A Sliding Windows based Dual Support Framework for Discovering Emerging Trends from Temporal Data

M. Sulaiman Khan[1], F. Coenen[2], D. Reid[1], R. Patel[3], L. Archer[3]

Abstract: In this paper we present the Dual Support Apriori for Temporal data (DSAT) algorithm. This is a novel technique for discovering Jumping Emerging Patterns (JEPs) from time series data using a sliding window technique. Our approach is particularly effective when performing trend analysis in order to explore the itemset variations over time. Our proposed framework is different from the previous work on JEP in that we do not rely on itemsets borders with a constrained search space. DSAT exploits previously mined time stamped data by using a sliding window concept, thus requiring less memory, minimum computational cost and very low dataset accesses. DSAT discovers all JEPs, as in "naïve" approaches, but utilises less memory and scales linearly with large datasets sets as demonstrated in the experimental section.

1 Introduction

Trend mining is a data mining technique directed at the identification of hidden trends in time series data. There are various approaches to trend mining, many of them founded on time series analysis techniques, but also other established approaches such as Association Rule Mining (ARM). ARM, in its most standard form, is concerned with the identification of patterns (known as frequent itemsets) in data within binary valued attributes. The most common framework for ARM is the "support-confidence" framework (Agrawal and Srikant, 1994). In this framework "support" is the frequency with which an itemset appears in the input data and "confidence" is a measure of the reliability of the identified Association Rules

[1] Department of Computer Science, Liverpool Hope University, L16 9JD, UK email: {khanm,reidd}@hope.ac.uk

[2] Department of Computer Science, University of Liverpool, L69 3BX email: frans@liv.ac.uk

[3] Transglobal Express Ltd. Wirral, UK email: {reshma,lawson}@transglobal.co.uk

(ARs). An itemset is said to be frequent if its support exceeds some user defined support thresholds.

In Temporal ARM the attributes in the data are time stamped in some way. One category of Temporal ARM is known as Emerging and Jumping Pattern (JEP) mining (Dong and Li, 1999). An Emerging Pattern (EP) is usually defined as an itemset whose support increases over time according to some "change ratio" threshold. A Jumping Pattern (JP) is an itemset whose support changes much more rapidly than that for an EP. EPs and JPs are distinguished by their change ratio threshold and thus for many purposes can be considered to be synonymous. The discovery of JEPs entails a significant computational overhead due to the large number of itemsets that must be identified to facilitate comparison. To avoid this overhead most JEP mining approaches concentrate on a subset of the potential frequent itemsets such as the set of *maximal* itemsets; that is, the itemsets that show the greatest negative or positive change in value. The computational cost of comparing all items sets across all time stamps tends to render this approach to be, computationally prohibitively, expensive. In this paper we present the Dual Support Apriori Temporal (DSAT) algorithm, an approach to JEP mining that utilizes the entire "data space", but avoids the computational overhead, by using a sliding window mechanism.

The main novelty of the proposed approach is in the adoption of a dual support mechanism in which each itemset holds two support counts, called $supp_1$ and $supp_2$, that benefits: (i) efficient memory utilization, (ii) few IO overheads and (iii) less computation cost. Under the dual support framework $supp_1$ holds the support counts of itemsets in the "oldest" data segment that disappears whenever the window "slides" and $supp_2$ holds support counts for itemsets in the overlap between two windows and the recently added data segment as shown in Figure 1.

The dual support mechanism utilises the already discovered frequent itemsets from the previous windows and avoid re-calculating support counts for all itemsets that exists in the overlapped datasets between two windows, this is illustrated in section 4. Moreover it only required databases access for the most resent segment, thus less IO operations and less memory utilization.

The paper is organized as follows. In section 2 the related work and the problem domain are described in more detail. Section 3 provides a sequence of definitions. To facilitate understanding of the dual support framework a worked example is presented in Section 4. The DSAT algorithm, in its entirety, is then presented in section 5 and evaluated in section 6.

2 Related Work

There are many commercial applications that produce significant amounts of temporal data collected and stored electronically on a daily bases, examples include: web server logs; supermarket transactional data, and network traffic. There are many studies directed at the efficient application of temporal forms of ARM to time stamped data sets (Li and Lee, 2009; Lee et al. 2003; Chang and Lee, 2004). The main issue in temporal ARM is the high computational cost of the processing of the data so as to take account of the temporal dimension. Jiang and Gruenwald (Jiang, 2006) compare the temporal data processing models found in temporal ARM, such as: Landmark, Damped and Sliding Windows and their usage depending on the application area. Jiang and Gruenwald also discus issues related to memory management, data structures to store frequent sets and various modified ARM algorithms for temporal ARM.

