International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 2, Issue 5 (March 2013) PP: 49-53

Top-K Influencing Itemset on Data Streams - A Comparative Study on Simple Sliding Window and Sliced Sliding Window

G. Sandhya[1], S. Kousalya Devi[2], C. Kumar Charlie Paul[3]

[1]*P.G. Scholar, Department of CSE, A. S. L. Pauls College of Engineering and Technology, India*
[2]*Associate Professor, Department of CSE, A. S. L. Pauls College of Engineering and Technology, India*
[3]*Principal, A. S. L. Pauls College of Engineering and Technology, India*

Abstract: *The notion of sliding window is to identify the vigorous itemset in which the vigorous itemset refers to the newest itemset that comes in recently. The simple sliding window permits the newest itemset to come inside and also allows the oldest itemset to move outside. The sliced sliding window splits the sliding window into slices. Each slice in the sliding window permits the newest itemset to come inside and also allows the oldest itemset to move outside. The simple sliding window uses the Brute Force Approach and the Event Processing Approach for determining Top-K influencing itemset on data streams. The sliced sliding window uses the Improved Brute Force Approach and the Improved Event Processing Approach for determining Top-K influencing itemset on data streams. The sliding window identifies Top-K influencing itemset on data streams based on preferences and multi-dimensional analysis. The paper compares the CPU time, accuracy as well as efficiency of both the sliding windows. Experimental evaluation using the real and synthetic dataset shows that the sliced sliding window is accurate. Also, the sliced sliding window reduces the CPU time.*

Keywords: *Event processing algorithm, Preference based query, Sliced sliding window, Simple sliding window, Top-K influencing itemset.*

I. INTRODUCTION

The Top-K influencing query identifies the Top-k itemset which influences other itemset using some preferences. The query maintains a user defined ranking function, which allocates a value to each itemset and has scaling invariance property which means that if scaling applies to dimension values then, the result will be the same. The sliced sliding window supports the count based sliding window. In count based sliding window [7], the number of vigorous itemset remains constant. The moving out time of some itemset will be equal to the incoming time of the same amount of other itemset. This method serves the purpose of book-keeping, which deletes a moving out itemset and introduces a new itemset. Consequently, it simply calculates and updates the scores of the Top-K influencing itemset.

The work carries a thorough experimental estimation based on real datasets and synthetic datasets. This provides evidence regarding the CPU time, accuracy and efficiency of the Top-K influencing itemset. Remaining of the paper organizes as follows. Section 2 illustrates the work linked with the problem. Section 3 compares the simple sliding window and the sliced sliding window. Both the sliding windows identify the Top-K influencing itemset based on the Brute Force Approach and the Event Processing Approach. Section 4 presents the experimental estimation of based on some datasets. Section 5 concludes the work which carries out throughout the paper.

II. RELATED WORK

Preferences [6] take place in Game Theory, Computational Geometry and others disciplines. The Top-K Scrutinizing algorithm [7] finds the Top-K dominating itemset based on the Bucket In-Out method. The batch counting [8] computes the scores of skyline itemset in batch instead of iteratively applying separate range queries. This involves a light weight technique [8] to obtain the upper bound score of non-leaf entries at low cost and the lazy counting technique to delay the counting of itemset for the purpose of forming better groups for batch counting. To be efficient, the tree traverses with a cautiously designed priority order aiming at minimizing the I/O cost. According to [6], the moving outside time of an itemset will be equal to the sum of the coming inside time of an itemset and the number of vigorous itemset. The entering and running out time instances characterize the sequence of itemset formed by the data.

Reverse Top-K queries [4] are necessary for the manufacturers to access the potential market and impact of their products based on competition. A grid-based indexing scheme [5] facilitates an efficient search and update operations, evading costly re-organization costs. The method uses an adaptive grid which separates equally the tuples in each dimension. The schema formulates and tackles the problem of Probabilistic Top-K Dominating (PTD) query [3] in the context of uncertain databases. The Sliding Window Combinatorial

Approximation (SWCA) [2] establishes frequent itemset over sliding windows in data streams. The Min Top-K algorithm [1] maintains a Minimal Top-K candidate set (MTK) which determines the Top-K value and thus eliminates the need for re-calculation.

