Discovery of Sequential Patterns with Quantity Factors

Karim Guevara Puente de la Vega
Universidad Católica de Santa María /Arequipa
Universidad Nacional de San Agustín / Arequipa
kguevara@ucsm.edu.pe

Cesar Beltrán Castañón
Departamento de Ingeniería
Pontificia Universidad Católica del Perú / Lima
cbeltran@pucp.pe

Abstract

The sequential pattern mining stems from the need to obtain patterns that are repeated in multiple transactions in a database of sequences, which are related to time, or another type of criterion. This work presents the proposal of a new technique for the discovery of sequential patterns from a database of sequences, where the patterns not only provide information on how these relate to the time, but also, that in the mining process itself should be included the quantity factors associated with each of the items that are part of a sequence, and as a result of this process can be obtain information relating to how they relate these items with regard to the amounts associated. The proposed algorithm uses divide and conquer techniques, as well as indexing and partitioning of the database.

1 Credits

This document was written as part of the development of the 1st Symposium on Information Management and Big Data, SIMBig 2014. It has been adapted from the instructions for earlier ACL.

2 Introduction

The sequential pattern mining is the process by which you get the relationships between occurrences of sequential events, to find if there is a specific order in which these events occur. In relation to this area of study there are many investigations, all of them makes use of the restriction of minimal support, some include other restrictions, such as for example the time interval in which it is required that the events happen, also the use of taxonomies as defined by the user, and the fact of allowing the items in a sequence not necessarily must have occurred in a single transaction, but could be in two or more, always and when their times of each of these transactions is within some small window of time determined by the user.

In addition, the algorithms for mining sequential patterns of dealing with the previous sequential patterns in a uniform manner, despite the fact that these patterns individually in a sequence can have important differences such as the associated amount to each item that make up each pattern.

For the foregoing reasons, in the present paper proposes a technique by which it is intended to exploit these intrinsic relationships of the sequential patterns, in this specific case the relationship to the amount of each of the items. The inclusion of this aspect in the sequential pattern mining, you can afford to get a set of sequential patterns that are not only common but also let us know how these amounts associated with each item that is included in a sequential pattern frequent relates. The inclusion of the restriction of quantity within the extraction process of the frequent sequential patterns we could provide information much more meaningful.

The article is organized as follows: Section 2 is on the previous work. Section 3 gives a description of the problem. Section 4 introduces the technical proposal. Section 5 shows the experiments and results. The conclusions and future work are shown in section 6 and finally the references.

3 Previous works

The techniques of discovery of association rules are essentially boolean, due to which are discarded the quantities of the items purchased and only pay attention to if something was purchased or not. An important area of study is the sequential pattern mining that involves the extraction of patterns that are repeated in multiple transactions in a transactional database, which are related to time or another type of sequence.

The problem of the sequential pattern mining was introduced by Agrawal and Srikant (1995) set the example of the typical income of clients in a rental shop videos. Customers often rent "Star Wars", then "Empire Strikes Back" and then "Return of the Jedi". All these incomes not necessarily should have been made consecutively, that is to say, there could be customers that

leased any other video in the middle of the previous sequence, so that these sequences of transactions also fall into the same pattern.

The researches on mining sequential patterns are based on events that took place in an orderly fashion at the time.

Most of the implemented algorithms for the extraction of frequent sequences, using three different types of approaches according to the form of evaluating the support of the candidate sequential patterns. The first group of algorithms is based on the ownership apriori, introduced by Agrawal and Srikant (1994) in the mining of association rules. This property suggests that any sub pattern from a frequent pattern is also frequent, allowing pruning sequences candidates during the process of lead generation. Based on this heuristics, Agrawal and Srikant (1995) proposed algorithms as the AprioriAll and AprioriSome. The substantial difference between these two algorithms is that the AprioriAll generates the candidates from all the large sequences found, but that might not be lowest panning values, however, AprioriSome only counts those sequences that are large but lowest panning values, thus reducing the search space of the patterns.

In subsequent work, Srikant and Agrawal (1996) propose the same algorithm GSP (Generalization Sequential Patterns), also based on the technical apriori, surpassing previous in 20 magnitudes of time. Until this time, the algorithms that had been proposed for mining sequential patterns focused on obtaining patterns taking into account only the minimal support given by the user. But these patterns could fit into transactions that had been given at intervals of time very distant, which was not convenient for the purposes of mining. So, in this paper, we propose the idea that in addition to the minimal support, the user could be in the ability to specify your interest in obtaining patterns that fit into transactions that have been given in certain periods of time, and this is made from the inclusion of restrictions on the maximum and minimum distance, the size of the window in which the sequences and the inheritance relationships - taxonomies, which are cross-relations through a hierarchy.