One category of temporal ARM, as noted in Section 1 above, is Jumping and Emerging Patterns (JEPs) mining as first proposed by (Dong and Li, 1999). In common with many subsequent JEP algorithms Dong and Li compared maximal itemsets generated using a Max-Miner style of algorithm (Bayardo, 1998). A maximal itemset is a frequent (supported) itemset whose supersets are all infrequent (i.e. their support value is below the user specified support threshold). By identifying only maximal itemsets all frequent itemsets can be found by virtue of the DC property (although only the precise support values for the maximal sets are known). The advantage of identifying only maximal itemsets is one of computational efficiency. This is particularly important in the context of JEP mining because of the large number of itemsets that must be identified across time stamps. In addition, to facilitate comparison of itemsets, a low support threshold must also be used hence adding to the magnitude of the problem. However, the maximal frequent set approach does not guarantee the identification of all JEP.

Many JEP mining algorithms have been reported in the literature (Imberman and Tansel, 2004; Bailey et al. 2002; Rioult, 2004; Grandinetti et al. 2005; Tseng et al. 2006). Most of these algorithms adopt a maximal frequents itemset approach as first proposed by Dong and Li (1999). For example Tseng et al. (Tseng, 2006) extends the work of Dong and Li (1999) and proposed EFI-Mine (Emerging Frequent Itemsets) algorithm that discovers JEPs using the technique similar to data streams . The main issue with these existing approaches to JEP mining is that they tend to use only maximal frequent itemsets to identify JEPs. Thus, although efficient, they do not guarantee to find all JEPs.

Our proposed DSAT algorithm differs from the previous work in that we consider all identified frequent itemsets across time stamps. The computational overhead that is normally associated with this approach is avoided by using the dual support concept together with a sliding window approach that requires less memory and data access than would be required otherwise.

3 Preliminaries

In this section a number of formal definitions are presented to facilitate understanding of the rest of the paper. Firstly it is necessary to define the concept of classical ARM. Given a set of items $I = \{i_1, i_2, ..., i_m\}$ and a database of transactions $D = \{t_1, t_2, ..., t_n\}$ where $t_i = \{I_{i1}, I_{i2}, ..., I_{ip}\}$, $p \leq m$ and $I_{ij} \in I$; where $X \subseteq I$ with $K = |X|$ is a k-itemset or simply an itemset. Let a database D be a multi-set of subsets of I as shown in table 1. Each $T \in D$ supports an itemset $X \subseteq I$ if $X \subseteq T$ holds. An AR is an expression $X \Rightarrow Y$, where X, Y are itemsets and $X \cap Y = \phi$ holds. Number of transactions T supporting an item X w.r.t D is called the *support* of X, $Supp(X) = |\{T \in D \mid X \subseteq T\}| / |D|$. The strength or *confidence* for an association rule X => Y is the ratio of the number of transactions that contain $X \cup Y$ to the number of transactions that contain X, Conf (X → Y) = Supp (X U Y)/ Supp (X).

Emerging patterns, as noted above, are itemsets whose support increases significantly from one data set to another i.e. from w_i to w_{i+1}. An itemset X is called an emerging pattern if the $supp(X) \geq \sigma$ and $GR(X) \geq \delta$ where σ *and* δ are user specified support and growth rate thresholds respectively. *Jumping patterns* are the specialized case of emerging patterns where $GR(X) \rightarrow \infty$ and this is when $supp(X, D_1) \rightarrow 0$. The growth rate of an itemset X from D_1 to D_2 is defined as:

$$GrowthRate(X) = \begin{cases} 0 & if\,(supp(X, D_1) = 0 \; and \; supp(X, D_2) = 0) \\ \infty & if\,(supp(X, D_1) = 0 \; and \; supp(X, D_2) \neq 0) \\ \dfrac{supp(X, D_2)}{supp(X, D_1)} & otherwise \end{cases} \quad \textbf{(1)}$$

Time series databases contain data collected over a period of time and can be processed by a sliding window. Each window w_i represents some sequence of time stamped data $w_i = \{t_1, t_2, ...t_w\}$ where t_i is a single time stamp. The amount of data contained in the window may therefore very as the window is progressed along the time series.

