DATA MINING OPPORTUNITIES IN MODERN INFORMATION SYSTEMS

Romanas Tumasonis[1,2]*, Inga Tumasoniene[3]

[1]Vilnius College, Jasinskio street 15, Vilnius, Lithuania
[2]Vilnius Business College, Kalvariju street 125, Vilnius, Lithuania,
[3]Vilnius Gediminas Technical University, Sauletekio al. 11, Vilnius, Lithuania

Received 12 February 2008, accepted 17 June 2008

Abstract. The paper deals with the search and analysis of the subsequences in large volume sequences (texts, DNA sequences etc.). A new algorithm ProMFS for mining frequent sequences is proposed and investigated. The algorithm is based on the estimated probabilistic-statistical characteristics of the appearance of the elements of the sequence and their order. The algorithm builds a new much shorter sequence and makes decisions on the main sequence in accordance with the results of analysis of the shorter one.

Keywords. Data mining, text mining, frequent sets, frequent sequences.

Short name: Data mining opportunities

** Corresponding author, email: r.tumasonis@eif.viko.lt*

Introduction

The task of association mining is to discover a set of attributes shared among a large number of objects in a given database. For example, consider the books database where the attributes represent authors or books and the objects represent customers. The discovered patterns are the set of books most frequently bought together by the customers. An example could be that 35% of the people who buy Den Brown's *The Da Vinci code* also buy *Angels and Demons*. The store can use this knowledge for promotions, shelf placement and etc. There are many potential application areas for association rule technology, which include, store layout, catalogue design [1], customer segmentation [2], telecommunication alarm diagnosis and so on.

The general task of discovering all frequent associations and rules in very large databases is quite challenging. The search space is exponential in the number of database attributes, and with many of database objects the problem of input/output minimization becomes paramount. Though, most current approaches are iterative in nature, requiring multiple database scans, which is obviously very expensive. Some of the methods, especially those using some form of sampling, can be sensitive to the data-skew, which can adversely affect performance [3]. Also, most approaches use very complicated internal data structures which have poor locality and add additional space and computation overheads. Our goal is to overcome all of these limitations [4].

Many algorithms combining the features listed above depend on the database format, the decomposition technique, and the search procedure used. Most popular algorithms is Eclat [5] (Equivalence CLAss Transformation), MaxEclat [6], Clique [7], MaxClique [8], Top-Down [9], and AprClique [10]. Our algorithms not only minimize input/output costs by making only one database scan, but also minimize computation costs by using efficient search schemes. The algorithms are particularly effective when the discovered frequent item sets are long. Our tide list-based approach is also insensitive to the data-skew.

1. Data mining algorithms

The task of discovering all frequent sequences in large databases is quite challenging. The search space is extremely large. For example, with *m* attributes there are *O(m k)* potentially frequent sequences of length *k*. With many millions of objects in the database, the problem of input/output minimization becomes paramount. Most algorithms are iterative in nature and requiring as many full database scans as the longest frequent sequence, which is naturally very expensive.

Assume that we have a set *L*, consisting of *m* distinct elements, also called *items*.

$$L = \{i_1, i_2, ..., i_m\} \quad (1)$$

An itemset is a nonempty unordered collection of items. A sequence is an ordered list of itemsets. A sequence ($\alpha_1 \to \alpha_2 \to ... \to \alpha_n$) is denoted as $_\cup\alpha$, where the sequence element α_j is an itemset. An item can occur only once in an itemset, but it can occur multiple times in different item sets of a sequence. We solve a partial problem, where itemset consists of one item only. A sequence α is a subsequence of another sequence β:

$$\alpha = (\alpha_1 \to \alpha_2 \to \alpha ... \to \alpha_n), \quad (2)$$

$$\beta = (\beta_1 \to \beta_2 \to ... \to \beta_n), \quad (3)$$

if there exist such numbers $t_1, t_2, ..., t_n$, where $t_{j+1} = t_j + 1,\ j = 1, ..., n$ and $\alpha_j = \beta_{t_j}$ for all a_j. Here, β_{t_j} are elements of the set *L*. We analyze the sequence (the main sequence) *S* that is formed from single elements of *L* (not their sets, as in the classical formulation of the problem). In general, the number of elements in *S* is much larger than that in *L*. We have to find the most frequent subsequences in *S*. The problem is to find subsequences whose appearance frequency is more than some threshold called *minimum support*, i.e. the subsequence is frequent if it occurs in the main sequence no less frequently than the minimum support [11].

