Processing Count Queries over Event Streams at Multiple Time Granularities

Aykut Ünal*, Yücel Saygın †, Özgür Ulusoy*

*Department of Computer Engineering, Bilkent University, Ankara, Turkey.
†Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey.
E-mail: {unala, oulusoy}@cs.bilkent.edu.tr, ysaygin@sabanciuniv.edu

Abstract

Management and analysis of streaming data has become crucial with its applications in web, sensor data, network traffic data, and stock market. Data streams consist of mostly numeric data but what is more interesting is the events derived from the numerical data that need to be monitored. The events obtained from streaming data form event streams. Event streams have similar properties to data streams, i.e., they are seen only once in a fixed order as a continuous stream. Events appearing in the event stream have time stamps associated with them in a certain time granularity, such as second, minute, or hour. One type of frequently asked queries over event streams is count queries, i.e., the frequency of an event occurrence over time. Count queries can be answered over event streams easily, however, users may ask queries over different time granularities as well. For example, a broker may ask how many times a stock increased in the same time frame, where the time frames specified could be hour, day, or both. This is crucial especially in the case of event streams where only a window of an event stream is available at a certain time instead of the whole stream. In this paper, we propose a technique for predicting the frequencies of event occurrences in event streams at multiple time granularities. The proposed approximation method efficiently estimates the count of events with a high accuracy in an event stream at any time granularity by examining the distance distributions of event occurrences. The proposed method has been implemented and tested on different real data sets and the results obtained are presented to show its effectiveness.

Index Terms - Count Queries, Data Streams, Event Streams, Time Granularity, Association Rules, Data Mining

1 Introduction

The amount of electronic data has increased significantly with the advances in data collection and data storage technologies. Traditionally, data are collected and stored in a repository and queried or mined for useful information upon request. However, in the

case of applications like sensor networks and stock market, data continuously flow as a stream and thus need to be queried or analyzed on the fly. Streaming data (or data streams) brought another dimension to data querying and data mining research. This is due to the fact that, in data streams, as the data continuously flow, only a window of the data is available at a certain time. The values that appear in data streams are usually numerical, however what is more interesting for the observers of a data stream is the occurrence of events in the data stream. A very high value or an unusual value coming from a sensor could be specified as an interesting event for the observer. The events occurring in a stream of data constitute an event stream, and an event stream has the same characteristics as a data stream, i.e., it is continuous and only a window of the stream can be seen at a time. Basically, an event stream is a collection of events that are collected from a data stream over a period of time. Events in an event stream are observed in the order of occurrence, each with a timestamp that captures the time unit supported by the system. The time unit used can be day, hour, second or any other granularity. Experts would like to extract information from an event stream, such as the value of an event at a specific time-tick; frequency of certain events, correlations between different events, regularities within a single event; or future behavior of an event. Relationships among the events can be captured from event streams via online data mining tools.

1.1 Motivation

Given an event stream at a particular granularity, we are interested in frequencies of events in the event stream at coarser time granularities. Consider, for instance, a stock broker who wants to see how many times a stock peaks in hourly, daily and weekly basis. For each time granularity (i.e., hour, day, week), the counts change. For fraud detection

in telecommunication, it may be interesting to know the count of different calls made by a suspicious person hourly or daily. Data stream coming from sensor networks in a battle field for detecting the movements around a region can be queried to find out the count of moving objects in an hourly and daily fashion to estimate the military activities. All these example queries require the analysis of the event streams at various granularities, such as hour, day, and week.

1.2 Contribution

The main focus of our work is to find the frequencies of events in an event stream at different time granularities. Our main contribution is to propose a method that efficiently estimates the count of an event at *any* time granularity and runs in linear time with respect to the length of the given stream. With our method, the event stream is analyzed only once, and summary information is kept in an incremental fashion for frequency estimation. Our method utilizes distance histograms of event occurrences for event count prediction at multiple time granularities. Distance histograms can also be used for event occurrence prediction besides event count prediction. Although the distance histograms induce some storage overhead, this overhead could be justified by their multiple uses. We discuss event occurrence prediction via distance histograms in Section 6.

Most of the Data Mining methods proposed so far are based on finding the frequencies of data items and then generating and validating the candidates against the database [1]. Even the methods that do not perform candidate generation rely on finding the frequencies of the items as the initial step [17]. Therefore, in addition to efficient answering of count queries at multiple time granularities, our methods can also be used by data mining algorithms on data streams to find frequent itemsets at multiple time

granularities.

The rest of the paper is organized as follows. The next section summarizes the related work. Section 3 explains the basic concepts and the notation used throughout the paper. Section 4 presents the method proposed to predict the count of an event at different time granularities. Section 5 gives the results of several experiments conducted on real life data to evaluate the accuracy of the method and the impact of several parameters. Section 6 provides a brief discussion on estimation of event occurrences through distance histograms. Finally, the last section concludes the paper with a discussion of the proposed method and further research issues.

2 Related Work

In this section, we summarize previous work related to our method which can be divided into three categories: Data Mining, Time Granularity, and Histograms.

2.1 Data Mining

Handling of data streams has become a major concern for database researchers with the increase of streaming data sources like sensor networks, phone calls in telephone networks [3, 9], client requests for data in broadcast systems [29] and e-commerce data on World Wide Web, stock market trades, and HHTP requests from a web server. Given these huge data sources, data mining researchers moved into the domain of mining data streams [11]. In this emerging area, the temporal dimension and time granularities are yet to be explored.

