Using Count Prediction Techniques for Mining Frequent Patterns in Transactional Data Streams

Chao-Wei Li and Kuen-Fang Jea
Department of Computer Science and Engineering
National Chung-Hsing University
Taichung 402, Taiwan, R.O.C.

Abstract—***We study the problem of mining frequent itemsets in dynamic data streams and consider the issue of concept drift. A count-prediction based algorithm is proposed, which estimates the counts of itemsets by predictive models to find frequent itemsets out. The predictive models are constructed based on the data in the data stream and serve as a description of the concept of the stream. If there is a concept drift in the stream, the description of the concept can be updated by reconstructing the predictive models. According to our experimental results, the proposed algorithm is efficient and has stable performance. Besides, using respective predictive models for count-predictive mining would preserve the quality of mining answers effectively (in terms of accuracy) against the change of the concept.***

Keywords - data mining; data streams; frequent itemsets; concept drifts; count prediction

I. INTRODUCTION

Recently in many applications like retail and wholesale, data are in the form that being generated from endpoints and transmitted to the central management system in a continuous manner. Data with such a presentation form is termed *data stream*. The main features of a data stream are: (a) it is unbounded in terms of quantity, and (b) it possesses a dynamic rate of transmission as well as an unfixed data distribution. For those applications, techniques of data management and *knowledge discovery* are important and necessary.

Data mining is a process of finding useful information such as patterns from large databases [1, 2]. It has been widely studied in the past years. Some common classes of tasks involved in it include clustering, classification, regression analysis, frequent-itemset discovery, and association-rule learning. Discovering knowledge from data streams, that is, *data stream mining*, is valuable to many applications, but it is a difficult task. Many traditional mining approaches such as *multiply scanning* on the entire data (to generate mining answers) are infeasible in the real environment, which is mainly due to the reason that the data stream is continuous, dynamic, and infinite in volume. Therefore, the methods to data stream mining differ significantly from those to database mining.

Data stream mining has some basic constraints and requirements [12]. Firstly, elements of the data stream cannot be examined anymore as they have been out of the memory. Secondly, in contrast with the unlimited amount of stream data, the consumption of memory should be confined within a range. Thirdly, the current mining result on the data stream should be available anytime the user invokes a request. Fourthly, even though the data characteristic of the data stream may change with time, which is known as *concept drift*, the mining system with a mining algorithm should work normally and produce mining results with acceptable quality. As a result, a good algorithm for data stream mining needs to possess high efficiency and small memory consumption. Besides, the quality of mining results should be maintained acceptably in the presence of concept drifts.

In this research, we study the problem of mining frequent itemsets in concept-drifting data streams. In addition to *frequent-itemset discovery*, the issue of *concept drift* is considered and addressed. The main contributions of the research are as follows. Firstly, we propose a novel mining algorithm which uses a procedure of *count prediction* to get the counts of itemsets and find the frequent itemsets out. Secondly, we have successfully applied the technique of *regression analysis* to the area of frequent pattern mining (for the purpose of constructing the count-prediction functions). Thirdly, the proposed algorithm includes a way to give the concept of the data stream a *description*, so concept drifts would be handled by giving the changed concept a new and fitting description.

The rest of this paper is organized as follows. Section 2 briefly describes the related work regarding mining frequent itemsets as well as handling concept drifts in data streams. Section 3 gives the preliminaries and problem statement of this research. In Section 4, the idea to solve the problem is explained, and a mining algorithm is proposed accordingly. Section 5 shows the results of experimental evaluation on the proposed algorithm. Finally, Section 6 gives conclusions concerning this research.

II. RELATED WORK

The research work to data stream mining can basically be classified into three categories according to the data processing model [11]. One class is based on the *landmark window model* [3, 5, 13, 14]. In the model, there is a time point called *landmark*. The coverage of mining includes all data elements between the landmark and the current point. Another class refers to the *sliding window model* [4, 16, 20].

978-1-4673-0024-7/10/$26.00 ©2012 IEEE

In the model, a *window* which slides with time is present. The window always covers a certain number of most recent elements and the mining task at any time point focuses on elements within the current window. Yet another class takes the *damped window model* [17, 18, 19]. In the model, each element is associated with a *weight* which is relative to the time. Newly-received elements have higher values of weight than earlier-received elements.