The main problem is as follows: it is necessary to define only potentially useful subsequences, check up and prolong them gradually. Two algorithms are analyzed that analytically eliminate such subsequences that actually cannot be frequent. Two implementations of *Generated-Sequence-Pattern algorithm* (GSP) with and without economizing memory are tested. A recursive approach to sequence mining is suggested. It enables saving memory, too. The algorithms are compared and the cases, in which one algorithm works better than another one, are determined,

The new algorithm for mining frequent sequences ProMFS is based on the estimation of the statistical characteristics of the main sequence: i) the probability of an element in the sequence; ii) the probability for one element to appear after another one; iii) the average distance between different elements of the sequence

The main idea of the algorithm is the following:

i) some characteristics of the position and interposition of elements are determined in the main sequence;

ii) much shorter new model sequence $\tilde{C}$ is generated according to these characteristics;

iii) new sequence is analyzed with the GSP algorithm (or any similar one);

iv) frequency of subsequences in the main sequence is estimated by the results of the GSP algorithm applied on the new sequence.

Let us assume the following set *L=acc. to Eq. (1)* as the set consisting of *m* distinct elements. Probability of occurrence of element i_j in the main sequence where $i_j \in L,\ j = 1, ..., m$ could be calculated as follows:

$$P(i_j) = \frac{V(i_j)}{VS}, \quad (4)$$

V(i) represents the number of elements i_j in the main sequence *S*; *VS* represents the length of the sequence. Note that

$$\sum_{j=1}^{m} P(i_j) = 1, \quad (5)$$

Probability $P(i_j \mid i_v)$ represents the appearance of element i_v after element i_j, where $i_j, i_v \in L,\ j, v = 1, ..., m$. Note that

$$\sum_{v=1}^{m} P(i_j \mid i_v) = 1, \quad (6)$$

for all *j*=1…m.

Distance $D(i_j \mid i_v)$ between elements i_j and i_v, where $i_j, i_v \in L$, $j, v = 1,..., m$, represents the number of elements that are between i_j and the first found i_v from i_j to the end of the main sequence (i_v is included). The distance between two adjacent elements of the sequence is equal to 1. Set of average distances represents a matrix of average distances Ã. Elements of the matrix are as follows:

$$a_{jv} = \text{Average}\,(D(i_j \mid i_v),\ i_j, i_v \in L),\ j, v = 1,..., m. \qquad (7)$$

All these characteristics can be obtained during one search through the main sequence. According to these characteristics, a much shorter model sequence $\tilde{C}$ is generated, the length of which is *l*. Denote its elements by c_r, $r = 1,..., l$. The model sequence $\tilde{C}$ will contain elements from *L*: $i_j \in L$, $j = 1,..., m$. For the elements c_r, a numeric characteristic $Q(i_j, c_r)$, $r = 1,..., l$, $j = 1,..., m$, is defined. Initially, $Q(i_j, c_r)$ is the matrix with zero values that are specified after the statistical analysis of the main sequence. The complementary function $\rho(c_r, a_{rj})$ is introduced that increases the value of characteristics $Q(i_j, c_r)$ by 1. The first element c_1 of the model sequence $\tilde{C}$ is that from *L* corresponding to $\max(P(i_j))$, $i_j \in L$. According to c_1, the function

$$\rho(c_1, a_{1j}) \Rightarrow Q(i_j, 1 + a_{1j}) = Q(i_j, 1 + a_{1j}) + 1 \qquad (8)$$

$j = 1,..., m$, is activated. Remaining elements c_r, $r = 2,..., l$, are chosen in the way described below. Consider the *r*-th element c_r of the model sequence $\tilde{C}$. The decision, which symbol from *L* should be chosen as c_r, will be made after calculating $\max(Q(i_j, c_r))$, $i_j \in L$. If for some p and t we obtain that

$$Q(i_p, c_r) = Q(i_t, c_r) \qquad (9)$$

then element c_r is chosen by maximal value of conditional probabilities, i.e. by

$$\max(P(c_{(r-1)} \mid i_p), P(c_{(r-1)} \mid i_t)) \qquad (10)$$

$c_r = i_p$ if following condition is satisfied:

$$P(c_{(r-1)} \mid i_p) > P(c_{(r-1)} \mid i_t) \qquad (11)$$

and $c_r = i_t$ if following condition is satisfied:

$$P(c_{(r-1)} \mid i_p) < P(c_{(r-1)} \mid i_t) \qquad (12)$$

If these values are equal, i.e.