In these algorithms based on the principle of apriori, the greater effort focused on developing specific structures that allow sequential patterns represent the candidates and in this way make the counting operations support more quickly.

The second group is the algorithms that seek to reduce the size of the set of scanned data, by means of task execution of projection of the initial data base and the obtaining of patterns, without involving a process of lead generation. Using this technique and approach under the divide and rule, Han et al. (1996) proposed the algorithm FreeSpan (*Frecuent Pattern-Project Sequential Pattern mining*), and Pei et al. (2001) proposes PrefixSpan (*Prefix-projected Sequential Pattern mining*). In these algorithms the database of sequences is projected recursively in a set of small databases from which the fragments of sub sequences grow based on the current set of frequent sequences, where the patterns are extracted.

Han et al. (1996)] show that FreeSpan extracts the full set of patterns and is more efficient and considerably faster than the algorithm GSP. However, a sub sequence can be generated by the combinations of sub strings in a sequence, while the projection in FreeSpan must follow the sequence in the initial database without reducing the length. In addition, it is very expensive the fact that the growths of a sub sequence it will be explored in any point of the division within a candidate sequence. As an alternative to this problem, Pei (2001) proposes PrefixSpan. The general idea is to examine only the prefixes for the sub project only sequences and their corresponding sub sequences postfijas within databases planned. In each of these databases planned, it will find the sequential patterns expanded exploration only local patterns frequently. PrefixSpan extracts the full set of patterns and their efficiency and implementation are considerably better both GSP and FreeSpan.

The third group is formed by algorithms that kept in memory only information necessary for the evaluation of the bracket. These algorithms are based on the calls of occurrence lists that contain the description of the location where the patterns occur in the database. Under this approach, Zaki (2001) proposes the SPADE algorithm (*Sequential Pattern Discovery using Equivalence classes*) where he introduces the technical processing of the data base to vertical format, in addition there is a difference from the algorithms based on apriori, it does not perform multiple passes on the database, and you can extract all the frequent sequences in only three passes. This is due to the incorporation of new techniques and concepts such as the list of identifiers (id-list) with vertical format that is associated with the sequences. In these lists by means of temporary unions can be generated frequent sequences. Also used the grid based approach to

break down the search space in small classes that can be processed independently in the main memory. Also, uses the search in both breadth and depth to find the frequent sequences within each class.

In addition to the techniques mentioned earlier, Lin and Lee (2005) proposes the first algorithm that implements the idea of indexing called Memisp memory (Memory Indexing for sequential pattern mining). The central idea of Memisp is to use the memory for both the data streams as to the indexes in the mining process and implement a strategy of indexing and search to find all frequent sequences from a sequence of data in memory, sequences that were read from the database in a first tour. Only requires a tour on the basis of data, at most, two for databases too large. Also avoids the generation of candidates and the projection of database, but presented as disadvantage a high CPU utilization and memory.

The fourth group of algorithms is composed of all those who use fuzzy techniques. One of the first work performed is the Wang et al. (1999), who propose a new data-mining algorithm, which takes the advantages of fuzzy sets theory, to enhance the capability of exploring interesting sequential patterns from the databases with quantitative values. The proposed algorithm integrates concepts of fuzzy sets and the AprioriAll algorithm to find interesting sequential patterns and fuzzy association rules from transaction data. The rules can thus predict what products and quantities will be bought next for a customer and can be used to provide some suggestions to appropriate supervisors.

Wang et al. (1999) propose fuzzy quantitative sequential patterns (FQSP) algorithm, where an item's quantity in the pattern is represented by a fuzzy term rather than a quantity interval. In their work an Apriori-like algorithm was developed to mine all FQSP, it suffers from the same weaknesses, including: (1) it may generate a huge set of candidate sequences and (2) it may require multiple scans of the database. Therefore, an Apriori-like algorithm often does not have a good performance when a sequence database is large and/or when the number of sequential patterns to be mined is large.

Chen et al. (2006) propose divide-and-conquer fuzzy sequential mining (DFSM) algorithm, to solve the same problem presented by Hong using the divide-and-conquer strategy, which possesses the same merits as the PrefixSpan algorithm; consequently, its performance is better than Wang et al.

Fiot (2008) in her work suggests that an item quantitative is partitioned into several fuzzy sets. In the context of fuzzy logic, a diffuse item is the association of a fuzzy set b to its corresponding item x, i.e. $[x,b]$. In the DB each record is associated with a diffuse item $[x,b]$ according to their degree of membership. A set of diffuse items will be implicated by the pair (X,B), where X is the set of items, and B is a set of fuzzy sets.