In the area of classification, the term *concept* is considered as the whole distribution of the problem in a certain time point [6], which is represented by the joint distribution of the input (attributes) and the output (classes). Therefore, a *concept drift* represents a change in the distribution of the problem, which may be an unconditional change, a conditional change, or a dual change [7]. There are studies on handling concept drifts in online learning. In [9], the concept of *model granularity* is introduced, which results in a rule-based stream classifier. The authors showed that with a classifier of finer granularity, the local components in the classifier being affected by a concept can be pinpointed efficiently. In [8], the authors proposed a general framework of *weighted ensemble classifiers*. The classifiers in the ensemble are individually weighted based on their expected classification accuracy on the testing data. In [10], the authors presented an approach which builds a global set of classifiers and pertinently selects classifiers from the set to form the ensemble as well as represent the current concept of the stream.

In the literature on data-stream frequent-pattern mining, regardless of the processing model being adopted, existing methods cope with the data of a data stream either element by element [13, 14, 20] or batch by batch [3, 4, 5]. There are several notable methods. Manku and Motwani proposed an algorithm called *Lossy Counting* [3]. The algorithm uses an error parameter ε to ensure that no false-negative answers are present as well as the errors in individual itemset counts are bounded. Leung and Khan proposed a tree structure called *DSTree* [4]. The tree captures the contents of transactions in a sliding window to support exact stream mining. Li and Jea proposed the *SWCA algorithm* [16] which includes the application of the theory of *approximate inclusion-exclusion* [15]. The algorithm discovers frequent itemsets through a procedure called *support approximation*. Yu et al. proposed an algorithm called *FDPM* [5] which is based on the *Chernoff bound*. The algorithm uses a running error parameter to prune itemsets (control mining quality) and uses another reliability parameter to control memory size. However, to our best knowledge, most methods in the literature do not refer to the significant issue on data stream mining, namely *concept drift*.

III. PROBLEM DESCRIPTION

We define the basic terms related to the problem of this search as follows. Let $I = \{x_a, x_b, \ldots, x_m\}$ be a set of *attributes* where an attribute corresponds to an *item*. An *itemset* is a subset of I and also called a *pattern*. The length of an itemset is the number of items included in the itemset. Itemsets with a length of n are in the n_{th} order and called n-itemsets. A *transactio*n is a set of items, and it supports an itemset X if X is a subset of it. An itemset is supported at most once by a transaction. The *frequency count* or simply the *count* of an itemset in a set of transactions is the number of occurrences that the itemset has within the transaction set. With respect to a *threshold* of count, an itemset is called *frequent* or *large* if its count is equal to or greater than the threshold; otherwise it is called *infrequent*.

A *data stream* over I is a continuous sequence of elements where each element is a transaction. The *concept* of the data stream at a certain point of time refers to the data as well as the distribution of the data at that time. In terms of frequent pattern mining, it would be characterized by both the frequency counts of items and the frequency correlations between different items and/or itemsets. A *concept drift* therefore represents a change in either the data or the distribution of the data, or possibly in both. Concept drifts would let the set of frequent itemsets vary from time to time.

Given a data stream of transactions in which the concept may change with the time, together with two user-specified parameters, namely a count threshold called *minimum support* (*ms*) and a window size to the *sliding window* (*sw*), the goal of this research is to discover frequent itemsets from recent data elements of the data stream effectively. More specifically, we are to find the itemsets whose counts (i.e., numbers of occurrences) equal or exceed $ms \times sw$, from the data elements within the current sliding window, each time as there is a request for frequent patterns.

IV. PROPOSED METHOD

To solve the problem described above, we adopt a *prediction*-based mining approach. The frequency correlations between different items result in different frequencies of the combinations of items, that is, the itemsets. As long as the correlations (which are abstract) can be learned or modeled, we would predict the frequency counts of longer itemsets even if we do not know them, based on the frequency counts of single items and/or shorter itemsets. Once the counts of itemsets are known through *count prediction*, the frequent itemsets can be decided according to the minimum support. As a result, we record itemsets of the first two orders including their counts and use them to predict the counts of higher-order iemsets (which are not recorded). To describe the issue of count prediction more specifically, given two frequency counts concerning itemsets in the first two orders, a value of count is predicted for an itemset being the superset of the itemsets. A *function* or *model* is thus necessary for the procedure of count prediction.