Association rule mining has been well studied in the context of data mining, however there is no work on mining associations at multiple time granularities. The work we have

performed can also be applied to association rule mining at multiple time granularities. The problem and the corresponding terminology in association rule mining was first introduced in market basket analysis, where the items are products in your shopping card and associations among these purchases are looked for [1]. Each record in the sales data consists of a transaction date and the items bought by the customer. The issue of discovering frequent generic patterns (called episodes) in sequences was explained by Mannila et.al. in [23] where the events are ordered in a sequence with respect to the time of their occurrence at a certain time granularity. In their work, an episode was defined as a partially ordered set of events, and can also be described as a directed acyclic graph. Their iterative algorithm builds candidate episodes using the frequent episodes found in the previous iteration. They extended their work in [22] to discover generalized episodes, and proposed algorithms for discovering episode rules from sequences of events. In [8], Das et.al. aimed at finding local relationships from a time series, in the spirit of association rules, sequential patterns, or episode rules. They convert the time series into a discrete representation by first forming subsequences using a sliding window and then clustering these subsequences using a pattern similarity measure. Rule finding algorithms such as episode rule methods can be used directly on the discretized sequence to find rules relating temporal patterns. In a recent work, Gwadera et.al. investigated the problem of the reliable detection of an abnormal episode in event sequences, where an episode is a particular ordered sequence occurring as a subsequence of a large event stream within a window of size w, but they did not consider the case of detecting more than one episode [15]. This work was extended in [2] to the case of many pattern sequences, including the important special case of all permutations of the same sequence. All these works are different from ours in that they investigate temporal relationships but only at a single time granularity.

Cyclic associations where each association has a cyclic period associated with it were studied by Özden et al. in [25]. But the authors only investigated the case where the database has a fixed time granularity. Another work by Pavlov et al. considered count queries for itemsets on sparse binary transaction data [26]. The authors used probabilistic models to approximate data for answering queries on transactional data. In [21], Mannila and Smyth used enthropy models to answer count queries over transactional data. In both of these works, the authors did not consider the time dimension. Again a recent work by Bouicaut et. al. describes methods for approximate answering of frequency queries over transactional data without considering time dimension and time granularities [6].

2.2 Time Granularity

Given an event stream, we are interested in estimating the frequencies of event occurrences at coarser time granularities. Data analysis at multiple time granularities was already explored in the context of sequential pattern mining by Bettini et al. [5]. However, the target of their work is completely different from ours in that, they try to find sequences with predefined beginning and ending timestamps, and they would like to find sequences that have these predefined timestamps at multiple time granularities. Our target, however, is to find frequencies of event occurrences at multiple time granularities without any time restriction.

Temporal aggregation queries were well studied and several approaches have been proposed recently [10, 12, 14, 20, 24, 30, 32, 34]. However, all these works consider only a single time granularity, where this granularity is usually the same as the granularity used to store the time attributes. To the best of our knowledge, the only work exploring the aggregate queries of streaming data in the time dimension at multiple time granularities

appeared in [33], where Zhang et.al. present specialized indexing schemes for maintaining aggregates using multiple levels of temporal granularities: older data is aggregated using coarser granularities while more recent data is aggregated with finer detail. If the dividing time between different granularities should be advanced, the values at the finer granularity are traversed and the aggregation at coarser granularity is computed. Their work is different from ours in that, they calculate the exact aggregate function of the stream at predefined coarser time granularities by performing queries. However, we scan the stream only once and estimate the frequency of the event at any arbitrary time granularity without storing any information at intermediate time granularities.

2.3 Histograms

In order to estimate event occurrence frequencies at coarser time granularities, we obtain statistical information from the event stream which is similar to histograms. In order to construct an histogram on an attribute domain X, the data distribution τ of attribute X is partitioned into β (≥ 1) mutually disjoint subsets, called *buckets*. A uniform distribution is assumed within each bucket, i.e., the frequency of a value in a bucket is approximated by the average of the frequencies of all values in the bucket. The point in histogram construction is the partitioning rule that is used to determine the buckets. Various types of histograms have been proposed and used in several commercial systems. The most popular ones are the *equi-width* [19] and *equi-height* [19, 27] histograms. Equi-width histograms group contiguous ranges of attribute values into buckets such that the widths of each bucket's range is the same. Equi-height histograms are partitioned such that the sum of all frequencies in each bucket is the same and equal to the total sum of all frequencies of the values in the attribute domain divided by the number of buckets. Another important class of histograms is the *end-biased* [18] histograms, in which some

of the highest frequencies and some number of the lowest frequencies are explicitly and accurately stored in individual buckets, and the remaining middle frequencies are all grouped in one single bucket. Indeed, this type of histogram is the most suitable data structure for our count estimation algorithm, because, the experiments we conducted on real-life data show that the distribution of the distance between two occurrences of an event in a history tends to have high frequencies for some small distance values, and very low frequencies for the remaining larger values. Therefore, we use end-biased histograms, in which some of the values with the highest and lowest frequencies are stored in individual buckets, and the remaining values with middle frequencies are grouped in a single bucket. Readers who are interested in further detailed information on histogram types, construction and maintenance issues are referred to [28], which provides a taxonomy of histograms that captures all previously proposed histogram types and indicates many new possibilities. Random sampling for histogram construction has also been widely studied, and several algorithms have been proposed and used in many different contexts in databases [7, 13, 16, 27]. The aim of all these works is to use only a small sample of the data to construct approximate histograms that gives reasonably accurate estimations with high probabilities.

3 Basic Concepts and Notation

This section includes the definitions of some basic concepts and the notation used throughout the paper. For ease of reference, a summary of the most frequently used notation is given in Table 3.1.

We start by defining *granularity*, the most fundamental concept [4].

Definition 3.1 *A* granularity *is a mapping G from the positive integers (the* time-ticks*)*

Notation	Description
$\preceq$	finer than
S_1	Base Stream
S_g	An Event Stream at granularity g
$c_{gg'}$	Transformation coefficient of the transformation $S_g \rightarrow S_{g'}$
d_i^g	A distance of length i in S_g
D_g	Distance distribution of S_g

Table 3.1: Summary of Notation

to subsets of the Time Domain satisfying:

1. $\forall i, j \in Z^+$ *such that* $i < j$, $G(i) \neq \emptyset$ *and* $G(j) \neq \emptyset$, *each number in* $G(i)$ *is less than all numbers in* $G(j)$,

2. $\forall i, j \in Z^+$ *such that* $i < j$, $G(i) = \emptyset$ *implies that* $G(j) = \emptyset$.

