximum ${}^{m-1} C_2$ elements in the upper triangle of *diffInVar*. Thus, line 13 takes $O(m^2 \times \log(m))$ time. The while-loop at line 15 repeats maximum ${}^{m-1} C_2$ times. Each of the loops at lines 17, 22, and 40 takes $O(m)$ time. To store a partition it takes $O(m)$ time. To compute goodness value for a particular partition, it takes $O(m^2)$ time. Thus, the lines 15-50 take $O(m^4)$ time. The time complexity of *bestPartition* algorithm is $O(m^4)$.

Table 2. Time database characteristics

DB	NT	ALT	AFI	NI	DB	NT	ALT	AFI	NI
M ₀	812	24.00000	295.27272	66	M ₅	812	24.00000	221.45454	88
M ₁	812	24.00000	286.58823	68	M ₆	812	24.00000	216.53333	90
M ₂	812	24.00000	249.84615	78	M ₇	812	24.00000	191.05882	102
M ₃	812	24.00000	282.43478	69	M ₈	812	24.00000	229.27058	85
M ₄	812	24.00000	259.84000	75	M ₉	816	24.00000	227.72093	86
E ₀	33	7.00000	4.62000	50	E ₅	33	7.00000	3.91525	59
E ₁	33	7.00000	5.13333	45	E ₆	33	7.00000	3.50000	66
E ₂	33	7.00000	5.50000	42	E ₇	33	7.00000	3.91525	59
E ₃	33	7.00000	4.81250	48	E ₈	33	7.00000	3.39706	68
E ₄	33	7.00000	3.39706	68	E ₉	39	7.00000	4.55000	60

5. Related Work

Liu et al. [10] have proposed stable association rules based on testing of hypothesis. In this case, the distribution of test statistic under null hypothesis is normal for large sample size. Thus, the stable association rules are determined based on some assumptions. Due to these reasons, we define stable items based on the concept of stationary time series data [8].

In the context of interestingness measures, Tan et al. [11] have described several key properties of twenty one interestingness measures proposed in statistics, machine learning and data mining literature. One should examine these properties in order to select right interestingness measure for a given application domain. Wu et al. [12] have proposed two similarity measures for clustering a set of databases.

Zhang et al. [13] have proposed an efficient and scalable data clustering method BIRCH based on a new in-memory data structure called CF-tree. Estivill-Castro and Yang [14] have proposed an algorithm that remains efficient, generally applicable, multi-dimensional but is more robust to noise and outliers. Jain et al. [15] have presented an overview of pattern clustering methods from a statistical pattern recognition perspective, with a goal of providing useful advice and references to fundamental concepts accessible to the broad community of clustering practitioners. In this paper, we cluster items in multiple databases based on supports of items. Thus, the above algorithms are not suitable under this framework. Yoon et al. [16] have proposed a bitmap approach to trend clustering using similarity as well as dissimilarity matches of trend sequences.

Yang and Shahabi [17] have proposed an algorithm to determine the stationarity of multivariate time series data for improving the efficiency of many correlation based data analysis.

Zhang et al. [18] designed a local pattern analysis for mining multiple databases. Zhang et al. [19] studied various issues related to multi-database mining.

6. Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have

been implemented on a 1.6 GHz Pentium IV with 256 MB of memory, using the software visual C++ (version 6.0). We present the experimental results using two real datasets *mushroom* [20], and *ecoli* [21]. Dataset *ecoli* is a subset of *ecoli database* and it has been processed for the purpose of conducting experiments.

Table 1. Dataset characteristics

Dataset	N T	ALT	AFI	NI
<i>mushroom</i>	8124	24.00000	1624.80000	120
<i>ecoli</i>	336	7.00000	25.83517	91

Let DB, NT, ALT, AFI, and NI denote database, the number of transactions, average length of a transaction, average frequency of an item, and number of items respectively. We present some characteristics of these datasets in Table 1. Each dataset has been divided into 10 databases, called input databases, for the purpose of conducting experiments. The input databases obtained from *mushroom* and *ecoli* are named as M_i, and E_i, for i = 0, 1, ..., 9. We present some characteristics of the input databases in Table 2. In Table 3, we present top stable items in multiple databases.

Table 3. Top 10 stable items in multiple databases

<i>mushroom</i>		<i>ecoli</i>	
item	degOfVar	item	degOfVar
85	0.00000	1	0.00129
8	0.00018	99	0.00181
12	0.00027	91	0.00225
75	0.00030	4	0.00252
89	0.00030	94	0.00363
62	0.00090	15	0.00385
22	0.00104	12	0.00389
20	0.00107	19	0.00402
82	0.00124	3	0.00404
33	0.00141	10	0.00443

The best partition of items in *mushroom* dataset is obtained at level 0.24760. The amount of intra variation, inter similarity, and goodness value are 413.59719, 377.58308, and 789.18027 respectively. The best partition contains two classes. The best class is given as follows: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,

75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, }. It has average degree of variation 0.05682.