III. COMPARISON OF SIMPLE SLIDING WINDOW AND SLICED SLIDING WINDOW

The comparison of the simple sliding window and the sliced sliding window for identifying the Top-K influencing itemset based on the Brute Force Approach and the Event Processing Approach takes place as follows.

1. Simple Sliding Window

The simple sliding window adopts the count based sliding window. The simple sliding window identifies the Top-K influencing itemset based on the Brute Force Approach and the Event Processing Approach.

1.1 Brute Force Approach

The Brute Force Approach takes place in a count based sliding window. In a count based sliding window, if a newest itemset comes inside the sliding window the oldest itemset will move outside the sliding window and the time progresses. Hence, the moving out time of the itemset will be equal to the sum of the moving out time of the itemset and the number of vigorous itemset. Consequently, the checks for finding the influencing itemset will be large. The simple sliding window forces the oldest itemset to move outside the sliding window whenever the newest itemset comes inside. Therefore, the approach is termed as the Brute Force Approach.

1.2 Event Processing Approach

The Event Processing Approach includes an event processing time, event generation time and score of an itemset at the event generation time. Each update applies a chain of operation. Initially, the approach updates the book-keeping structure and deletes the itemset which moves outside and enters the itemset which comes inside. Furthermore, the approach updates the scores of the Top-K influencing itemset. After that, the approach processes the entering itemset to determine whether they should be a part of Top-K. The approach then tries to locate an itemset such that, it influences the itemset which comes inside and it is not a part of Top-K. If such an itemset does not exist or the upper bound of the score is larger than or equal to the top score, then the approach calculates the itemset which comes inside from scratch. If score of the itemset which comes inside is greater than the top score, then it inserts the itemset which comes inside in the Top-K. Otherwise, the approach produces an event for the itemset which comes inside by using three parameters such as the id j of the itemset, its score and *now*, which is the current time occurrence. First, it calculates the event processing time.

Subsequently, it ensures whether time is greater than or equal to *now*. If so, it inserts the event into the priority queue. Examine that, the event processing time is less than *now* as if the parameter score is larger than top score. Thus, it always generates an event, since either the upper bound or the exact score is less than the top score. At last, it processes all the events with the event processing time equal to *now*. Then, the approach tries to re-calculate the event processing time value by using the upper bound of the number of itemset influenced by the i^{th} itemset for an event. If the event again inserts into the event priority queue, then consider the subsequent event. If not, if the upper bound assessment is poor, the calculated event time is likely to be less than *now*. In such a case, the approach proceeds with the exact score calculation of the number of itemset influenced by the i^{th} itemset and either inserts the i^{th} itemset in the Top-K or re-calculates the event time based on the exact score of the i^{th} itemset. Subsequently, it controls the moving out Top-K influencing itemset. If a Top-K itemset expires in *now* and no updates in top score then, there is no need to try to calculate event times. In this case, it sets the top score to -1 to forcibly insert other itemset in Top-K. Afterward, it calculates the score of the itemset of the initially examined event and inserts the itemset in Top-K. Eventually, it tries to re-calculate the event time of the remaining events.

2. Sliced Sliding Window

The sliced sliding window splits the sliding window into slices. The sliced sliding window also adopts the count based sliding window. The sliding window gets split into five slices which are equal in size. The sliced sliding window identifies the Top-K influencing itemset based on the following Improved Brute Force Approach and Improved Event Processing Approach.

2.1 Improved Brute Force Approach

The Improved Brute Force Approach also takes place in a count based sliding window. Each slice of the sliding window carries out the Brute Force Approach and is known as the Improved Brute Force Approach.