Table 1. Super market database

Tid	Items		Tid	Items	
T_1	A, B, C		T_5	A, B, C, D	
T_2	B, C, D, E	D_1	T_6	A, B, C, D	D_2
T_3	B, C, E		T_7	A, B, C	
T_4	B, E		T_8	A, D, E	

Suppose we are given a retail dataset covering two days, D_1 and D_2 respectively. The growth rate of an itemset X from D_1 to D_2 is denoted as $GR(X, D_i, D_{i+1})$ and is defined as in (Dong and Li, 1999):

$$GR(X) = \frac{supp(X, D_2)}{supp(X, D_1)} \qquad \textbf{(2)}$$

As data in different windows is un-evenly distributed, it is necessary to correct the above equation by multiplying it with $| D_1 | / | D_2 |$, otherwise a bias will favor the EPs process for the dataset with large number of transactions as mentioned in (Cremilleux et al. 2003). Thus equation 1 will become:

$$GR(X) = \frac{supp(X, D_2)}{supp(X, D_1)} \times \frac{D_1}{D_2} \qquad \textbf{(3)}$$

Given $\sigma > 0$ as support threshold and $\delta > 1$ as growth rate threshold, a frequent pattern X is said to be an emerging pattern from D_1 to D_2 if $GR(X) \geq \delta$. For the data in table 1, if we set $\delta = 3$, then ABC is an EP and ABCD is a JP from D_1 to D_2 because $supp(ABC, D_1) = 1$ and $supp(ABC, D_2) = 3$ and by using equation 2 the $GR(ABC) \geq \delta$, similarly $supp(ABCD, D_1) = 0$ and $supp(ABCD, D_2) = 2$ thus $GR(ABCD) \rightarrow \infty$. But $supp(BCD, D_1) = 1$ and $supp(BCD, D_2) = 2$ and by using equation 2 the $GR(BCD) \leq \delta$ thus neither JP nor EP.

In the proposed dual support framework for discovering JEPs, transactions in each window are logically partitioned into three segments as $w_i = \{p_1, p_2, p_3\}$ except w_1 because it only consists $w_1 = \{p_1, p_2\}$ where $p_i \leq D_i$ as shown in figure 1.

p_1 holds data that disappears in the next increment, p_2 holds data that is overlapped between two windows w_i and w_{i+1} i.e. $p_2 = D_i \cap D_{i+1}$ and p_3 consist of data that is added to w_{i+1} after the increment or window slide as shown in the figure 1, where $p_1 = \{t_2\}$, $p_2 = \{t_3, t_4, t_5\}$ and $p_3 = \{t_6\}$ for w_2.

The itemset support counts are denoted as: S_{i1} *and* S_{i2}, where $i = w_i$. For the first window support count of itemsets in p_1 are recorded into S_{11}, i.e. $S_{11} = | p_1 |$, and support counts of itemsets in p_2 are recorded into S_{12}, $S_{12} = | p_2 |$. After the window is incremented, from w_1 to w_2, S_{11} is set to zero because the support it holds does not contribute in the next window. S_{12} from the first window is copied into S_{22} in w_2, and the support count of itemsets from w_1 is decremented by p_1. Itemsets from p_3 are generated using S_{22} and then integrated into the already generated itemsets from w_1. Also, S_{22} of any itemset from w_1 is incremented if it exists in p_3.

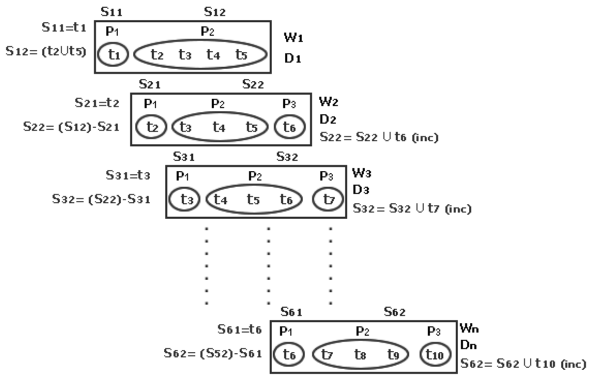


Figure 1 JEPs with dual support framework

The dual support framework therefore uses less memory, features limited IO operations and fewer computations (by utilising the already discovered frequent sets from previous windows), and avoids re-calculating support counts for item-sets that exists between overlapped windows.

4 Dual Support Framework Example

In this section we present an example to illustrate the proposed dual support framework using a sliding windows technique. Table 2 shows five datasets D_1 to D_5 for days starting from 1 to 5. We used days for simplicity but in real applications this could be any temporal interval. For this application we set window size to 3, window slide to 1, support threshold to 25% and growth rate threshold to 2.