In addition, it argues that a sequence *g-k-sequence* $(s_1, s_2, \ldots, s_p)$ is formed by g item sets diffuse $s=(X,B)$ grouped to diffuse k items $[x,b]$, therefore the sequential pattern mining diffuse consists in finding the maximum frequency diffuse *g-k-sequence*.

Fiot (2008), provides a general definition of frequency of a sequence, and presents three algorithms to find the fuzzy sequential patterns: *SpeedyFuzzy*, which has all the objects or items of a fuzzy set, regardless of the degree, if it is greater than 0 objects have the same weight, *MiniFuzzy* is responsible for counting the objects or items of a fuzzy set, but supports only those items of the sequence that candidate have a greater degree of belonging to a specified threshold; and *TotallyFuzzy* that account each object and each sequence. In this algorithm takes into account the importance of the set or sequence of data, and is considered the best grade of membership.

4 Description of the Problem

A sequence s, denoted by $<e_1e_2 \ldots i_n>$, is an ordered set of n elements, where each element e_i is a set of objects called *itemset*. An *itemset*, which is denoted by $(x_1 [c_1], x_2 [c_2], \ldots, X_q[c_q])$, is a non-empty set of elements q, where each element x_j is an item and is represented by a literal, and c_j is the amount associated with the item x_j that is represented by a number in square brackets. Without loss of generality, the objects of an element are supposed to be found in lexicographical order by the literal. The size of the sequence s, denoted by $|s|$, is the total number of objects of all elements of the s, so a sequence s is a *k-sequence*, if $|s|=k$.

For example, $<(a[5])(c[2])(a[1])>$, $<(a[2],c[4])(a[3])>$ and $<(b[2])(a[2],e[3])>$ are all 3-sequences. A sequence $s = <e_1e_2 \ldots i_n>$ is a sub-sequence of another sequence of $s'=<e_1'e_2' \ldots e_m'>$ if there are $1 \leq i_1 < i_2 < \ldots < i_n \leq m$ such that $e_1 \subseteq e_{i1}'$, $e_2 \subseteq e_{i2}'$, ... , and $e_n \subseteq e_{in}'$. The sequence s'

contains the sequence *s* if *s* is a sub-sequence of *s'*.

Similarly, *<(b,c)(c)(a,c,e)>* contains *<(b)(a,e)>* where the quantities may be different.

The support (*sup*) of a sequential pattern X is defined as the percentage on the fraction of records that contains X the total number of records in the database. The counter for each item is increased by one each time the item is found in different transactions in the database during the scanning process. This means that the counter of support does not take into account the quantity of the item. For example, in a transaction a customer buys three bottles of beer, but only increases the number of the counter to support {beer} by one; in other words, if a transaction contains an item, then, the support counter that item only is incremented by one.

Each sequence in the database is known as a sequence of data. The support of the sequence *s*, is denoted as *s.sup*, and represents the number of sequences of data that contain *s* divided by the total number of sequences that there is in the database. *minSup* threshold is the minimum specified by the user. A sequence *s* is frequent if $s.sup \geq minSup$, therefore it will be a sequential pattern frequently.

Then, given the value of the *minSup* and a database of sequences, the problem of the sequential pattern mining is to discover the set of all sequential patterns whose supports are greater equal to the value of the minimum support ($s.sup \geq minSup$).

Definition: given a ρ pattern and a frequent item *x* in the database of sequences, ρ' is a:

- **Pattern Type-1**: if ρ' can be formed by adding to ρ the itemset that contains the item *x*, as a new element of ρ.
- **Pattern Type-2**: if ρ' can be formed by the extension of the last element of ρ with *x*.

The item x is called *stem* of the sequential pattern ρ', and ρ prefix is the pattern of ρ'.

That is, the following database of sequences of figure 1, which includes amounts for the items and that, has six sequences of data.

Sequences
C1 = <(a[1],d[2]) (b[3],c[4]) (a[3],e[2])>
C2 = <(d[2],g[1]) (c[5],f[3]) (b[2],d[1])>
C3 = <(a[5],c[3]) (d[2]) (f[2]) (b[3])>
C4 = <(a[4],b[2],c[3],d[1]) (a[3]) (b[4])>
C5 = <(b[3],c[2],d[1]) (a[3],c[2],e[2]) (a[4])>
C6 = <(b[4],c[3]) (c[2]) (a[1],c[2],e[3])>

Figure 1. Database of sequences

Consider the sequence *C6*, which consists of three elements, the first has the objects *b* and *c*, the second has the object *c*, and the third has the objects *a, c,* and *e*. Therefore, the support of *<(b)(a)>* is 4/6 since all the sequences of data with the exception of *C2* and *C3* contain a *<(b)(a)>*. The sequence *<(a,d)(a)>* is a sub sequence of both *C1* and *C4*; and therefore, *<(a,d)(a)>.sup=2/6*.