The first condition states that the mapping must be *monotonic*. The second one states that if a time-tick of G is empty, then all subsequent time-ticks must be empty as well. Intuitive granularities such as *second, minute, hour, day, month* all satisfy these conditions. For example, the *months* in year 2002 can be defined as a mapping G such that $\{G(1)$=*January*,..., $G(12)$=*December*$\}$, and $G(i) = \emptyset$ for all $i > 12$. Since the mapping G satisfies both conditions, *month* is a valid granularity. There is a natural relationship between granularities as follows [4]:

Definition 3.2 *Let* G *and* H *be two granularities. Then,* G *is said to be* finer than H, *denoted as* $G \preceq H$, *if for each time-tick* i *in* G, *there exists a time-tick* j *in* H *such that* $G(i) \subseteq H(j)$.

If $G \preceq H$, then H is said to be *coarser than* G. For example, *day* is finer than *week*, and coarser than *hour*, because every *day* is a subset of a *week* and every *hour* is a subset of a *day*.

Definition 3.3 *An* Event Stream S_g *is a collection of time-ticks at granularity* g *and an event corresponding to each time-tick. More formally,* $S_g = \{< t_i, e_i > | i \geq 1, t_i \in T_g, e_i \in E\}$*, where* T_g *is the set of time-ticks at granularity* g*, and* E *is the universal set of event states for the particular system in concern. The* Length *of the stream is equal to the total number of time ticks registered for that stream, and is denoted as* $S_g.length$*.*

Definition 3.4 *An event stream can be given with a particular granularity to be transformed to coarser granularities. The event stream generated by the application in concern is called* the Base Stream*, denoted by* S_1*, and its time granularity is called* the Base Granularity*.*

As an example, consider the daily percentage price changes of a particular stock exchanged in a stock market between January 1^{st}, 2002 and December 31^{st}, 2002. Here, *event* is the price change of the stock, *granularity* is *business-day*, T_g is the set of all business-days in year 2002, and E is the set of all possible event states, such as $E = \{fall, no\ change, rise\}$ or $E = \{(-\infty, -2\%), [-2\%, 0), [0, 0], (0, 2\%], (2\%, \infty)\}$ (At each time-tick, the event has one of the five states according to the interval the price change falls into). In our work, we are interested in event streams whose set of all possible event states are 0 and 1, namely $E = \{0, 1\}$.

Definition 3.5 *A* 0/1 event stream *is an event stream where each time-tick records the state of the event at that time-tick, which is equal to 1 if the event occurs, and 0 otherwise.*

When we transform an event stream S_g at time granularity g to another event stream $S_{g\prime}$ at granularity $g\prime$, we obtain a different set of time-ticks and different sets of events associated with these time-ticks. Before we give the formal definition of transformation of a stream, the following two concepts need to be introduced.

Definition 3.6 *Suppose that an event stream S_g is transformed to an event stream $S_{g'}$. Then,* Transformation Coefficient*, denoted by $c_{gg'}$, is the total number of time-ticks in S_g that correspond to a single time-tick in $S_{g'}$.*

For example, seven *day*s form one *week*, yielding a transformation coefficient equal to 7.

Definition 3.7 *A* Transformation Operation *is a mapping $P : E^c \rightarrow E$ that takes event states at* c *successive time-ticks where* c *is the transformation coefficient, and returns a single event state according to the particular operation in use.*

Some common transformation operations are *MinMerge, MaxMerge, AvgMerge, SumMerge, Union, and Intersection.* For example, *MinMerge* operation returns the minimum event value from the set of c events, where c is the transformation coefficient. The other operations are defined similarly. In this paper, we are interested in 0/1 (boolean) event stream where the universal set of event states is {0,1}, and we use *mergeOR* operation that logically ORs the event values at corresponding time-ticks. Besides *mergeOR*, some other transformation operations can also be used as long as their output is also a boolean event stream.

Definition 3.8 *Let $S_g = \{< t_i, e_i > | i \geq 1, t_i \in T_g, e_i \in E\}$ be an event stream, P be a transformation operation, and c be the transformation coefficient. Then, the* transformation *of S_g to another stream $S_{g'}$ with granularity g' is provided in such a way that, $S_{g'} = \{< t'_j, e'_j > | j \geq 1, t'_j \in T_{g'}, e'_j \in E\}$, where $e'_j = P(e_{(j-1)*c+1}, e_{(j-1)*c+2}, \ldots, e_{j*c})$ and $t'_j \in T_{g'}$ corresponds to time-ticks $[t_{(j-1)*c+1}, t_{j*c}] \subseteq T_g$.*

Consider the transactional database of a retail company that stores the purchased items in a daily basis. And consider the transformation of the "milk purchase history" at granularity *day* to granularity *week*. Then, the i^{th} week corresponds to the days between

$[day_{(i-1)*7+1}, day_{i*7}]$, and stores 1 if the milk is purchased on any of the corresponding days. For instance, the first week corresponds to the first 7 days, and the third week corresponds to days [15,21]. Note that, stream S_g can be transformed to $S_{g\prime}$ only if $g < g\prime$, and $c_{gg'}$ is an integer, i.e., $g\prime$ is a multiple of g. During the transformation, the event corresponding to a time-tick $t'_j \in T'_g$ is constructed by applying the transformation operation P to the event sequence of length $c_{gg'}$ in S_g at time-ticks corresponding to t'_j. Since the only transformation operation we use is *mergeOR*, we omit the specification of the operation used in transformations throughout the paper. Then, the transformation of S_g to $S_{g'}$ becomes equivalent to dividing S_g into blocks of length $c_{gg'}$ and checking whether the event occurs at any time-tick in each of these blocks. If so, the corresponding t'_j in $S_{g'}$ records 1, and 0 otherwise. Note that the number of the blocks of length $c_{gg'}$ is equal to $\lceil S_g.length/c_{gg'} \rceil$, which also gives the cardinality of $T_{g'}$.