$$P(c_{(r-1)} \mid i_p) = P(c_{(r-1)} \mid i_t) \qquad (13)$$

then c_r is chosen depending on $\max(P(i_p), P(\mid i_t))$. After choosing the value of c_r, the function

$$\rho(c_r, a_{rj}) \Rightarrow Q(i_j, r + a_{rj}) = Q(i_j, r + a_{rj}) + 1 \qquad (14)$$

is activated. All these actions are performed consecutively for every $r = 2, \ldots, l$. In this way, we get the model sequence that is much shorter than the main one and that may be analyzed by the GSP algorithm with much less computational efforts.

We have changed one characteristic (the average distance between different elements of the sequence) to the frequent distance between different elements of the sequence. F is the matrix of these frequent distances. The elements of the matrix are described follows:

$$f_{jv} = \text{Frequent}\,(D(i_j \mid i_v),\ i_j, i_v \in L),\ j, v = 1,..., m. \qquad (15)$$

2. Results of the research

Let us compare these two different implementations and recursive algorithm by a special example. Suppose we have a text file with 90 000 A and B symbols. Our goal is to find all the frequent sequences. We have compared the time expenditure and memory use. The results are shown in Fig. 1 and Fig. 2

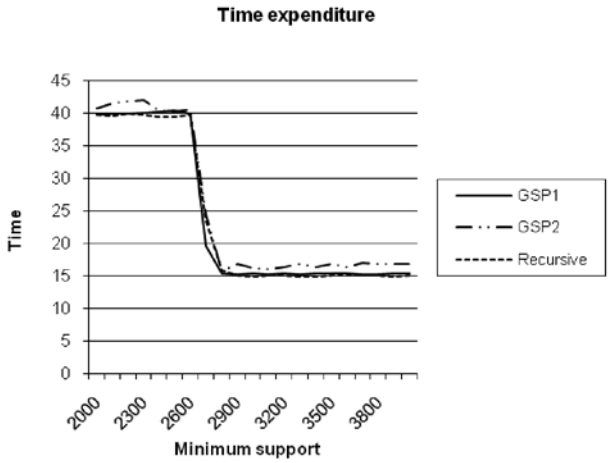

Fig. 1. Time expenditure of GSP implementations and recursive algorithm.

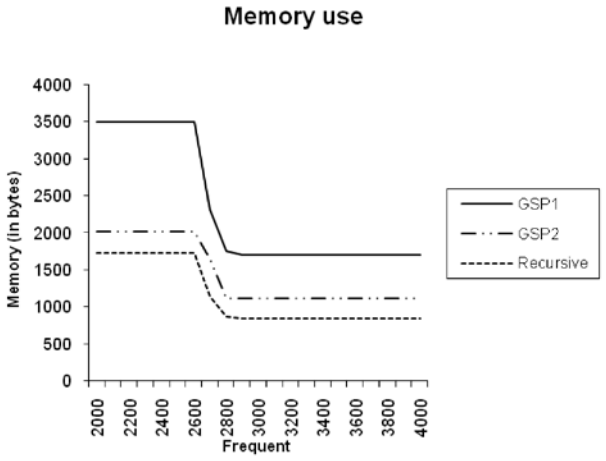

Fig. 2. Memory use of GSP implementations and recursive algorithm

The probabilistic mining of frequent sequences was compared with the GSP algorithm. We have generated a text file of 100,000 letters (1000 lines and 100 symbols in one line). L={A, B, C}, i.e. m=3, $i_1 = A$, $i_2 = B$, $i_3 = C$. In this text we have included one very frequent sequence ABBC. This sequence is repeated 20 times in one line. The remaining 20 symbols of the line are selected at random. First of all, the main sequence (100,000 symbols) was tested with the GSP algorithm. The results are presented in Fig. 3 and Fig. 4. They will be discussed in more detail together with the results of ProMFS. ProMFS generated the model sequence $\tilde{C}$ of length l=40.