Given the pattern *<(a)>* and the frequent item *b*, gets the pattern type-1 *<(a)(b)>* adding *(b)* to *<(a)>*, and the pattern type-2 *<(a,b)>* by the extension of *<(a)>* with *b*.

Similarly, *<(a)>* is the prefix pattern (ρ_pat) which in turn is a frequent sequence, and *b* is the stem of both: *<(a)(b)>* and *<(a,b)>*.

Note that the sequence *null*, denoted by <>, is the ρ_pat of any *1-frequent sequence*. Therefore, a *k-sequence* is like a frequent pattern type-1 or type-2 of a *(k-1)-frequent sequence*.

5 Algorithm for the discovery of sequential patterns with quantity factors - MSP-QF

The algorithm for mining sequential patterns with quantity factors, arises from the need to discover from a database of sequences, the set of sequential patterns that include the amounts associated with each of the items that are part of the transactions in the database, since having this additional information can be known with greater precision not only what is the relationship with respect to the time that exists between the various items involved in the transactions of a sequence, but also as is the relationship to the amount of these items.

The algorithm MSP-QF, it is based on the idea of the use of prefixes, and the creation of indexes from the database of sequences or other indices that are generated during the mining process, where recursively searching for frequent patterns. As a result of the exploration of a particular index, fewer and shorter sequences of data need to be processed, while the patterns that are found will be made longer.

In addition, if the database is very large sequence uses the techniques of partitioning in a manner that the algorithm is applied to each of the partitions as if it were a database of lesser size.

5.1 Procedure of the algorithm MSP-QF

Listed below are the steps of the proposed algorithm.

Step 1: Partitioning and scanning of the database of sequences. Depending on the size of the database are applicable to so it can be partitioned and formatted through and then to scan each of the partitions of independently. For each partition, the sequences are constructed and stored in the structure *DBSeq*. At the same time generates the index of items where is stored the support for each one of them, which is found during the scanning process.

Step 2: The index of items are filtered out those that are frequent, i.e., whose support is greater than or equal to minSup determined by the user. All these items come to form sequences of size |s| =1, therefore, form the set of 1-sequences. For all these sequences frequent item is to write the amounts associated with each item to the time it is saved in the whole of frequent patterns.

Step 3: For each one of the frequent patterns ρ, found in step 2, or as a result of the step 4, the index is constructed ρ_idx, with inputs (*ptr_ds*, *pos*), where *ptr_ds* refers to a sequence of the DB in which appears the ρ pattern, and *pos* is the pair (*posItemSet*, *posItem*), where *posItemSet* is the position of the *itemset* in the sequence and *posItem* the position of the item in the itemset from the sequence where the pattern appears. The values of *pos* allow the following scans are performed only on the basis of these positions in a certain sequence.

Step 4: Find the stems of type-1 and/or type-2 for each ρ pattern and its corresponding index ρ*_idx* generated in the previous step, considering only those items of the sequences referred to in ρ*_idx* and the respective values of pos. At the same time as are the stems are calculated their supports, and in addition is added to the list of quantities of the item that is part of the stem the amount referred to in the item of the sequence of the DB which is being examined. The information of the stems and their quantities are stored in another index of stems. This step is repeated for the same pattern, until they were no longer more stems from this.

Step 5: When there is no more stems, filtered index stems all those who are frequent. For all stems (sequences) frequently, we proceed to discretize the quantities that were associated with each item and stored in the set of frequent patterns. For this, before adding it to the set of frequent patterns, we proceed to verify that the common pattern found recently has not already been added before this set as a result of applying the algorithm to a partition of the database that was processed with previously. If frequent pattern already exists in the set of frequent patterns, the discretization process is again applied to the set of quantities associated with the sequence is stored as a frequent pattern and set of quantities of newly discovered frequent pattern; otherwise, the common pattern found in the current partition is added directly to the set of frequent patterns.

Then we proceed to perform recursively steps 3, 4 and 5 with each one of the frequent patterns that are found in the process.

Discretization Function: This function is responsible for making the set of quantities associated with an item, the range of values given by the mean and standard deviation of this set. For example, given the sequences of the figure 1, the set of quantities associated with the item <(a)> is: 1,5,4,3,1, which after being discretized would be the interval formed by: [2.8±1.6]

To summarize the steps carried out in the proposed algorithm, figure 3 shows a schematic of the entire procedure.

5.2 Algorithm specification MSP-QF

Here we show the specification of the proposed algorithm MSP-QF.