The count of an event at granularity $g\prime$ can be found by constructing $S_{g\prime}$ and counting the time-ticks at which the event occurred. However, this naive method is quite infeasible in case of event streams where the stream is available only once and as a set of windows. Considering this limitation incurred by event streams, we propose a method that reads the given stream once and then estimates the count of the event at any coarser granularity efficiently and accurately. This is accomplished as follows: *Distance* between two successive occurrences of an event is defined as the number of time-ticks between these occurrences. We examine the distribution of the distances within the whole sequence, and then observe the possible values to which each particular distance value can transform during the transformation of S_g to $S_{g\prime}$. We formulate these observations to be able to capture the possible distance transformations along with their corresponding probabilities. The formal definitions of *distance* and *distance distribution* can be given as follows:

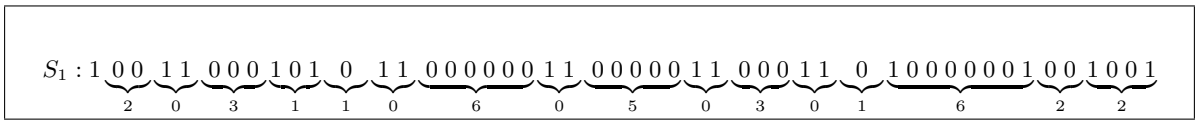

Figure 3.1: An Example of Event Stream

Definition 3.9 *Given a 0/1 event stream S_g ($g \geq 1$), the* distance *between two event occurrences is defined to be the number of zeros between the time-ticks at which the event occurs in the stream.*

A distance *of length* i *in S_g is denoted by d_i^g. If the event occurs at any two successive time-ticks, then we have a distance of length 0 (d_0^g).*

Definition 3.9 becomes ambiguous when a stream starts or ends with zero(s). These special cases are treated in Section 4.6 in detail.

Definition 3.10 *The* Distance Distribution *of an event stream S_g is the set of pairs*

$$D_g = \{(d_0^g, c_0^g), (d_1^g, c_1^g), (d_2^g, c_2^g), \ldots, (d_{m_g}^g, c_{m_g}^g)\}$$

where m_g is the maximum distance value observed in S_g, and c_i^g gives the count *of the distance d_i^g in S_g ($0 \leq i \leq m_g$).*

For convenience, we use array notation to refer the counts of distance values such that $D_g[i] = c_i^g$.

As an example, consider the *base event stream* S_1 given in Figure 3.1. Corresponding distance distribution is given in Table 3.2.

d_i^1	$D_1[i]$	$F_1[i]$
0	5	0.3125
1	3	0.1875
2	3	0.1875
3	2	0.1250
4	0	0.0
5	1	0.0625
6	2	0.1250
Total	**16**	**1.0**

Table 3.2: The Distribution of Distance

d_i^1 : possible distance values in S_1

$D_1[i]$: count of d_i^1

$F_1[i]$: relative frequency of d_i^1

$$(F_1[i] = \frac{D_1[i]}{\sum_{j=0}^{m_g} D_1[j]})$$

4 Estimation of an Event's Count at Coarser Granularities

The aim of our work is to estimate accurately the count of an event in an event stream at *any* time granularity g by using an efficient method in terms of both time and space considerations. The brute-force technique to scan the given stream and generate the stream at each time granularity in question is unacceptable due to the fact that when the person monitoring the event streams wants to query it in a different time granularity, the part of the event stream that contains the past events can not be brought back for further analysis. The method we propose in this paper is based on analyzing the event stream only once as it flows continuously. Some statistical information about the frequency and distribution of the event occurrences is collected, and used to estimate the frequency (or count) of the event at any coarser time granularity. One can think

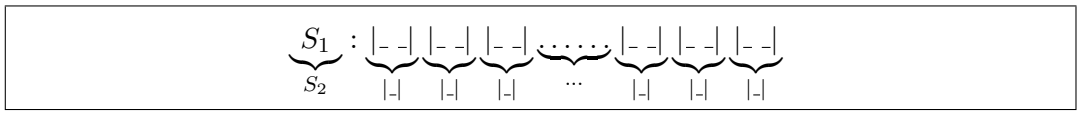

Figure 4.2: Transformation with Granularity 2

that the event frequencies could be calculated for all possible time granularities as the event stream flows, but this is also not practical since there exist a large number of possible time granularities. In order to show how a particular distance can transform to different values with certain probabilities, we first analyze the transformation of a base event stream (i.e., a stream with granularity 1) to event streams with granularities 2 and 3. Understanding how transformation takes place with small granularities will help to generalize the estimation method for arbitrary granularities.

4.1 Estimation at Time Granularity 2

For the simplest case, consider the transformation of the base event stream S_1 (at granularity 1) to event stream S_2 (at granularity 2). During this transformation, we will examine how the distance array D_1 changes and transforms to D_2. As we have already mentioned, this transformation is equivalent to dividing S_1 into blocks of length 2 and checking whether the event occurs at any time-tick in these blocks. If so, the corresponding time-tick t_i in S_2 records 1, and 0 otherwise. This is shown in Figure 4.2.

A distance d_0^1 indicates a subsequence "11" of length 2 in S_1. During the transformation of S_1 to S_2, there are two cases : Either both of 1s are in the same block, or they are in two successive blocks. As shown in Figure 4.3, the first case yields a single 1 in S_2, which means that d_0^1 vanishes in D_2 (also in S_2); while the second one preserves both 1s in S_2, i.e., d_0^1 in S_1 transforms to d_0^2 in S_2. From a probabilistic point of view, both of these cases have 50% probability and are equally likely to happen.

Similarly, a distance d_1^1 represents the subsequence "101" in S_1 and yields two differ-

ent cases which are specified in Figure 4.4. However, for the distance d_1^1, the two cases give the same result indicating that d_1^1 in S_1 always becomes d_0^2 in S_2.