For w_1, data sets $D_{1,2,3}$ will be used because $|w|$ is set to 3 as shown in figure 2a. The supports for an itemset is calculated in such a way that $supp_1$ under w_1 holds the number of occurrences of an itemset for D_1 and $supp_2$ holds the number of occurrences of an itemset for the rest of the datasets in w_1. Applying DSAT algorithm following 2-frequent itemsets are generated {(A, B), (A, C), (A, D), (B, C), (B, D), (C, D)}.

Table 2. Example transitional data for 5 weeks

Tid	D_1	D_2	D_3	D_4	D_5
	Day 1	*Day 2*	*Day 3*	*Day 4*	*Day 5*
T_1	A B D	C D	A B C D	C D	A D
T_2	C D	A B E	C D E	A B C E	B C E
T_3	B C	A C D	A C	A C	A C D E
T_4	B D	B C D	A E	A B C D	C

After generating frequent itemsets the window slides (w_2); D_4 is added and D_1 is removed as shows in figure 2b. Itemsets generated in w_1 are cloned in w_2 to avoid itemset re-generation. $supp_1$ for the cloned itemsets in w_2 is set to zero as it no longer contributes to the current window.

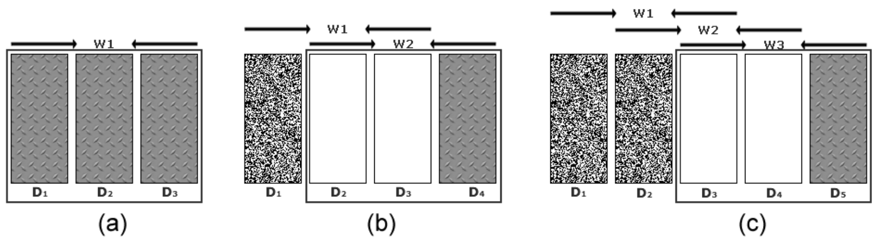


Figure 2 Sliding Windows for table 2 datasets

Frequent itemsets are generated using only D_4 and $supp_2$, consequently the rest of the itemsets are adjusted. $supp_1$ is then calculated for all generated itemsets in w_2 from D_2 and is then subtracted from $supp_2$ so that the accumulative support $supp = supp_1 \cup supp_2$ of itemsets gives the total support count. 2-Frequent itemsets generated for w_2 are {(A, B), (A, C), (A, D), (A, E), (B, C), (B, D), (C, D)}. {A, E} and {A, C} are the discovered JEPs from the frequent sets for w_1 and w_2. This is shown in equation 3. The same procedure is repeated for w_3 as shows in figure 2 where the only JEP discovered from w_2 and w_3 is {C, E}.

Note that if we have a high support threshold there is an option to eliminate an emerging pattern in w_{i+1}. This means that a potential frequent itemset in w_i is no longer considered. If we do need to consider this itemset then a lower support threshold can be used without adversely affecting the efficiency and memory of the system. This is demonstrated in the experiment section.

5 The DSAT Algorithm

Our DSAT algorithm was developed using tree data structures, in a fashion similar to the Apriori algorithm (Agrawal and Srikant, 1994), and comprises of two major steps:

1. Apply Apriori to produce a set of frequent itemsets using the sliding window approach.
2. Process and generate a set of JEPs such that the interestingness threshold (Growth Rate) is above some user specified threshold.

Steps involved in the DSAT algorithm are as follows:

For the initial window

1. Load the initial dataset $D_{1-\backslash w\backslash}$ into the memory (w_i).
2. Apply ARM algorithm using sliding windows.
3. Use dual supports for each itemset, $supp_1$ for p_1, $supp_2$ for p_2.
4. Generate frequent sets.
5. Slide window and carry forward all the frequent sets from $w_i \rightarrow w_{i+1}$.

For the sliding window

6. Clone the frequent sets from window w_i to incremented window w_{i+1}.
7. Decrement $supp_1$ of itemsets using p_1.
8. Update itemsets' $supp_2$ as described in section 4.
9. Load only the incremented transactions p_3 into memory.

10. Calculate the $supp_2$ for all the existing itemsets and generate any new itemsets by only considering the incremented time stamp p_3.
11. Calculate the growth rate of itemsets using both windows w_i and w_{i+1}.
12. Those itemsets with growth rate $\geq$ the threshold are emerging patterns and the itemsets those support approaches to zero in w_i and have support $\geq$ specified jumping threshold in w_{i+1} are the jumping patterns
13. Store JEPs' for the current window w_{i+1}.
14. Go to step 5.