Algorithm *MSP-QF*

In: *DB = database sequences*
minSup = minimum support
partition = number of sequences included in each of the partitions
Out: set of all sequential patterns with quantity factors.
Procedure:
1. Partitioning the DB
2. Each partition scan it in main memory and:
   (i) build sequences and store them in DBSeq structure.
   (ii) index the items and determine the support of each item.
   (iv) associate the quantities of each item in a sequence list of item quantities in the index.
3. Find the set of frequent items
4. For each frequent item x,
   (i) form the sequential pattern ρ = <(x)>
   (ii) call *Discretize*(ρ) to discretize the set of quantities associated with each item x$\in\rho$.
   (iii) storing ρ in the set frequent patterns.
   (iv) call *Indexing* (*x*, <>, *DBSeq*) to build the ρ*-idx* index.
   (v) call *Mining* (ρ, ρ*-idx*) to obtain patterns from index ρ*-idx*.

Subrutine *Indexing* (*x*, ρ, *set_Seq*)

Parameters:

x = one stem type-1 or type-2;
ρ = prefix pattern (ρ-*pat*);
set_Seq = set of data sequences
/ * If *set_Seq* is an index, then each data sequence in the index is referenced by the element *ptr_ds*¸ which is formed at the input (*ptr_ds*, *pos*) index * /

Out: índex ρ'-*idx*, where ρ' represents the pattern formed by the stem *x* and prefix pattern ρ-*pat*.

Procedure:

1. For each data sequence *ds* of *set_Seq*
   (i) If *set_Seq* = *DBSeq* the *pos_inicial* = *0*, else *pos_inicial* = *pos*.
   (ii) Find the stem in each sequence *ds* from the position (*pos_inicial* + *1*),
      1. If the stem *x* is in position *pos* in ds, then insert a pair (*ptr_ds, pos*) in ρ'-idx index, where *ptr_ds* reference to *ds*.
      2. If the stems *x* is equal to the item *x'* of the *ds* sequence, added the quantity *q* associated with the item x', to the list of quantities related to x.
2. Return the ρ'-*idx* index.

Subrutina *Mining*(ρ,ρ-*idx*)

Parameters:

ρ = a pattern;
ρ-*idx* = an índex.

Procedure:

1. For each data sequence *ds* referenced by *ptr_ds* of input *(ptr_ds, pos)* in ρ-*idx*,
   (i) Starting from the (*pos +1*) position until |ds|, determining potential stems and increase in one support each of these stems.
2. Filter those stems that have a large enough support.
3. For each stem x found in the previous step,
   (i) form a sequential pattern ρ" from the prefix pattern ρ-*pat* and the stem *x*.
   (ii) call Discretize(ρ') to discretize the amounts associated with the items of ρ'.
   (iii) call Index(ρ', ρ, ρ'-idx) to build the index ρ'-idx.
   (iv) call Mining(ρ', ρ'-idx) to discover sequential patterns from index ρ'-idx

Subrutina *Discretize*(ρ)

Parameters:

ρ = a pattern that is a sequence;

Output: the arithmetic mean and standard deviation of the amounts associated with each item ρ pattern.

Procedure:

1. For each itemset $\gamma \in \rho$ do
   a) For each item $x \in \gamma$ do
      (i) Calculate the arithmetic mean and standard deviation of the set of quantities associated with the item *x*
      (ii) storing the arithmetic mean and standard deviation in the pattern ρ

Figure 2. Specification of the algorithm MSP-QF

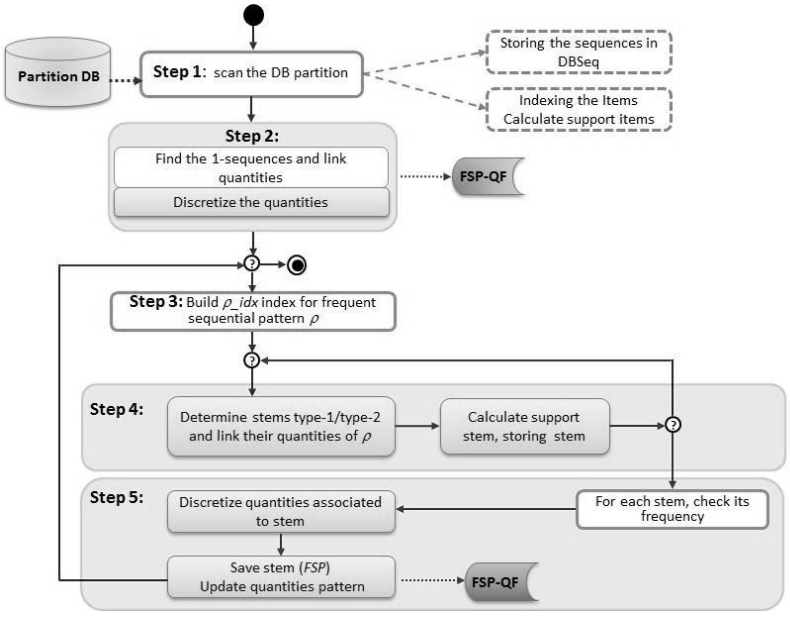

Figure 3. Schema of the procedures of the algorithm MSP-QF

6 Experiments and Result

The experiments to test the technical proposal were implemented in two different scenarios, which are described below.