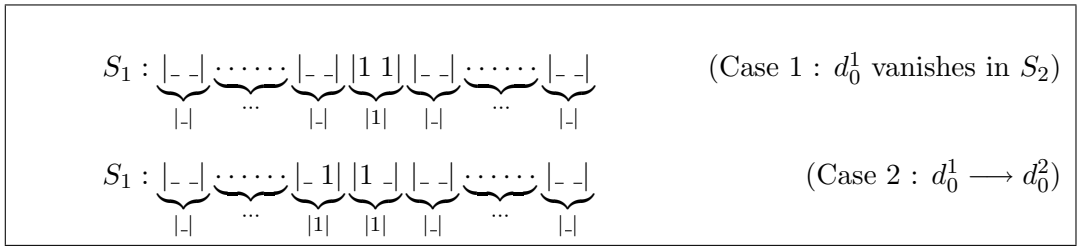

Figure 4.3: Transformation $D_1 \longrightarrow D_2$ for d_0^1

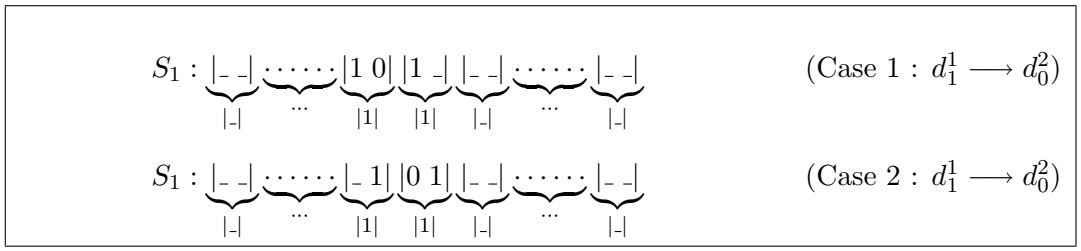

Figure 4.4: Transformation $D_1 \longrightarrow D_2$ for d_1^1

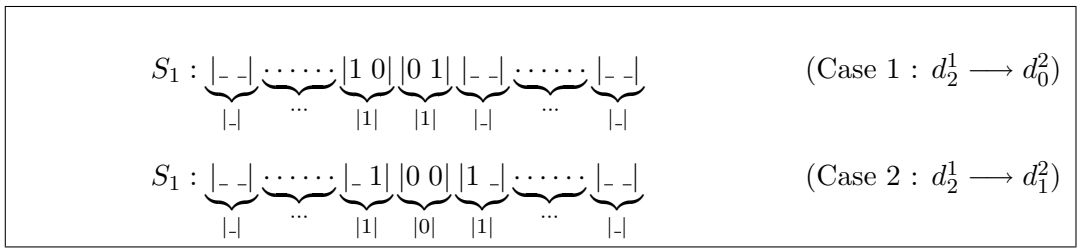

Figure 4.5: Transformation $D_1 \longrightarrow D_2$ for d_2^1

A similar analysis for d_2^1 in S_1 shows that d_2^1 becomes either d_0^2 or d_1^2 with equal probabilities, which can be figured as shown in Figure 4.5.

Table 4.3 lists the transformation of D_1 to D_2 for distance values ranging from 0 to 9. As the table shows clearly, this transformation can be summarized as follows: $\forall i \geq 1$, if i is odd then $d_i^1 \longrightarrow d_{\lfloor i/2 \rfloor}^2$, otherwise $d_i^1 \longrightarrow d_{(i/2)}^2$ *or* $d_{(i/2-1)}^1$ with equal probability. The first case implies that only d_{2i+1}^1 in S_1 can transform to d_i^2 in S_2, and

all distances d^1_{2i+1} transform to d^2_i. The second case implies that both distances d^1_{2i} and d^1_{2i+2} in S_1 can transform to distance d^2_i in S_2, and half of these distances transform to d^2_i. Equation 1, which takes both cases into account using a probabilistic approach, formulates this relation accordingly. Ignoring the second case and assuming that always the first case takes place yields a different formula for the transformation. Although it seems not intuitive to ignore the second case, the second estimation that counts only the first case gives reasonably good results if the base stream is long enough. However, the first approximation gives even better results than the second one.

D_1	D_2
0	vanish ; 0
1	0
2	0 ; 1
3	1
4	1 ; 2
5	2
6	2 ; 3
7	3
8	3 ; 4
9	4

Table 4.3: Transformation $D_1 \longrightarrow D_2$

$$D_2[i] = \frac{D_1[2 \cdot i]}{2} + D_1[2 \cdot i + 1] + \frac{D_1[2 \cdot i + 2]}{2} \tag{1}$$

4.2 Estimation at Time Granularity 3

Now, we can examine how the distance array D_1 changes and becomes D_3 during the transformation of event stream S_1 (at granularity 1) to event stream S_3 (at granularity 3). The only difference from the transformation to an event stream at time granularity 2 is the length of the blocks in S_1, which now is three and we thus have three different cases for each distance value in D_1. This is shown in Figure 4.6.

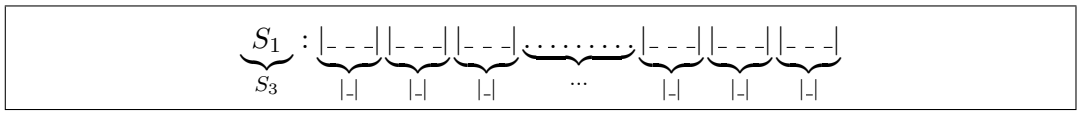

Figure 4.6: Transformation with Granularity 3

Again a distance d_0^1 indicates a "11" subsequence of length 2 in S_1. Three cases to consider during the transformation of S_1 to S_3 are: Both of 1s can be in the same block with 2 different possible placement in that block, or they can be in different successive blocks. As shown in Figure 4.7, the first two cases yield a single 1 in S_3, which means that d_0^1 vanishes in D_3; while the third one preserves both 1s in S_3, i.e., d_0^1 in S_1 transforms to d_0^3 in S_3. Thus, a zero distance in S_1 vanishes in S_3 with probability 2/3, and becomes a zero distance in S_3 with probability 1/3.