This model sequence was examined with the GSP algorithm using the following minimum supports: 8, 9, 10, 11, 12, 13, and 14. The results are presented in Fig. 3 and 4. Fig. 3 shows the number of frequent sequences found by both GSP and ProMFS. Fig. 4 illustrates the expenditure of computation time used by both GSP and ProMFS to obtain the results of Fig. 3 (the minimum support in ProMFS is Ms=8; the results are similar for larger Ms). The results in Fig. 3 indicate that if the minimum support in GSP analyzing the main sequence is comparatively small (less than 1500 with the examined data set), GSP finds much more frequent sequences than ProMFS. When the minimum support in GSP increases from 2500 to 6000, the number of frequent sequences by GSP decreases and the number of those by ProMFS increases. In the range of [2500, 6000], the number of frequent sequences found by both GSP and ProMFS is rather similar. As the minimum support in GSP continues growing, the number of frequent sequences found by both algorithms becomes identical. When comparing the computing time of both algorithms (see Fig. 5), we can conclude that ProMFS operates considerably faster. In the range of the minimum support in GSP [2500, 6000] ProMFS needs approximately 20 times less of computation time as compared with GSP to obtain the similar result.

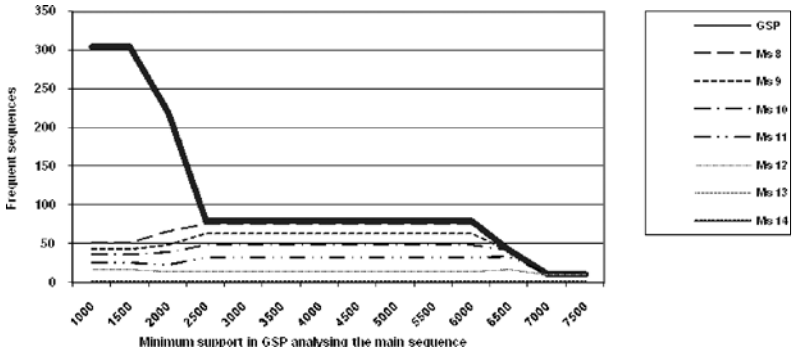

Fig. 3. The number of frequent sequences found by GSP and ProMFS (the minimum support in ProMFS is *Ms*=8, … ,14).

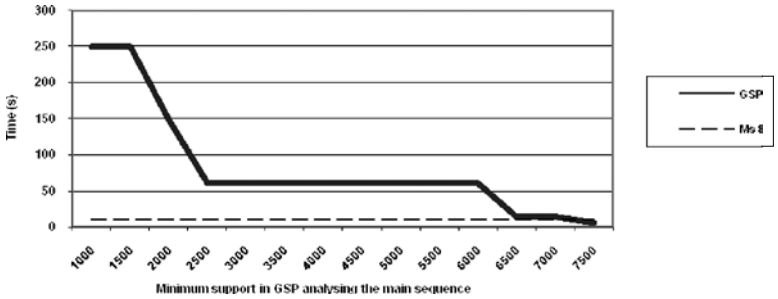

Fig. 4. The computing time by GSP and ProMFS (the minimum support in ProMFS is *Ms*=8).

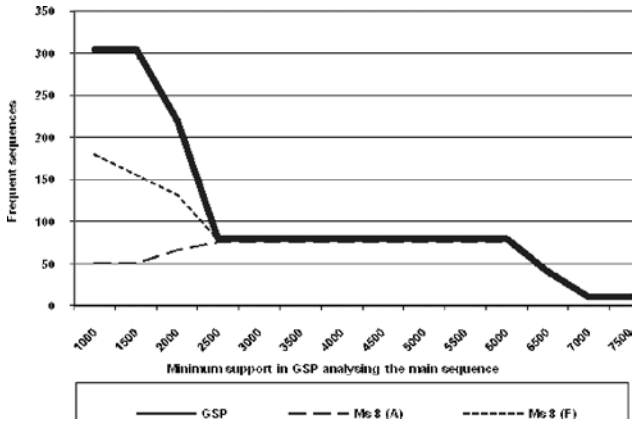

Fig. 5. The computing time by GSP and ProMFS (the minimum support in ProMFS is *Ms*=8).