6.1 Scenario 1: Real data

The technique was applied in the analysis of the market basket of a supermarket. These tests consisted of obtain the set of frequent sequential patterns from the basis of data obtained in the course of three non-consecutive periods. The first period goes from mid-December of 1999 until mid-January 2000. The second period goes from early 2000 until the beginning of June of the same year. The third period goes from late August 2000 until the end of November 2000. This database consists of 88163 transactions, 3000 items unique to approximately 5133 customers.

The purpose of testing is to discover patterns of customer usage in the supermarket, plus get the amount of each of the items that will be purchased by these customers as a result of applying the proposed technique, which will allow us to have more accurate and significant in terms of the quantity purchased of each of the items.

Seven tests were carried out with minimum media 10 %, 2%, 1.5%, 1.0%, 0.75%, 0.50% and 0.25%, which were observed in figure 4. These results were compared with results of the technical Memisp.

minSup (%)	MEMISP		MSP-QF	
	Exe.Time (seg.)	No. Patterns	Exe.Time (seg.)	No. Patterns
10.00	4	50	5	50
2.00	12	824	15	824
1.50	16	1371	19	1371
1.00	22	2773	27	2773
0.75	28	4582	35	4582
0.50	39	9286	50	9286
0.25	72	30831	89	30831

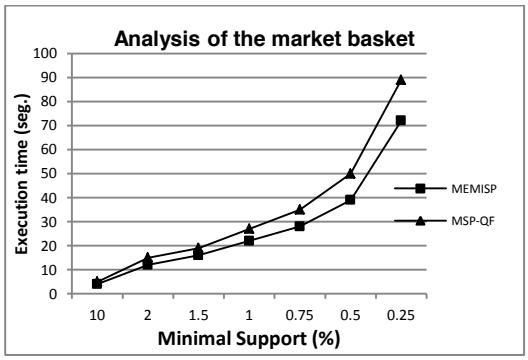

Figure 4. Results of the tests for scenario 1

In the test with *minSup=2%* were obtained 824 sequential patterns with quantity factors, some of which are:

Olive[1.06±0.51]$
Olive[1.03±0.5]$ Paprika[0.37±0.23]$
Olive[1.01±0.5]$ Porro[0.57±0.27]$
Celery[0.56±0.25]$ Lettuce[0.53±0.24], Lentils[0.54±0.23]$
Celery[0.56±0.26]$ Lettuce[0.55km Air ±0.24], Paprika[0.34±0.17]$
Lettuce[0.61±0.25], Lentils[0.56±0.26]$ Porro[0.59±0.24], Paprika[0.33±0.15]$
Porro[0.54±0.27], Lentils[at 0.62±0.25]$ Lentils[0.58±0.25]$ Paprika[0.35±0.17]$

Of these sequential patterns we can clarify the following with regard to purchases made by customers:

- Customers buy only olives in a quantity of 1.06±0.51.
- Customers who have purchased a first time only olive, returning a next time for chili or by porro, with quantities of 0.37±0.23 and 0.57±0.27 respectively . Those who buy after pepper, purchased before olives in a quantity equal to 1.03±0.5, while those who acquire porro did so with an amount equal to 1.01±0.5.
- Those who buy lettuce at the same time buy lentils in amounts equal to 0.61±0.25 and 0.56±0.26 respectively. Later, these same customers buy porro and paprika with amounts equal to 0.59±0.24 and 0.33±0.15.
- Those who buy porro, in the same transaction also buy lentils. Later return to buy only lentils, and a next time buy only paprika, in the amounts listed in the pattern.

6.2 Scenario 2: Synthetic data

This second scenario is generated multiple databases (datasets) of synthetic form by means of the Synthetic Data Generator Tool.

The process followed to synthetic generation of the dataset, it is the describing Agrawal and Srikan (1995), and under the parameters referred to in the work of Lin and Lee (2005).

In this scenario, tests were carried out both of effectiveness, efficiency and scalability.

The evidence of effectiveness and efficiency were made with dataset generated with the following parameters: $NI = 25000$, $NS = 5000$, $N = 10000$, $|S| = 4$, $|I| = 1.25$, $corr_S = 0.25$, $crup_S = 0.75$, $corr_I = 0.25$ and $crup_I = 0.75$.