For the purpose of predicting the frequency counts of itemsets, we employ the technique of *regression analysis* which is widely used for forecasting and prediction. In regression analysis, an observed dataset of *instances* is necessary for constructing the so-called *predictive model* or *regression function*. In this research, an instance consists of the frequency count of a higher-order itemset and the two counts of its first-two-order subsets. In other words, there are one dependent variable (i.e., the count of an itemset) and two independent variables (i.e., the counts of its subsets in the 1_{st} and 2_{nd} orders) in the prediction procedure. The way to generate the instances is described as follows. A *subsequence* of the data stream is taken, and the itemsets which are frequent in the data subsequence are discovered and stored in the data

structure. Any practical algorithm such as *Apriori* [1] or *FP-growth* [2] would serve the task. Next, for every frequent itemset in the data structure with a length greater than 2, an instance is generated by collecting the counts of an itemset and its first-two-order subsets. A dataset of instances is obtained as a result. Afterward the approach of *linear regression* is used on the dataset of instances to model the relationship. More specifically, it constructs a regression function by fitting a predictive model to the instances using the *least squares* approach. The model takes the form of $y=\beta_1 x_1+\beta_2 x_2+\varepsilon$, where y is the count of an itemset (having a length greater than 2), x_1 and x_2 are the counts of the itemset's 1-subsets and 2-subsets respectively, and β_1 and β_2 are the regression coefficients to independent variables x_1 and x_2 respectively.

The proposed algorithm for frequent itemset mining, which is based on count prediction and called *CPM* (count-predictive mining), is presented in Fig. 1. The CPM algorithm processes the data in a data stream batch by batch (Lines 15-19), and the slide of the sliding window and the update of the data structure proceed in batch mode accordingly. The first batch of data is used to generate the observed instances and construct the predictive models (Lines 1-11) and thus corresponds to the aforesaid subsequence of stream data. In the algorithm, one predictive model is constructed for each order of itemsets having a length greater than 2 (Line 9), so there is a *group* of predictive models in practice for estimating

Algorithm CPM (Count-Predictive Mining)

Data structure: a prefix tree *P*; a linked list *F*
Input: a data stream *DS*; a minimum-support threshold *ms*; a sliding-window size *sw*
Output: a set of frequent-itemset answers

Pseudocode:

```
01. Receive a batch of transactions from DS;
02. Find the frequent itemsets in the batch and store them in F;
03. n ← 3;
04. while frequent itemsets in the nth order exist in F
05.     foreach frequent n-itemset X in F
06.         Find from F the 1- and 2-subsets of X and sum up their counts respectively;
07.         Collect the counts of X' 1-subsets, X' 2-subsets, and X itself to form an instance;
08.     end foreach
09.     Construct a count-prediction model of y=β1x1+β2x2+ε for itemsets of the nth order;
10.     n ← n+1;
11. end while
12. Divide the sliding window conceptually into m panes, each with a size of sw/m;
13. while data of DS is still streaming in
14.     Set F to be empty;
15.     while there is no output demand from the user
16.         Receive sw/m transactions from DS as a batch;
17.         Count in the batch the numbers of occurrences of 1-items and 2-itemsets;
18.         Update the entries of itemsets in P accordingly;
19.     end while
20.     Merge the counts of the m panes to get itemsets' counts in the current window;
21.     Select all frequent 1-items and 2-itemsets from P and insert them into F;
22.     n ← 2;
23.     while frequent itemsets in the nth order are discovered in F
24.         n ← n+1;
25.         Generate candidate n-itemsets from the frequent (n-1)-itemsets;
26.         Filter out those n-itemsets which have infrequent subsets;
27.         Get the counts of the n-itemsets by the count-prediction model for the nth order;
28.         Insert the n-itemsets whose counts ≥ ms×sw into F;
29.     end while
30.     Output the itemsets in F as the mining answers;
31. end while
```

Figure 1. Algorithm CPM (count-predictive mining).

the counts of itemsets of different orders. As a batch of stream elements is received, CPM enumerates and counts itemsets of lengths 1 and 2 and stores the itemsets as the *synopsis* in its data structure (Lines 16-18). The data structure for storing the itemsets is a *prefix tree*. When there is a mining demand, CPM selects the frequent 1-items and 2-itemsets from the data structure (Lines 20-21) and then calculates the frequency counts of higher-order itemsets through the procedure of count prediction (Lines 22-29). In the procedure, itemsets greater than 2 in length are count predicted in order, from shorter lengths to longer lengths, and the *Apriori property* [1] is applied to reduce the number of candidate large itemsets (Line 26). After the procedure the itemsets whose counts are equal to or greater than $ms \times sw$ are outputted as the mining answers (Line 30). Afterward the algorithm continues processing the incoming data of the data stream, if any (Line 13).