6 Experimental Evaluation

In this section the proposed DSAT algorithm is evaluated with different datasets in order to asses the quality, efficiency and effectiveness of our approach. In the experiments, synthetic and real datasets (with binary and quantitative attributes) are used.

6.1 Datasets

Table 3 overviews the evaluation datasets. It should be noted that the datasets contains both sparse and dense data, since most AR discovery algorithms were designed for these types of problems.

Table 3. Real and Synthetic datasets used for experiments

Dataset	Type	Time Duration	Number of Transactions	Distinct Items	Max. Trans. Size
Server Logs (SL)	Real	11/04/08–15/04/09	49,577	1,372	16
Point of Sale (PS)	Real	28/09/07–27/09/08	92,685	3,736	19
Transglobal (TG)	Real	12/09/07–08/05/09	8,000	3,000	5
T10I4D100K (TDK	Synthetic	Not specified	100,000	1,000	29

The first three datasets comprising transactions recorded for almost one year. All the datasets are time stamped and partitioned, except T1014D100K, so that the number of transactions varies in each partition. T1014D100K is divided into ten equal partitions of size 10K for experimental purpose. The Transglobal dataset contained quantitative attributes and we discretised the quantitative attributes to binary ones according to the technique proposed in (Sulaiman et al. 2009). All the raw datasets were cleaned and filtered to make them suitable for temporal ARM analysis.

6.1.1 Data Pre-Processing

The raw data was pre-processed by cleaning and filtering it to make it suitable for temporal ARM analysis. The server logs data included Apache server log files comprising of two hundred and thirty files all together with 7,688,244 transactions. The log files were all time stamped and contain plenty of information not required for our analysis. Filtration algorithms were developed to extract only the relevant information. After cleaning and filtering the log files, only 49,577 useful transactions out of 7,688,244 were extracted from April 2008 till April 2009. The Point of sale data (supplied by a news agent) comprises of 174,632 transactions and after cleaning and filtration this was reduced to 92,685 customer transactions from September 2007 till September 2008. The Transglobal data, supplied by a freight forwarding enterprise, contained categorical and quantitative attributes time stamped from September 2007 till May 2009. In this case the data was discretised and transformed into a Boolean format. The Transglobal data, unlike the other two data sets, also required sequencing in ascending order according to date (time stamp).

6.2 Comparisons with Apriori

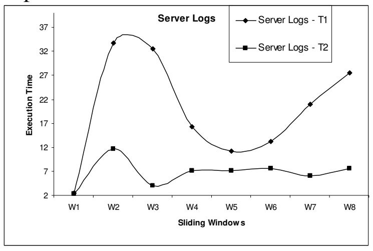
Two sets of experiments were conducted to demonstrate: (i) the efficiency of the proposed approach and (ii) the temporal effect on the itemsets (JEPs) as an outcome of the ARM analysis. The experiments demonstrated that the proposed approach was a useful form of trend analysis. All the experiments were conducted on a P4; 1GB, 3GHz machine with windows XP installed using jdk1.4.2.

6.2.1 DSAT Performance

To compare the performance of DSAT we modified the classical Apriori algorithm to deal with temporal data in a conventional manner i.e. process each sliding window and compare it with the proposed DSAT algorithm. The comparison illustrated that DSAT outperformed the Apriori naïve approach for temporal ARM.

6.2.2 Effect of Varying Data Size and Support Threshold

Figures 3, 4, 5 and 6 show the execution time for Apriori and the DSAT algorithms on four real and synthetic datasets with quantitative and binary attributes. In the figures T1 represents the execution time for the modified classical Apriori ARM and T2 represents the execution time for the proposed DSAT algorithm.



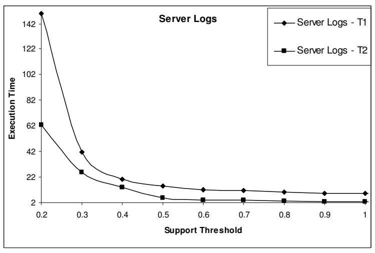
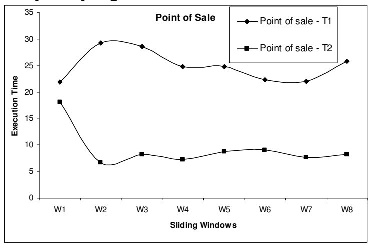