In each slice, if a newest itemset comes inside the oldest itemset will move outside and the time progresses. Hence, the moving out time of the itemset will be equal to the sum of the moving out time of the itemset and the number of vigorous itemset. Consequently, the checks for finding the influencing itemset will be large for each slice. Therefore, the sliced sliding window forces the oldest itemset to move outside each slice when the newest itemset comes inside.

2.2 Improved Event Processing Approach

The Improved Event Processing Approach also includes an event processing time, event generation time and score of an itemset at the event generation time. Each update applies a chain of operation. Initially, the approach updates the book-keeping structure and deletes the itemset which moves outside and enters the itemset which comes inside. Also, the approach updates the scores of the Top-K influencing itemset. After that, the approach processes the entering itemset to determine whether they should be a part of Top-K. Subsequently, the approach calls the Slice In-Out which tracks the in-out operation of each slice when an itemset comes into the window. Each time when a slice in-out operation occurs, the approach deletes the earliest transaction of the corresponding slice which contains the summary of transactions from the current window at each sliding. Hence, it is not essential to save the whole transactions within the current window in memory all along to support sliding window.

The sliding window is divided into five slices of equal size and each slice corresponds to a set of transactions. This is to reduce the overhead of memory throughout the storage of transactions. If there are more slices, then the overhead of memory arises during storage of the transactions. The approach then checks in which slice the itemset comes inside and the oldest itemset moves outside. Afterward the approach tries to locate an itemset in the slice such that, it influences the itemset which comes inside and it is not a part of Top-K. If such an itemset does not exist or the upper bound of the score is larger than or equal to the top score, then the approach calculates the itemset which comes inside from scratch. If score of the itemset which comes inside is greater than the top score, then it inserts the itemset which comes inside in the Top-K. Otherwise, the approach produces an event for the itemset which comes inside by using three parameters such as the id j of the itemset, its score and *now*, which is the current time occurrence.

First, it calculates the event processing time. Subsequently, it ensures whether time is greater than or equal to *now*. If so, it inserts the event into the priority queue. Examine that, the event processing time is less than *now* as if the parameter score is larger than top score. Thus, it always generates an event, since either the upper bound or the exact score is less than the top score. At last, it processes all the events with the event processing time equal to *now*. Then the approach tries to re-calculate the event processing time value by using the upper bound of the number of itemset influenced by the i^{th} itemset for an event. If the event again inserts into the event priority queue, then consider the subsequent event. If not, if the upper bound assessment is poor, it is possible the calculated event time to be less than *now*. In such a case, the approach proceeds with the exact score calculation of the number of itemset influenced by the i^{th} itemset and either inserts the i^{th} itemset in the Top-K or re-calculates the event time based on the exact score of the i^{th} itemset. Subsequently, it controls the moving out Top-K influencing itemset. If a Top-K itemset expires in *now* and no updates in top score then, there is no need to try to calculate event times. In this case, it sets the top score to -1 to forcibly insert other itemset in Top-K. Afterward, it calculates the score of the itemset of the initially examined event and inserts the itemset in Top-K. Eventually, it tries to re-calculate the event time of the remaining events.

IV. EXPERIMENTAL EVALUATION

All the approaches implement using C Sharp and conduct experiments on a Pentium at 3.0 GHz with 1 GB of Random Access Memory (RAM). This uses the real datasets and the synthetic datasets, named as the School-Buses datasets (http://www.chorochronos.org). The dataset undergoes a multidimensional analysis. The analysis of School-Buses dataset consists of 145 trajectories of 2 school buses collecting (and delivering) students around Athens metropolitan area in Greece for 108 different days. The structure of each record is as follows: {object id, trajectory id, date, time, latitude, longitude, x, y}.

Table.1: Performance Analysis of Brute Force Approach versus Improved Brute Force Approach.