We give some analyses on the proposed algorithm. Assume that in a data stream the number of distinct attributes is m (for example, m=1000). Such a data stream may have up to m different orders of itemsets. With regard to the data stream, the CPM algorithm records only (the first) two orders of itemsets out of the m orders. Thus the memory consumed by its prefix tree structure is generally small. Besides, since the frequency counts of higher-order itemsets are obtained by predictive calculation (as needed) rather than by taking count of the occurrences, the efficiency of stream processing as well as synopsis maintenance would be high.

We also give a discussion on the issue of concept drift. In the proposed algorithm, the predictive models are formed based on the subsequence of the data-stream elements, or more specifically, the generated dataset of frequency-count relationship instances. The models would therefore serve as a *description* of the concept of the data stream (at the point of time they are constructed). As a concept drift happens, the accuracy of count prediction resulting from the models may fall because of the change in the concept, which will degrade the quality of mining. However, the concept is described by the predictive models which can be constructed using the techniques of regression analysis. With respect to a concept drift, we would take another subsequence of stream elements (having the changed concept) to construct a group of predictive models describing the changed concept. In other words, the predictive models to count prediction can be *reconstructed* (i.e., updated) for concept drifts. Compared with the original models, the reconstructed models adapt to the changed concept and can predict the counts of itemsets more accurately on the concept-drifted data. As a result, the quality of frequent pattern mining would be preserved effectively.

V. EXPERIMENTAL EVALUATION

TABLE I. DESCRIPTION OF THE TESTING CONCEPT-DRIFTING DATASET

	Duration (Element Number)	Attribute Number	Avg. Element Length	Max. Element Length	Avg. Large-Itemset Length
Concept A	499,243	2000	8.29	28	4
Concept B	500,757	1500	15.23	41	8

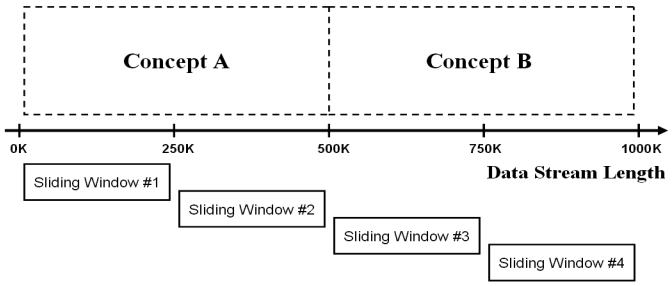

Figure 2. Relationship between the sliding windows and the stream concepts.

We have conducted several experiments to evaluate the performance of the proposed algorithm namely CPM, and the results are presented in this section. The experiments were carried out on a personal computer with a Pentium 2.80 GHz dual-core CPU and about 2.50 GB of available memory. The operating system is Windows XP Professional SP3. The program of the CPM algorithm is implemented in C++.

We use artificial data to simulate the data source, namely a concept-drifting data stream. The testing dataset has one million of transactions, and it is actually composed of two synthetic datasets representing distinct concepts, say, A and B. Each dataset is generated by the *synthetic data generator of IBM* [21] and has a size of approximately 500 thousands of transactions. Table I shows the statistics of the testing dataset.

The setting to the experiments is described as follows. The sliding window is 250 thousands of transactions in size and divided conceptually into 5 panes, that is, 50 thousands of transactions per pane. Elements of the testing data stream are processed batch by batch, where the size of a batch corresponds to that of a pane. The CPM algorithm takes a sequence of fifty-thousand transactions to construct the models for count prediction. The range of minimum support is between 0.1% and 1.0%. The answers of frequent itemsets are mined and outputted as an outcome every separate sliding window. Fig. 2 illustrates the relationship between the sliding windows and the concepts of the data stream. As one can see from the figure, a concept drift happens between the second and third sliding windows.

We first present the experimental results of *running time* and *memory consumption* in Fig. 3, where the value of minimum support is 0.5%. According to the figure, the performance of the CPM algorithm, in terms of both runtime and memory, is stable across the sliding windows having the same concept of data (for example, at sliding windows #1 and #2). More importantly, it is not affected greatly by the concept drift (that is, from sliding window #2 to sliding window #3). The proposed method counts and records itemsets of the first two orders, so the synopsis it has to keep is generally small in scale. No matter what concept the stream is currently in and/or what value the minimum support is specified, it can efficiently process the received data elements and maintain the extracted synopsis. This is the reason for the stable performance.