Figure 3a Execution time for Server Log data by varying windows

Figure 3b Execution time for Server Log data varying support thresholds



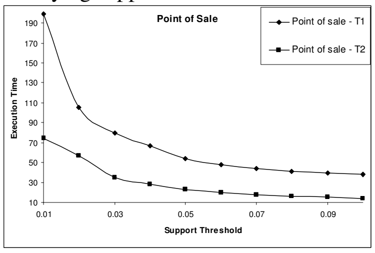
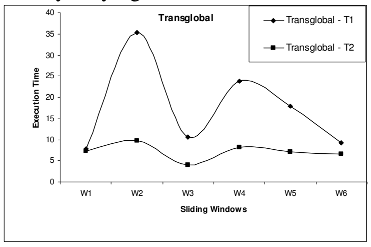


Figure 4a Execution time for Point of Sale data by varying windows

Figure 4b Execution time for Point of Sale data by varying support thresholds



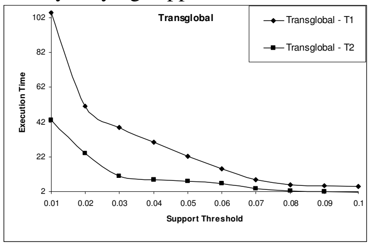
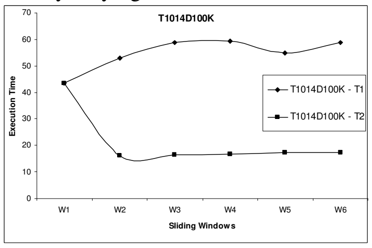


Figure 5a Execution time for Transglobal data by varying windows

Figure 5b Execution time for Transglobal data by varying support thresholds



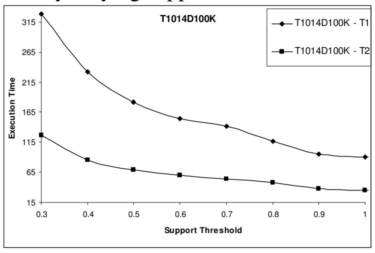


Figure 6a Execution time for Synthetic data by varying windows

Figure 6b Execution time for Synthetic data by varying support thresholds

For figures 3, 4, 5 and 6 (a), the x-axis represents the varying sliding windows and y-axis represent the execution time in seconds for the algorithms for different sliding windows. For figures 3, 4, 5 and 6(b), the x-axis represents the percentage support threshold for the datasets and y-axis represents the cumulative execution time in seconds for the support thresholds. Support thresholds were selected so that the Apriori algorithm could generate frequent itemsets within the given memory constraints so that execution time statistics could be obtained.

Two figures, for each dataset, are displayed in order to show that the DSAT outperforms the naïve Apriori approach not only on the cumulative execution time but also for each sliding window, regardless of various data sizes.

The figures demonstrate that DSAT outperforms the modified Apriori ARM algorithm because DSAT uses already generated frequent itemsets from the previous windows and thus only needs to generate frequent itemsets for the "most recent" transactions. The execution times in figures (a) are not linear because of the varying data sizes in different windows, but near linear in figures (b) as the accumulative windows execution time varies with the support thresholds. The result also displays the ARM property that by increasing support the execution time decreases and vice-versa, algorithm completion time increases (Sulaiman et al. 2007).

Moreover the classical ARM algorithm utilises more memory, compared to DSAT, because the use of very low support thresholds leads to the generation of a high number of frequent itemsets. In contrast, DSAT runs more effectively (because DSAT utilises already generated frequent sets from the previous windows, updates their support count for the current window, and only generate the frequent sets from the incremented time stamp as illustrated in Section 4).

6.3 Temporal Effects of Varying Windows and Threshold

The experiments described in this section show how the varying sliding windows affect the overall ARM analysis in discovering JEPs. Figures 7, 8 and 9 show the number of Emerging Patterns discovered (figures a), Jumping Patterns (figures b) and frequent itemsets (figures c) respectively for three different real datasets. The JEPs and the frequent itemsets in the figures are generated by varying support thresholds. As before the support thresholds were selected so that the classical Apriori algorithm would be able to generate large numbers of JEPs so that statistical comparison data could be obtained.