The best partition of items in *ecoli* dataset is obtained at level 0.05632. The amount of intra variation, inter similarity, and goodness value are 44.17961, 185.60862, and 227.78823 respectively. The best partition contains two classes. The best class is given as follows: {0, 1, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 43, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 94, 99, 100}. It has average degree of variation 0.01829.

We have studied execution time with respect to number of data sources. We observe in Figures 1 and 2 that, the execution time increases as the number of data sources increases.

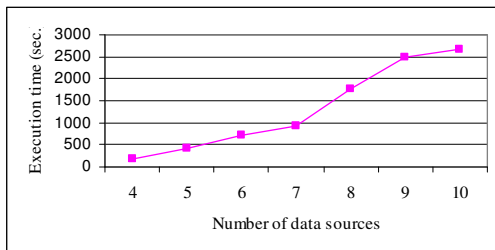


Figure 1. Execution time vs. number of data sources obtained from *mushroom*

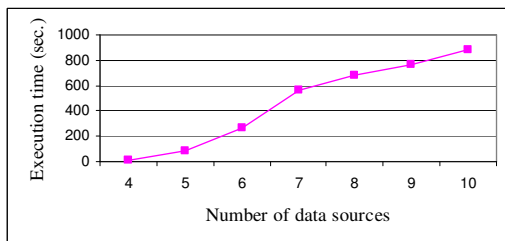


Figure 2. Execution time vs. number of data sources obtained from *ecoli*

7. Conclusions

Stable items are useful for modeling various strategies of an organization. Thus, it is necessary to identify stable items. We propose the notion of degree of stability of an item. We design an algorithm for clustering items in multiple databases based on degree of stability. The proposed technique is useful and effective.

References

[1] X. Wu, & S. Zhang, Synthesizing high-frequency rules from different data sources, *IEEE Transactions on Knowledge and Data Engineering*, 14(2), 2003, 353-367.

[2] C. Zhang, M. Liu, W. Nie, & S. Zhang, Identifying global exceptional patterns in multi-database mining, *IEEE Computational Intelligence Bulletin*, 3(1), 2004, 19-24.

[3] M. Leonard, & B. Wolfe, Mining transactional and time series data, *SUGI 30 Proceedings*, 2005, paper 080-30.

[4] J. Han, J. Pei, & Y. Yiwen, Mining frequent patterns without candidate generation, *Proceedings ACM SIGMOD Conf. Management of Data*, 2000, 1-12.

[5] R. Agrawal, & R. Srikant, Fast algorithms for mining association rules, *Proc. of 20th Very Large databases (VLDB) Conf.*, 1994, 487-499.

[6] R. Agrawal, T. Imielinski, & A. Swami, Mining association rules between sets of items in large databases, *Proc. ACM SIGMOD Conf. Management of Data*, 1993, 207-216.

[7] A.G. Bluman, *Elementary Statistics: A Step by Step Approach* (McGraw Hill, 2006).

[8] J. Brockwell, & D. Richard, *Introduction to Time Series and Forecasting* (Springer, 2002).

[9] C.L. Liu, *Elements of discrete mathematics* (McGraw-Hill, 1985).

[10] B. Liu, Y. Ma, & R. Lee, Analyzing the interestingness of association rules from the temporal dimension *IEEE International Conference On Data Mining*, Silicon Valley, CA, 2001, 377-384.

[11] P.-N. Tan, V. Kumar, & J. Srivastava, Selecting the right interestingness measure for association patterns, *Proceedings of SIGKDD Conference*, 2002, 32-41.

[12] X. Wu, C. Zhang, & S. Zhang, Database classification for multi-database mining. *Information Systems*, 30(1), 2005, 71-88.

[13] T. Zhang, R. Ramakrishnan, & M. Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2), 1997, 141-182.

[14] V. Estivill-Castro, & J. Yang, Fast and robust general purpose clustering algorithms. *Data Mining and Knowledge Discovery*, 8(2), 2004, 127-150.

[15] A.K. Jain, M. N. Murty, & P. J. Flynn, Data clustering: A review, *ACM Computing Surveys*, 31(3), 1999, 264-323.

[16] J.P. Yoon, L. Yixin, & N. Junghyum, A bitmap approach to trend clustering for prediction in time-series databases, *Proceedings of Data Mining and Knowledge Discovery: Theory, Tools, and Technology II*, Florida, April 2001.

[17] K. Yang, & C. Shahabi, On the stationarity of multivariate time series for correlation-based data, *Proceedings of ICDM 2005*, 805-808.

[18] S. Zhang, X. Wu, & C. Zhang, Multi-database mining, *IEEE Computational Intelligence Bulletin*, 2(1), 2003, 5-13.

[19] S. Zhang, C. Zhang, & X. Wu, *Knowledge discovery in multiple databases* (Springer, 2004).

[20] Frequent itemset mining dataset repository. <http://fimi.cs.helsinki.fi/data>.

[21] UCI ML repository content summary. <http://www.ics.uci.edu/~mllearn/MLSummary.html>.