The next experiment tests for mining quality in terms of *accuracy*. The accuracy (of a frequent-itemset mining outcome) is assessed by the *F-measure* which is the harmonic mean of the *precision* and *recall* (of the outcome). The higher an F-measure (ranging from 0% to 100%) is, the better the mining accuracy/quality is. Two methods are tested and compared with each other in the experiment, namely *CPM* and *CPM+*. In regard to the concept-drifting data stream, CPM constructs a group of predictive models and uses it consistently to count-prediction based mining over the stream, while CPM+ constructs individual groups of predictive models upon the concepts and uses them to count-prediction based mining respectively. In other words, CPM+ is for addressing concept drifts while CPM is not. We show the result of the experiment in Fig. 4, where two values of minimum support were specified namely 0.5% (larger) and 0.4% (smaller). In the figure, CPM and CPM+ achieve much the same high-accuracy before the concept drift, that is, at sliding windows #1 and #2. However, as the concept drift happens, the accuracy of CPM falls severely, for example, by nearly 30% at the minimum support of 0.4%. The reason is that the constructed count-prediction models (describing a concept) do not fit the changed concept anymore. On the other hand, CPM+ retains acceptable accuracy without a major decrement because the reconstructed count-prediction models adapt to the altered concept. According to this experiment, CPM+ would handle concept drifts effectively. Taking the results of all the experiments together, it is observed that the proposed algorithm possesses a satisfactory performance in mining frequent itemsets from a concept-drifting data stream.

VI. CONCLUSIONS

Data streams are dynamic sources to data mining algorithms. The concept in a data stream may change with the time, which could affect the performance of a data-stream

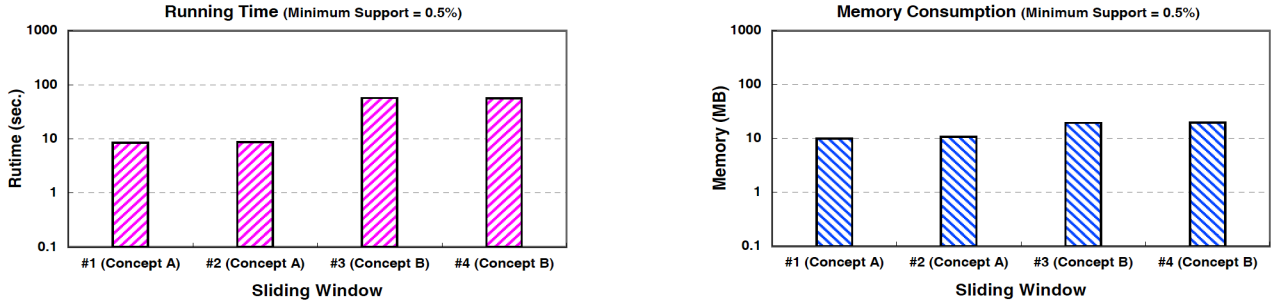

Figure 3. Experimental results of running time (left) and memory consumption (right).

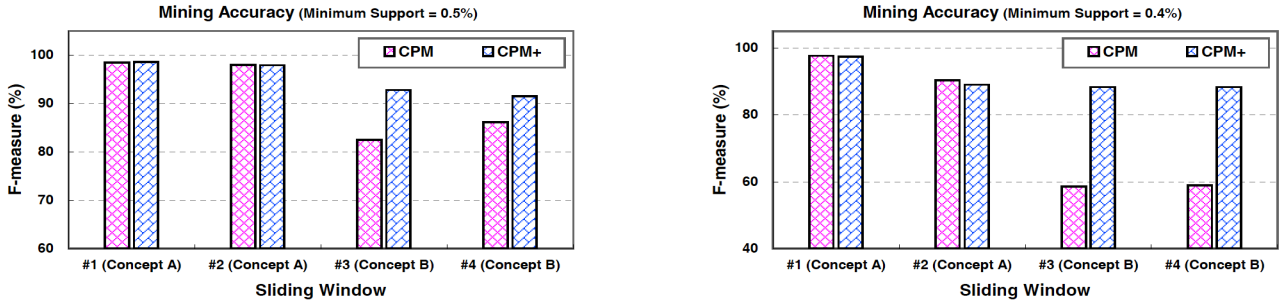

Figure 4. Experimental result of mining accuracy at the minimum supports of 0.5% (left) and 0.4% (right).

mining algorithm, for example, in terms of accuracy. In this paper, we study the problem of data-stream frequent-itemset discovery and propose a count-prediction based mining algorithm called CPM. The algorithm uses the techniques of regression analysis to construct predictive models. With respect to an input of data stream, CPM records simply 1-items and 2-itemsets and can calculate the counts of higher-order itemsets (through the predictive models) upon a mining request. Because the predictive models are constructed on the basis of the stream data, they serve as a description of the stream's concept at the time of construction. If a concept drift happens, the predictive models can be reconstructed to fit the new concept.