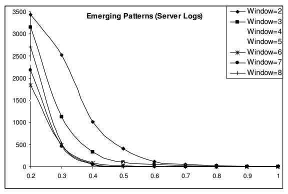


Figure 7a

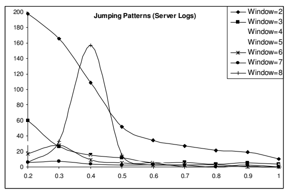


Figure 7b

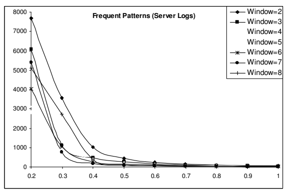


Figure 7c

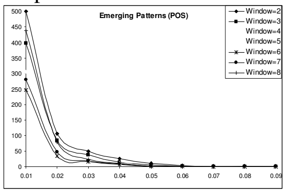


Figure 8a

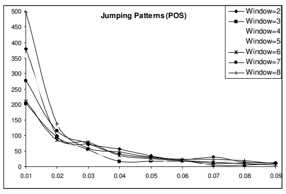


Figure 8b

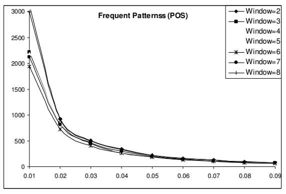


Figure 8c

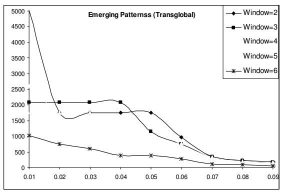


Figure 9a

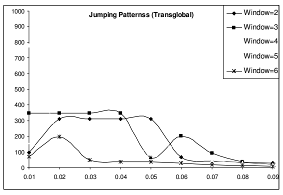


Figure 9b

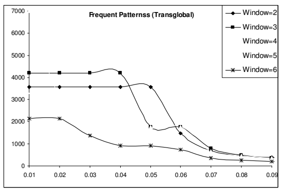


Figure 9c

In figures 7, 8 and 9 the x-axis represents various support thresholds and the y-axis the number of EPs, JPs and Frequent items in figures a, b and c respectively. Each curve represents a window from 1 to 8.

From the figures, the numbers of JEPs generated are not linear and there are abrupt differences in the numbers generated due to the number of transactions varying in each window and that the number of transactions changes as the window slides. However there is some linearity for frequent items (Figures 7c, 8c and 9c).

It can be seen from the figures that more JEPs are generated at lower support thresholds as compared to higher ones where in some cases the number approaches zero. The major issue in finding the JEPs is that they are normally generated at low support thresholds because an itemset could qualify as a JEP once its support at w_{i-1} is low as compared to w_i. A JEP can only be discovered once it becomes frequent, or at least is generated in the previous window.

6.4 Trend Analysis

The proposed approach can be usefully employed in trend ARM analysis where data is gathered for fixed or continuous time stamps. For example, the support of an itemset can be monitored over a period of time and it can help end users determine the causes any increment or decrement.

For example, figure 10 shows the support for eight different itemsets, i.e. users' clicks (hits), in web log data that has been monitored for almost a year. Each curve represents an individual itemset in terms of its support for one year from April 2008 till April 2009. The x-axis in the figure represents time in terms of sliding windows and the y-axis the number of time users hits the web pages (itemset).

The website was launched in March 2008, and it can be seen from the figure that the support for the itemsets is low at start up. However, the number of hits increased as more users visited the website, thus increasing the itemsets' support count. From the figure, the itemset $\{318, 375\}$ has support zero in the first two windows and it emerges as a Jumping pattern at the third window slide; the support kept increasing till the seventh window until it once again disappear in the eighth window. That is because initially page 375 did not exist in the website but was later added (as evidenced by the "jump" in support in the third window). However, later (in the eighth window) it was again removed from the site and the support returns to zero.

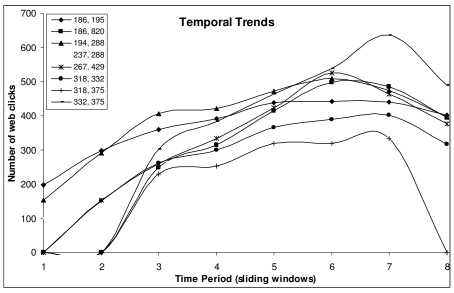


Figure 10 Varying itemsets supports over time

The JEP technique described in this paper is relevant for trend analysis because it not only explicitly highlight trends, but also gives an indication about what factors are influential in boosting or decreasing the relationship between items.

7 Conclusions and Future Work

We have presented a novel approach to efficiently extract JEPs in temporal data by using a sliding window coupled with a dual support mechanism. The advantages of the framework are less memory utilization, limited IO and fewer computations by utilising the previously computed frequent sets. This avoids re-calculation of support counts that already exist in between overlapped windows.