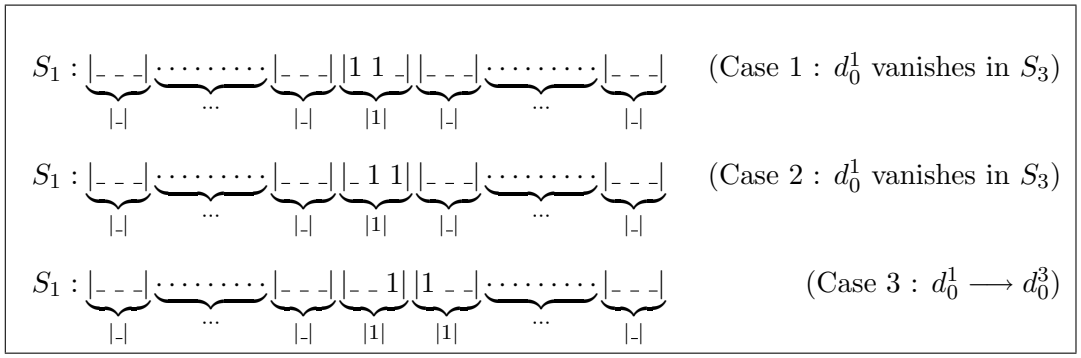

Figure 4.7: Transformation $D_1 \longrightarrow D_3$ for d_0^1

The same analysis for distances 1 to 3 are given in Figures 4.8, 4.9 and 4.10, respectively, without any further explanation. Table 4.4 lists the transformation of D_1 to D_3 for distance values 0 to 9 with associated probabilities given in parentheses. Equation 2 formulates this relation between D_1 and D_3.

$$D_3[i] = D_1[3 \cdot i]\frac{1}{3} + D_1[3 \cdot i + 1]\frac{2}{3} + D_1[3 \cdot i + 2]\frac{3}{3} + D_1[3 \cdot i + 3]\frac{2}{3} + D_1[3 \cdot i + 4]\frac{1}{3} \quad (2)$$

D_1	D_3
0	vanish (2/3) ; 0 (1/3)
1	vanish (1/3) ; 0 (2/3)
2	0
3	0 (2/3) ; 1 (1/3)
4	0 (1/3) ; 1 (2/3)
5	1
6	1 (2/3) ; 2 (1/3)
7	1 (1/3) ; 2 (2/3)
8	2
9	2 (2/3) ; 1 (1/3)

Table 4.4: Transformation $D_1 \longrightarrow D_3$

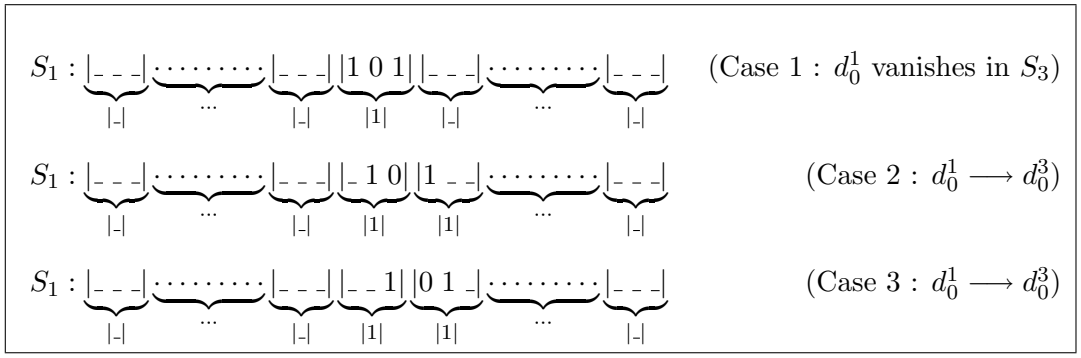

Figure 4.8: Transformation $D_1 \longrightarrow D_3$ for d_1^1

4.3 Estimation at Coarser Granularities

Consider the transformation of the base event stream S_1 to event stream S_g with an arbitrary time granularity $g \geq 2$. Instead of analyzing how a particular distance d_i^1 in S_1 transforms to a distance d_j^g in S_g, we find which distances in S_1 can transform to a particular distance d_j^g in S_g and their corresponding probabilities.

Let g be the target granularity and t be a distance in S_g, where $0 \leq t \leq MaxDist_g$. Let R be the possible distance values in S_1 that can transform to d_t^g. Formally, $R = \{d' \mid d' \in D_1, d' \longrightarrow t\}$. Using our block structure, this transformation can be figured as in Figure 4.11. Each block is of length g, and d' must be at least $(t \cdot g)$ in order to have