According to our experimental results, the proposed algorithm is feasible to discover frequent patterns and performs well. It has stable running efficiency and consumes a small amount of memory. Besides, as new predictive models are constructed upon the concept-changed data, the proposed algorithm would preserve the quality (accuracy) of its mining answers at an acceptable level in the presence of concept drifts. For a dynamic data stream, constructing and using respective predictive models on different concepts, or reconstructing the predictive models upon the changed concept, works well against concept drifts. As a future goal, we are to investigate practical means for concept-drift detection.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499, 1994.

[2] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," Data Mining and Knowledge Discovery, vol. 8, pp. 53–87, 2004.

[3] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," Proceedings of the 28th International Conference on Very Large Data Bases, pp. 346–357, 2002.

[4] C. K.-S. Leung and Q. I. Khan, "DSTree: a tree structure for the mining of frequent sets from data streams," Proceedings of the 6th International Conference on Data Mining, pp. 928–932, 2006.

[5] J. X. Yu, Z. Chong, H. Lu, Z. Zhang, and A. Zhou, "A false negative approach to mining frequent itemsets from high speed transactional data streams," Information Sciences, vol. 176, pp. 1986–2015, 2006.

[6] A. Narasimhamurthy and L. I. Kuncheva, "A framework for generating data to simulate changing environments," Proceedings of the 25th IASTED International Multi-Conference, pp. 384–389, 2007.

[7] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," Proceedings of the 7th Brazilian Symposium on Artificial Intelligence, pp. 286–295, 2004.

[8] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining, pp. 226–235, 2003.

[9] P. Wang, H. Wang, X. Wu, W. Wang, and B. Shi, "On reducing classifier granularity in mining concept-drifting data streams," Proceedings of the 5th International Conference on Data Mining, pp. 474–481, 2005.

[10] S. Ramamurthy and R. Bhatnagar, "Tracking recurrent concept drift in streaming data using ensemble classifiers," Proceedings of the 6th International Conference on Machine Learning and Applications, pp. 404–409, 2007.

[11] Y. Zhu and D. Shasha, "StatStream: statistical monitoring of thousands of data streams in real time," Proceedings of the 28th International Conference on Very Large Data Bases, pp. 358–369, 2002.

[12] M. N. Garofalakis, J. Gehrke, and R. Rastogi, "Querying and mining data streams: you only get one look," Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 635, 2002.

[13] R. Jin and G. Agrawal, "An algorithm for in-core frequent itemset mining on streaming data," Proceedings on the 5th International Conference on Data Mining, pp. 210–217, 2005.

[14] K.-F. Jea and C.-W. Li, "Discovering frequent itemsets over transactional data streams through an efficient and stable approximate approach," Expert Systems with Applications, vol. 36, pp. 12323–12331, 2009.

[15] N. Linial and N. Nisan, "Approximate Inclusion–Exclusion," Combinatorica, vol. 10, pp. 349–365, 1990.

[16] C.-W. Li and K.-F. Jea, "An adaptive approximation method to discover frequent itemsets over sliding-window-based data streams," Expert Systems with Applications, vol. 38, pp. 13386–13404, 2011.

[17] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu, "Mining frequent patterns in data streams at multiple time granularities," Data Mining Next Generation Challenges and Future Directions, pp. 191–212, 2004.

[18] J. H. Chang and W. S. Lee, "Finding recently frequent itemsets adaptively over online transactional data streams," Information Systems, vol. 31, pp. 849–869, 2006.

[19] D. Lee and W. Lee, "Finding maximal frequent itemsets over online data streams adaptively," Proceedings of the 5th International Conference on Data Mining, p.p. 266–273, 2005.

[20] H.-F. Li and S.-Y. Lee, "Mining frequent itemsets over data streams using efficient window sliding techniques," Expert Systems with Applications, vol. 36, pp. 1466–1477, 2009.

[21] IBM Quest Market-Basket Synthetic Data Generator, [http://www.cs.loyola.edu/~cgiannel/assoc_gen.html].