Number of Iterations	CPU Time in Milliseconds	
	Brute Force Approach	Improved Brute Force Approach
1	677	498
2	568	560
3	536	477
4	568	505

Table.1 shows the performance analysis of the Brute Force Approach versus Improved Brute Force Approach. The approaches have the following CPU time during the first iteration. The Brute Force Approach takes 677 milliseconds while the Improved Brute Force Approach takes 498 milliseconds. In the second iteration, the Brute Force Approach takes 568 milliseconds whereas; the Improved Brute Force Approach takes 560 milliseconds. The third iteration has the CPU time as follows. The Brute Force Approach takes 536 milliseconds; while the Improved Brute Force Approach takes 477 milliseconds. During the fourth iteration, the Brute Force Approach takes 568 milliseconds; whereas; the Improved Brute Force Approach takes 505 milliseconds. The iteration prolongs till it retrieves the Top-K influencing itemset.

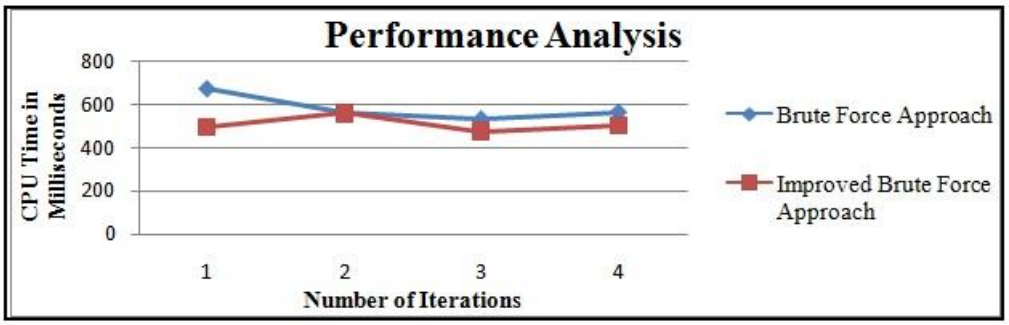

Fig.1: Iterations versus CPU time
(Brute Force Approach versus Improved Brute Force Approach)

Fig.1 plots the CPU time obtained from the Table.1 for each of the iterations in a graph. The graph takes the number of iterations in x-axis and the CPU time in y-axis. The solid diamond shape represents the CPU time of Brute Force Approach. The solid square shape represents the CPU time of Improved Brute Force Approach. From the Table.1 and Fig.1, it is clear that the Improved Brute Force Approach reduces the CPU time. Since, the Improved Brute Force Approach involves sliced sliding window it also reduces the memory overhead and the accuracy will be high. Accordingly, the sliced sliding window has the higher efficiency.

Table.2: Performance Analysis Event Processing Approach versus Improved Event Processing Approach.

Number of Iterations	CPU Time in Milliseconds	
	Event Processing Approach	Improved Event Processing Approach
1	313	188
2	391	203
3	332	276
4	303	219

Table.2 shows the performance analysis of the Event Processing Approach versus Improved Event Processing Approach. The approaches have the following CPU time during the first iteration. The Event Processing Approach takes 313 milliseconds while the Improved Event Processing Approach takes 188 milliseconds. In the second iteration, the Event Processing Approach takes 391 milliseconds whereas; the Improved Event Processing Approach takes 203 milliseconds. The third iteration has the CPU time as follows. The Event Processing Approach takes 332 milliseconds; while the Improved Event Processing Approach takes 276 milliseconds. During the fourth iteration, the Event Processing Approach takes 303 milliseconds; whereas; the Improved Event Processing Approach takes 219 milliseconds. The iteration continues till it retrieves the Top-K influencing itemset.