The results of these tests were compared with the results obtained for the algorithms PrefixSpan-1, PrefixSpan-2 and Memisp.

Efficiency Tests: Ran a first subset of tests for $|C|=10$ and a database of *200,000* sequences, with different values for *minSup*. The results are shown in figure 5.

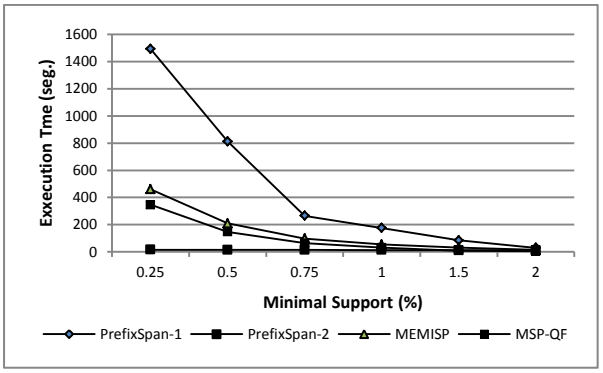

Figure 5. Test results for |C|=10 and |DB|=200K

The second subgroup of tests was conducted with a dataset with values for |C| and |T| of 20 and 5 respectively. This value of |T| implies that the number of items of transactions increases, which represents that the database is also larger and more dense with respect to the number of frequent sequential patterns that may be obtained. The results of these tests are those seen in figure 6.

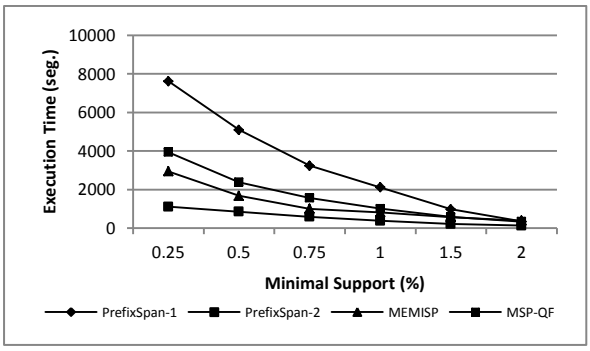

Figure 6. Test results for |C|=20, |T|=5 and |DB|=200K

A last subset of efficiency tests were carried out under the same parameters of the subset above with the exception of |T| increased to 7.5. The results are shown in figure 7.

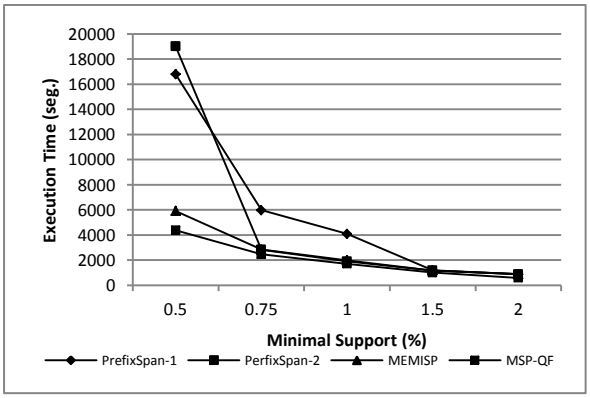

Figure 7. Test results for |C|=20, |T|=7.5 and |DB|=200K

Efficacy tests: Were carried out 92 efficacy trials with the same dataset of the tests of efficiency. In four of the tests carried out with values |C| =20 and |T| equal to 2.5, 5 and 7.5 respectively, it did not achieve the same amount of sequential patterns found with the algorithms PrefixSpan-1 and PrefixSpan-2. These 4 tests represent 4% of the total.

Scalability tests: The scalability tests were used datasets synthetically generated with the same values of the parameters of the first subset of tests of efficiency, and with minimal support equal to 0.75%. The amount of sequences in the dataset for these tests ranged from |DB|=1000K to 10000K, i.e. of a million to 10 million sequences. In figure 8, you can watch the results of these tests.

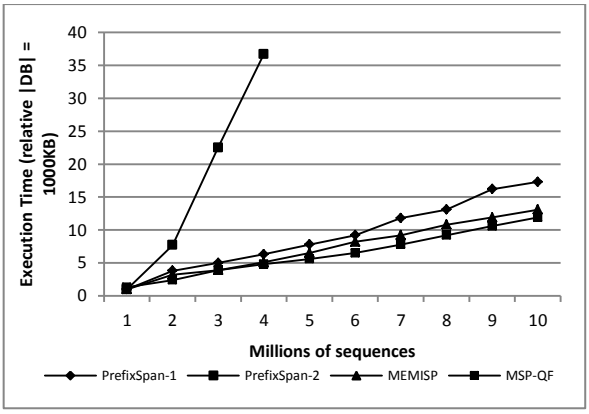

Figure 8, Results of scalability tests for *min-sup=0.75%* and different sizes of datasets

7 Conclusion

We have proposed an algorithm for the discovery of sequential patterns that allows, as part of the mining process, to infer from the amounts associated with each of the items of the transactions that make up a sequence, the quantity factors linked to the frequent sequential patterns.