The approach has been realized in the form of the DSAT algorithm. The evaluation of this algorithm has produced some very encouraging results. Future work will involve enhancing the efficiency of the algorithm by adopting a T-Tree data structure (Coenen et al. 2004) that uses indexing to further enhance the computational efficiency. Furthermore larger datasets and parallelisation of the DSAT algorithm will be investigated. We anticipate that a stream processing technique (Kapasi et al. 2003) would be particularly suitable for this purpose.

A Sliding Windows based Dual Support Framework for Discovering Emerging Trends from Temporal Data

Acknowledgement: we would like to thanks Dr. Maybin Muyeba for his valuable feedback and contribution in the research. Also, we would like to thank Michelle Verity and the team at LearnHigher, Mr. Ikram proprietor of the News Agency for providing the Server Logs and Point of Sale data respectively.

References

1. Agrawal, R. & Srikant, R., 1994. Fast algorithms for mining association rules. In Proc. 20th Int. Conf. Very Large Data Bases, VLDB. 487-499.
2. Li, H.F. & Lee, S.Y., 2009. Mining frequent itemsets over data streams using efficient window sliding techniques. Expert Systems with Applications, 36(2P1), 1466-1477.
3. Lee, C.H., Chen, M.S. & Lin, C.R., 2003. Progressive partition miner: An efficient algorithm for mining general temporal association rules. IEEE Transactions on Knowledge and Data Engineering, 1004-1017.
4. Chang, J.H. & Lee, W.S., 2004. A sliding window method for finding recently frequent itemsets over online data streams. Journal of Information Science and Engineering, 20(4), 753-762.
5. Jiang, N., 2006. Research issues in data stream association rule mining. ACM Sigmod Record, 35(1), 14-19.
6. Imberman, S.P. and Tansel, A.U. and Pacuit, E., 2004 An Efficient Method For Finding Emerging Frequent Itemsets,3rd International Workshop on Mining Temporal and Sequential Data, 112-121.
7. Bailey, J., Manoukian, T. & Ramamohanarao, K., 2002. Fast algorithms for mining emerging patterns. Lecture notes in computer science, 39-50.
8. Dong, G. & Li, J., 1999. Efficient mining of emerging patterns: Discovering trends and differences. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM New York, NY, USA, pp. 43-52.
9. Rioult, F., Mining strong emerging patterns in wide SAGE data. In Proceedings of the ECML/PKDD Discovery Challenge Workshop. Citeseer, pp. 484-487.
10. Grandinetti, W.M., Chesnevar, C.I. & Falappa, M.A., 2005. Enhanced Approximation of the Emerging Pattern Space using an Incremental Approach, Proceedings of VII Workshop of Researchers in Computer Sciences, Argentine, 263-267
11. Coenen, F., Leng, P. & Ahmed, S., 2004. Data structure for association rule mining: T-trees and P-trees. IEEE Transactions on Knowledge and Data Engineering, 774-778.
12. M. Sulaiman Khan, Muyeba, M., Tjortjis, C. & Coenen, F., 2007. An Effective Fuzzy Healthy Association Rule Mining Algorithm (FHARM). In Proc. 7th Annual Workshop on Computational Intelligence UKCI 2007.
13. (TDK) IBM Synthetic Data Generator, http://www.almaden.ibm.com/software/quest/resources/index.html
14. (SL) Server Logs data set is the courtesy of LearnHigher: http://www.learnhigher.ac.uk
15. (POS) Point of Sale data is provided by a News Agent/Grocery Store in Walsall
16. (TG) Freight forwarding enterprise data is provided by Transglobal Express Service http://www.transglobalexpress.co.uk
17. Cremilleux, B., Soulet, A. & Rioult, F., 2003. Mining the strongest emerging patterns characterizing patients affected by diseases due to atherosclerosis. In proceedings of the workshop Discovery Challenge, PKDD'03. 59-70.
18. Tseng, V. S., Chu, C.J. & Tyne Liang, 2006. An Efficient Method for Mining Temporal Emerging Itemsets From Data Streams. International Computer Symposium, Workshop on Software Engineering, Databases and Knowledge Discovery.

19. Bayardo Jr, R.J., 1998. Efficiently mining long patterns from databases. ACM SIGMOD Record, 27(2), 85-93.
20. Kapasi, U.J. et al., 2003. Programmable stream processors. Computer, 54-62.
21. M. Sulaiman Khan, Muyeba, M. & Coenen, F., 2009. Effective Mining of Weighted Fuzzy Association Rules, Rare Association Rule Mining and Knowledge Discovery: Technologies for Infrequent and Critical Event Detection, Advances in Data Warehousing and Mining (ADWM) Book Series, IGI Global. ISBN: 978-1-60566-754-6, 47-64.