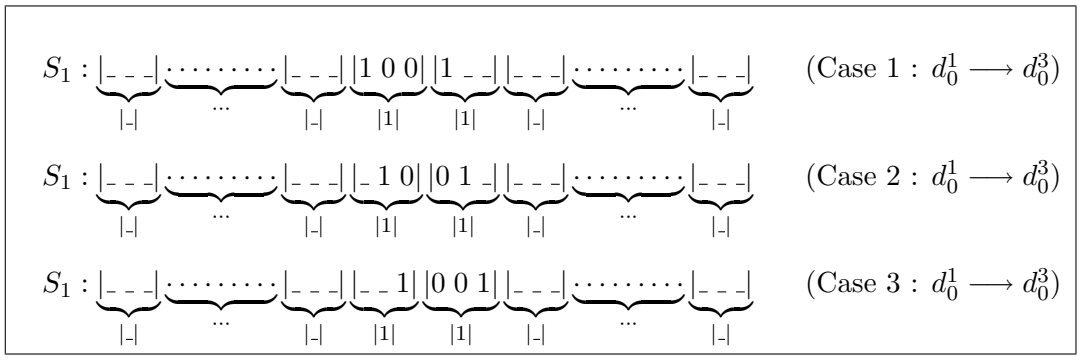

Figure 4.9: Transformation $D_1 \longrightarrow D_3$ for d_2^1

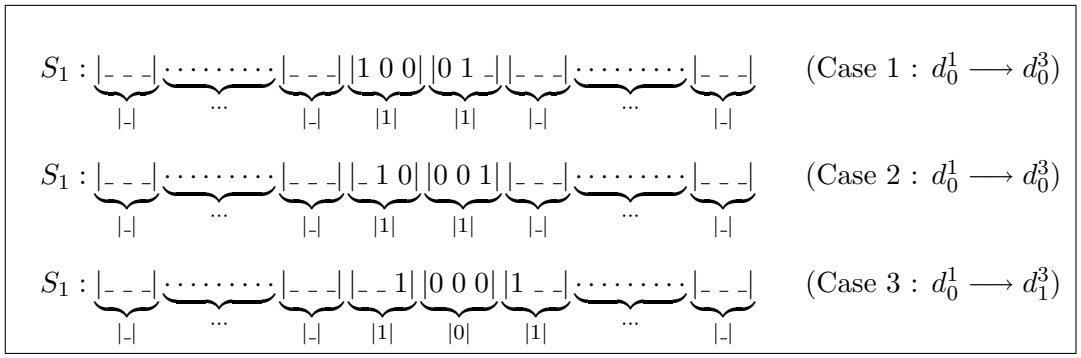

Figure 4.10: Transformation $D_1 \longrightarrow D_3$ for d_3^1

d_t^g. This represents the best case, because in order to have $d' = (t \cdot g) \longrightarrow t$, the d' zeros in S_1 must start at exactly $b_1[1]$, which is the first time-tick of the block b_1. The worst case occurs when the d' zeros start at $b_0[2]$ and ends at $b_{t+1}[g-1]$, spanning $(t \cdot g + 2 \cdot g - 2)$ time-ticks. Adding one more zero to d' zeros would fill either of the blocks b_0 and b_{t+1} and d' would become at least d_{t+1}^g in D_g. Thus, we have $R = [t \cdot g\ ,\ t \cdot g + 2 \cdot g - 2]$ and $R \subset Z$.

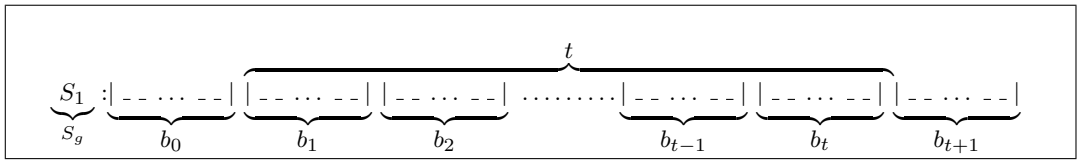

Figure 4.11: Transformation with Granularity g

Now, let us find the probability of $(d' \longrightarrow d_t^g)$ for each value in R, which will be referred to by $p(d' = i \longrightarrow t)$. As we have already mentioned above, the probability of $d' = (t \cdot g)$ is $1/g$ since the d' zeros must start at the first time-tick of any block of length g. For $d' = (t \cdot g + 1)$, the d' zeros can start at the points $b_0[g]$ or $b_1[1]$. The first case spans the points between $b_0[g]$ and $b_t[g]$, while the second one spans the points $b_1[1]$ to $b_{t+1}[1]$. Any other start point would leave either of the blocks b_0 or b_t unfilled and violate the transformation $d' \longrightarrow t$. Thus, only two out of g points are acceptable and $p(t \cdot g + 1 \longrightarrow t) = 2/g$. Similar analysis on different values of d' can be made to show the following relation:

$$\forall d' = t \cdot g + j \ , \ 0 \leq j \leq g - 1 \qquad \Rightarrow \quad p(d' \longrightarrow t) = \frac{j+1}{g} \tag{3}$$

Substituting $(t + 1)$ for (t) in Equation 3 gives

$$\forall d' = (t+1) \cdot g + j \ , \ 0 \leq j \leq g - 1 \qquad \Rightarrow \quad p(d' \longrightarrow t+1) = \frac{j+1}{g} \tag{4}$$

$$\forall d' = (t+1) \cdot g + j \ , \ 0 \leq j \leq g - 1 \qquad \Rightarrow \quad p(d' \longrightarrow t) = 1 - \frac{j+1}{g} \tag{5}$$

$$\forall d' = t \cdot g + g + j \ , \ 0 \leq j \leq g - 2 \qquad \Rightarrow \quad p(d' \longrightarrow t) = \frac{g-j-1}{g} \tag{6}$$

Equation 4 is straightforward. Equation 5 uses the fact that $\forall d' = t \cdot g + g + j \ , \ 0 \leq j \leq g - 1$, either $d' \longrightarrow t$ or $d' \longrightarrow t + 1$. Therefore, $p(d' \longrightarrow t) = 1 - p(d' \longrightarrow t + 1)$. Equation 6 is just the more explicit form of Equation 5. The combination of Equations 3 and 5 given below spans the whole R and is the desired generalization of Equations 1

and 2 to coarser time granularities.

$$D_g[i] = \sum_{j=0}^{g-1} D_1[g \cdot i + j] \, \frac{j+1}{g} + \sum_{j=1}^{g-1} D_1[g \cdot i + g - 1 + j] \, \frac{g-j}{g} \tag{7}$$

4.4 Calculation of Event Counts Using the Distance Matrix

Once we have estimated the distance array D_g, the count of 1s in S_g is found as follows: for $1 \leq i \leq D_g.length$, $D_g[i]$ gives the number of distances of length i, i.e., the number of blocks of successive zeros of length i. Thus, the total number of zeros in S_g is

$$Count_g(0) = \sum_{i=1}^{D_g.length} i * D_g[i]$$

Then, the total count of 1s in D_g is given by

$$Count_g(1) = D_g.length - Count_g(0)$$

where $D_g.length = \lceil n/g \rceil$ and n is the length of S_1.

4.5 Incremental Maintenance

The distance array can be updated incrementally for streaming data. At each time tick, a variable, say *current*, is updated according to the current state of the event. Whenever the event state is 1, the corresponding distance value $D_1[current]$ is incremented by one, and *current* is set to zero. For each 0-state, *current* is incremented by one. Equation 3 and 6 clearly show that the count estimations at granularity g can be incrementally updated as follows:

$$\begin{aligned} D_g[i] &\mathrel{+}= \frac{j+1}{g} \\ D_g[i-1] &\mathrel{+}= \frac{g-j-1}{g} \end{aligned} \tag{8}$$

where $current = g \cdot i + j$.