The technical proposal has been designed in such a way that uses a compact set of indices in which focuses the search for the sequential patterns from frequent patterns that have already been found earlier and that represent the prefixes of the patterns to find. That is why the size of the indexes is decreasing in accordance with the mining process progresses.

In addition, there has been that the information provided by the frequent patterns with factors of quantity, is much more accurate, since not only gives us information on how is the temporal relationship of the items in the various transactions,

but also, what is the relationship of the quantities of some items to others, which enriches the semantics provided by the set of sequential patterns.

Finally, the results obtained in section 5, we can conclude by saying that the technical proposal meets the objectives of the mining process; it is effective, is efficient and is scalable because it has a linear behavior in accordance with the sequence database grows, and that when applied to large data bases his performance turned out to be better than the techniques discussed in this work.

Reference

Agrawal Rakesh and Srikant Ramakrishnan. 1994. *Fast algorithms for mining association rules*. In Proceeding 20th International Conference Very Large Data Bases, VLDB.

Agrawal Rakesh and Srikant Ramakrishnan. 1995. *Minning Pattern Sequential*. 11th International Conference on Data Engineering (ICDE'95), Taipei, Taiwan.

Agrawal Rakesh and Srikant Ramakrishnan. 1996. *Mining Quantitative Association Rules in Large Relational Tables*. In Proceeding of the 1996 ACM SIGMOD Conference, Montreal, Québec, Canada.

Alatas Bilal, Akin Erhan and Karci Ali. 2008. *MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules*. Applied Soft Computing.

Chen Yen-Liang and Haung T. Cheng-Kui. 2006. *A new approach for discovering fuzzy quantitative sequential patterns in sequence databases*. Fuzzy Sets and Systems 157(12):1641–1661.

Fiot Céline. 2008. *Fuzzy Sequential Patterns for Quantitative Data Mining*. In Galindo, J. (Ed.), Handbook of Research on Fuzzy Information Processing in Databases.

Han Jiawei, Pei Jian, Mortazavi-Asl Behzad, Chen Qiming, Dayal Umeshwar and Hsu Mei-Chun. 1996. *Freespan: Frequent pattern-projected sequential pattern mining*. Conference of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.

Karel Filip. 2006. *Quantitative and Ordinal Association Rules Mining (QAR Mining)*. 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES 2006). South Coast, UK: Springer, Heidelberg.

Lin Ming-Yen and Lee Suh-Yin. 2005. *Fast Discovery of Sequential Patterns through Memory Indexing and Database Partitioning*. Journal of Information Science and Engineering.

Market-Basket Synthetic Data Generator, http://synthdatagen.codeplex.com/.

Molina L. Carlos. 2001. *Torturando los Datos hasta que Confiesen*. Departamento de Lenguajes y Sistemas Informáticos, Universidad Politécnica de Cataluña. Barcelona, España.

Papadimitriou Stergios and Mavroudi Seferina. 2005. *The fuzzy frequent pattern Tree*. In 9th WSEAS International Conference on Computers. Athens, Greece: World Scientific and Engineering Academy and Society.

Pei Jian, Han Jiawei, Mortazavi-Asl Behzad and Pinto Helen. 2001. *PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth*. In ICDE '01 Proceedings of the 17th International Conference on Data Engineering.

Srikant Ramakrishnan and Agrawal Rakesh. 1996. *Mining Sequential Patterns: Generalizations and Performance Improvements*. In Proc.5th Int. Conf. Extending Database Technology (EDBT'96), pages 3–17, Avignon, France.

Takashi Washio, Yuki Mitsunaga and Hiroshi Motoda. 2005. *Mining Quantitative Frequent Itemsets Using Adaptive Density-Based Subspace Clustering*. In Fifth IEEE International Conference on Data Mining (ICDM'05). Houston, Texas, USA: IEEE Computer Society.

Wang Shyue, Kuo Chun-Yn and Hong Tzung-Pei. 1999. *Mining fuzzy sequential patterns from quantitative data*", 1999 IEEE Internat. Conference Systems, Man, and Cybernetics, vol. 3, 1999, pp. 962–966.

Zaki Mohammed J. 2001. *SPADE: An efficient algorithm for mining frequent sequences*. Machine Learning, 42(1-2):31–60.