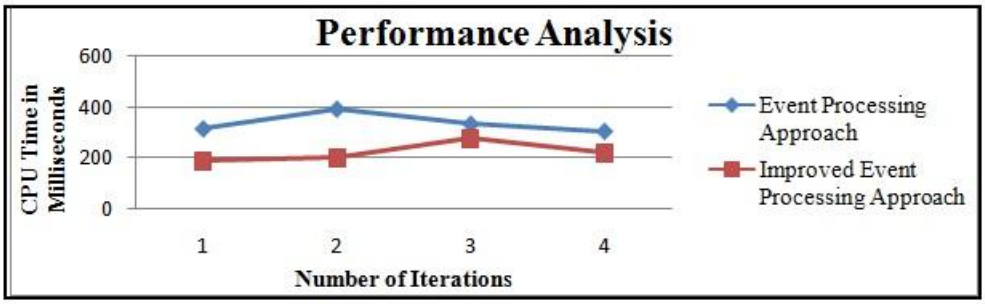

Fig.2: Iterations versus CPU time
(Event Processing Approach versus Improved Event Processing Approach)

Fig.2 plots the CPU time obtained from the Table.2 for each of the iterations in a graph. The graph takes the number of iterations in x-axis and the CPU time in y-axis. The solid diamond shape represents the CPU time of Event Processing Approach. The solid square shape represents the CPU time of Improved Event Processing Approach. From the Table.2 and Fig.2, it is clear that the Improved Event Processing Approach reduces the CPU time. Since, the Improved Event Processing Approach involves sliced sliding window it also reduces the memory overhead and the accuracy will be high. Thus, the sliced sliding window has the higher efficiency.

V. CONCLUSION

The sliding window identifies the vigorous itemset in which the newest itemset comes inside the sliding window recently. The simple sliding window permits the newest itemset to come inside and also allows the oldest itemset to move outside. The sliced sliding window splits the sliding window into slices. Each slice in the sliding window permits the newest itemset to come inside and also allows the oldest itemset to move outside. The simple sliding window uses the Brute Force Approach and the Event Processing Approach for determining Top-K influencing itemset on data streams. The sliced sliding window uses the Improved Brute Force Approach and the Improved Event Processing Approach for determining Top-K influencing itemset on data streams. The paper compares the CPU time, accuracy as well as efficiency of both the sliding window. Experimental evaluation using the real and synthetic dataset shows that the sliced sliding window is accurate. Also, the sliced sliding window reduces the CPU time.

REFERENCES

[1] Avani Shastri; Di Yang; Elke A. Rundensteiner; Mathew O. Ward. An Optimal Strategy for Monitoring Top-k Queries in Streaming Windows, *Proc. Int'l Conf. Extending Database Technology: Advances in Database Technology*, 2011.

[2] Chao-Wei Li; Kuen-Fang Jea. A Sliding-Window Based Adaptive Approximating Method to Discover Recent Frequent Itemsets from Data Streams, *Proc. IMECS*, 2010.

[3] Chen, L.; Lian, X. Top-k Dominating Queries in Uncertain Databases, *Proc. 12th Int'l Conf. Extending Database Technology: Advances in Database Technology*, 2009.

[4] Doulkeridis, C.; Kotidis, Y.; Nørvag, K.; Vlachou, A. Reverse Top-k Queries, *Proc. IEEE Int'l Conf. Data Engg.*, 2010.

[5] Kontaki, M.; Manolopoulos, Y.; Papadopoulos, A. N. Continuous Top-k Dominating Queries in Subspaces, *Proc. Panhellenic Conf. Informatics*, 2008.

[6] Kontaki, M.; Manolopoulos, Y.; Papadopoulos, A. N. Continuous Top-K Dominating Queries, *Proc. IEEE Transactions on Knowledge and Data Engineering*, 2012, *Vol. 24*, pp. 840-853.

[7] Kousalya Devi, S.; Kumar Charlie Paul, C.; Megala Devi, K.; Sandhya, G. Mining Precise Top-K Dominating Itemset from Data Streams, *Proc. IOSR Journal of Computer Engineering (IOSR-JCE)*, 2013, Vol. 9, Issue 1, pp. 13-17.

[8] Mamoulis, N.; Yiu, M. L. Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data, *Proc. Int'l Conf. Very Large Data Bases*, 2007, pp. 483-494.