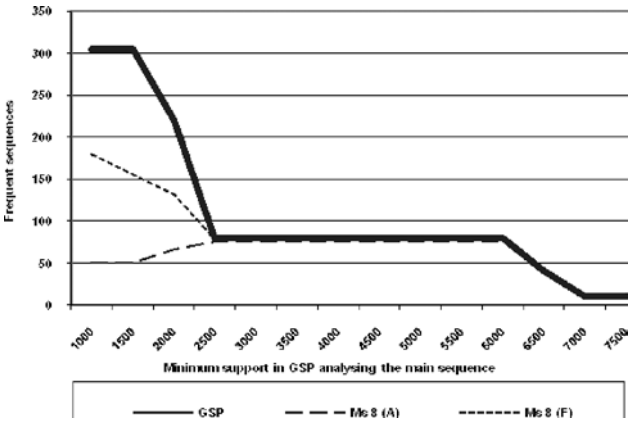

Fig. 6. The number of frequent sequences found by GSP and ProMFS with the average matrix (Ms 8 (A)) and the frequent matrix (Ms 8 (F)) (the support in ProMFS is *Ms*=8).

Fig. 6 illustrates the expenditure of computation time by both GSP and ProMFS with two different matrixes to obtain the results of Fig. 3 (the minimum support in ProMFS is *Ms*=8; the results are similar for larger *Ms*). The results in Fig. 3 indicate that if the minimum support in GSP analyzing the main sequence is comparatively small (less than 1500 with the examined data set), GSP finds many more frequent sequences than ProMFS. But the results of ProMFS with the matrix of frequent distance are better than with the matrix of average distance.

Databases are designed for programs to process automatically; text is written for people to read. Text mining methods can be used in bioinformatics for analysis of DNA sequences. *A genetic DNA sequence* is a succession of special symbols (letters) representing the primary structure of a hypothetical or real DNA strand or molecule. The four letters of sequence are A, C, G, and T, representing the adenine, cytosine, guanine, thymine, which is nucleotide subunits of a DNA. The DNA sequence printed with no spaces between letters: AAACTGGTCTGAC [2].

We have worked with real DNA data from [12] (*34565 Homo sapiens* chromosome 1 genomic contig). The size of the database is equal to 250 MB. The results are shown in Fig. 7 and Fig. 8.

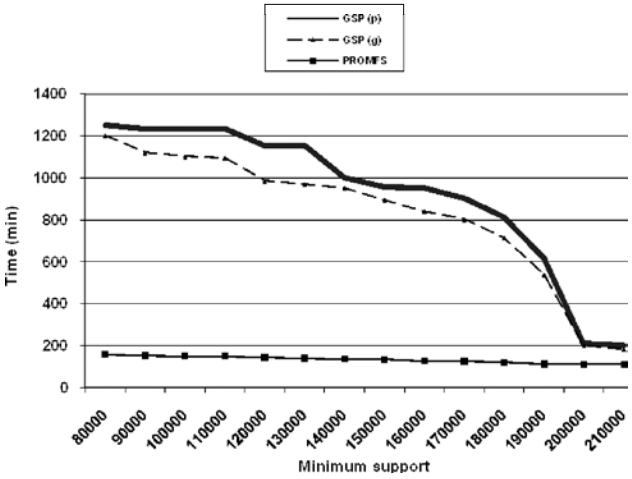

Fig. 7. The computation time by both GSP and ProMFS in *Homo sapiens* chromosome 1 genomic contig.

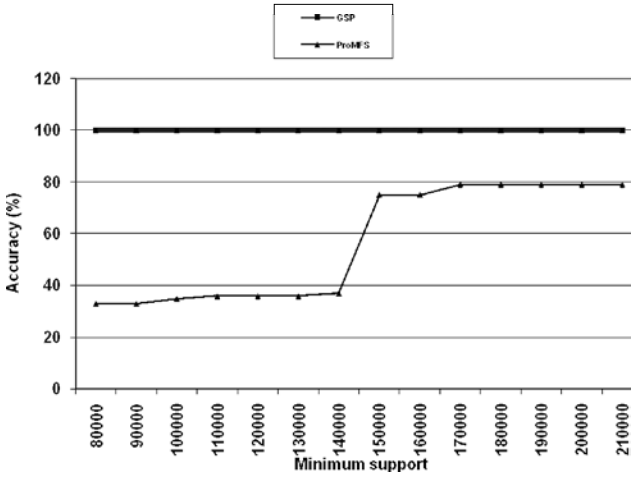

Fig. 8. The accuracy by both GSP and ProMFS in *Homo sapiens* chromosome 1 genomic contig.

Conclusions

Two implementations of the Generated sequence pattern (GSP) algorithm and a recursive algorithm have been examined. The first implementation disregards saving memory, while the second one minimizes the memory consumption. Both implementations and a recursive algorithm are time-intensive. Therefore, intense memory use requires relatively little time and is observed in the case of large data sets. The best memory saving algorithm is recursive algorithm because this algorithm uses memory just for frequent sequences, but not for all generated candidates.

The new algorithm ProMFS for mining frequent sequences with matrix of frequent distance has been proposed. It is based on the estimated probabilistic-statistical characteristics of the appearance of elements of the sequence and their order: the probability of an element in the sequence, the probability for one element to appear after another one, and the frequent distance between different elements of the sequence. The algorithm builds a much shorter new model sequence and makes the decision on the main sequence in accordance to the results of analysis of the shorter one. The model sequence may be analyzed by the GSP or other algorithm for mining frequent sequences: the frequency of subsequences in the main sequence is estimated by the results of the model sequence analysis.

The experimental research indicates that the new algorithm modification enables saving the computation time to a large extent and loses fewer sequences compared with the older algorithm modification, which is especially important when analyzing very large data sequences.

References

[1] Agrawal R.C., Agrawal C.C., Prasad V.V. (2000) Depth first generation of long patterns. *In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* Boston, Massachusetts. P. 108-118.

[2] [http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html], retrieved 09/09/2007

[3] Zaki M.J. (2000) *Parallel sequence mining on shared-memory machines.* In: Zaki, M.J., Ching-Tien Ho (eds): Large-scale Parallel Data Mining. Lecture Notes in Artificial Intelligence, Vol. 1759. Springer-Verlag Berlin Heidelberg New York. P. 161-189.

[4] Zaki M.J. (2001) *SPADE: An efficient algorithm for mining frequent sequences.* Machine Learning Journal, 42 (1/2) p. 31-60 (Fisher, D. (ed.): Special issue on Unsupervised Learning).

[5] Pei P.J., Han J., Wang W. (2002) Mining Sequential Patterns with Constraints in Large Databases. *In Proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM'02),* McLean, VA. P. 18-25.

[6] Pinto P., Han J., Pei J., Wang K., Chen Q., Dayal U. (2001) Multi-Dimensional Sequential Pattern Mining. *In Proceedings of the 10th ACM International Conference on Information and Knowledge Management* (CIKM'01), Atlanta, Georgia. P. 81-88.

[7] Pei J., Han J., Mortazavi-Asl B., Pinto H., Chen Q., Dayal U., Hsu M.-C. (2001) PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. *In Proc. 17th International Conference on Data Engineering ICDE. Heidelberg.* P. 215-224.

[8] Han J., Pei J. (2000) FreeSpan: Frequent pattern-projected sequential pattern mining. *In Proc. Knowledge Discovery and Data Mining.* P. 355-359.

[9] Ayres J., Flannick. J., Gehrke J., Yiu T.(2002) Sequential pattern mining using a bitmap representation. In Proc. *Knowledge Discovery and Data Mining.* P. 29-435.

[10] Zaki M.J. (2000) Scalable algorithms for association mining..*Knowledge and Data Engineering* IEEE Transactions onVolume 12, Issue 3.

[11] Kum H.C., Pei J., Wang W. (2003) ApproxMAP: Approximate Mining of Consensus Sequential Patterns. *In Proceedings of the 2003 SIAM International Conference on Data Mining (SIAM DM '03)*, San Francisco, CA. P. 311-315.

[12] [ftp://ftp.ncbi.nih.gov/genbank/], retrieved 09/09/2007.