4.6 Special Cases

Before applying the method to an input event stream S, two similar special cases should be considered. Depending on the implementation, one or both of these cases may degrade the accuracy of the method. Suppose that the values that appeared last in the stream S are one or more zeros, i.e., $S_1 : [\ \cdots\cdots\cdots\ ,\ 1\ ,\ \underbrace{0\ \cdots\ 0}_{d_k}\ ]$, where $d_k \geq 1$. And suppose that during the distance generation phase, the d_k zeros at the end are treated as a distance of length d_k, and $D[d_k]$ is incremented by 1, where D is the distance array. Then, since a distance is defined as the total number of successive 0s between two 1s in the stream, this kind of implementation implicitly (and erroneously) assumes the presence of a 1 at the end of the stream, just after the d_k 0s. This misbehavior results in an overestimate of the count of the event at coarser granularities by 1. Although an overestimate by 1 may seem insignificant, this can cause relatively high error rates for extremely sparse event streams or at sufficiently high granularities where the frequency of the event is very low.

The same effect could be made by one or more 0s at the beginning of the event stream, where the implicit (and erroneous) assumption would be the presence of a 1 before the 0s at the beginning of the stream. To prevent such misbehavior, the start and end of the stream should be considered separately from the rest, or the stream should be trimmed off from both ends during the preprocessing phase, so that it starts and ends with a 1.

4.7 Time and Space Requirements

In the preprocessing phase, we scan the base stream once and populate the distance array D_1, which takes $O(n)$ time and uses $O(max_1)$ space, where n is the length of the base stream S_1 and max_1 is the maximum distance at base granularity. For any particular

granularity g, we make the transformation $D_1 \longrightarrow D_g$ which takes $O(max_g \times g)$ time where max_g is the maximum distance at granularity g. Indeed, max_g is the length of D_g and is less than or equal to $\lceil max_1/g \rceil$. The space required to store the distance distribution D_g is also proportional to max_g. Thus, the run-time of our method is $O(n+max_g \times g) = O(n+(max_1/g) \times g) = O(n+max_1) = O(n)$, and the memory required is $O(max_g)$ if the stream is not stored after the distance distribution is constructed, and it is $O(n + max_g) = O(n)$ otherwise.

We use histograms to store the distance distributions of the event streams at base granularity. As explained before, various histogram types have been introduced and their construction and maintenance issues have been well studied so far, especially in the context of query result size estimation. We used *end-biased* histograms, where some of the values with the highest and lowest frequencies are stored in individual buckets, and the remaining values with middle frequencies are grouped in one single bucket.

5 Performance Experiments

In this section, we give some experimental results conducted on real life data. We used the data set gathered in [5] and available at http://cs.bilkent.edu.tr/~unala/stockdata. The data set is the closing prices of 439 stocks for 517 trading days between January 3, 1994, and January 11, 1996. We have used this data set to simulate event streams. For each stock in the data set, the price change percentages are calculated and partitioned into 7 categories: $(-\infty$,-5],(-5,-3], (-3,0], [0,0], (0,3], (3,5], (5,$\infty)$. Each category of price change for each stock is considered as a distinct event, yielding a total $439 \times 7 = 3073$ number of event types and $3073 \times 517 = 1,588,741$ distinct $< time-tick, eventstate >_{eventtype}$ pairs. For example, IBM_03 is an event type that represents a price change percentage of IBM stock that falls into (-3,0]. $< 200, 1 >_{IBM\_03}$

meaning that the event IBM_03 occurred on day 200 in the stream. If a stock is not exchanged for any reason on a particular business day, then all 7 events are registered as 0 for that stock on that day.

The machine we used for the experiments was a personal computer with a Pentium 4 1.4 GHz processor and 2 memory boards, each 64 MB RDRAM, totally 128 MB main memory.

In the experiments, we considered both single and multiple events (or eventsets). In Section 5.1 experimental results for a single event are presented. In Sections 5.2 and 5.3, multiple events are considered to show that our methods can also be generalized to eventsets. Frequencies of multiple events are predicted exactly the same way as single events, i.e., using the distance distributions for each event.

As mentioned before, the experiments we conducted show that the distribution of the distance between two occurrences of an event in a history tends to have high frequencies for some small distance values, and very low frequencies for the remaining larger values. Therefore, we use end-biased histograms, in which some of the values with the highest and lowest frequencies are stored in individual buckets, and the remaining values with middle frequencies are grouped in a single bucket.

5.1 Experiments for a Single Event

We first examined a single event in order to prove the accuracy of our method on finding the count (or frequency) of an event stream at coarser granularities. The count of an event stream at a particular granularity is equal to the number of time ticks at which the event occurred at that granularity. Table 5.5 shows the results of the experiment in which the event was defined as *no price change of McDonalds Corp. stock*. The first column gives the granularities at which the estimations are made. The next two columns specify

the actual count of the event at the corresponding granularity and the count estimated by our method, respectively. The last two columns give the absolute and relative errors of our estimations, respectively, with respect to the actual values. The frequency of the event at base granularity was 9.48% and the maximum distance was 72. Figure 5.12 plots the actual and estimated counts at multiple time granularities. Experiments conducted on a different set of real life data gave similar results, validating the accuracy of our method. The second data set also consists of stock exchange market closing prices, and is available at http://www.analiz.com/AYADL/ayadl01.html. The results obtained with this data set are not presented in this paper due to space limitations. Interested readers, however, can find detailed information about these experiments and their results in [31].

We then conducted 3 sets of experiments, each testing the behavior of the method with respect to 3 parameters: granularity, support threshold, and the number of events. In each experiment set, two of these parameters were held constant while several experiments were conducted for different values of the third parameter, and given a set of event streams, we estimated the frequent eventsets at granularity in concern. The following subsections present the results of these experiments.

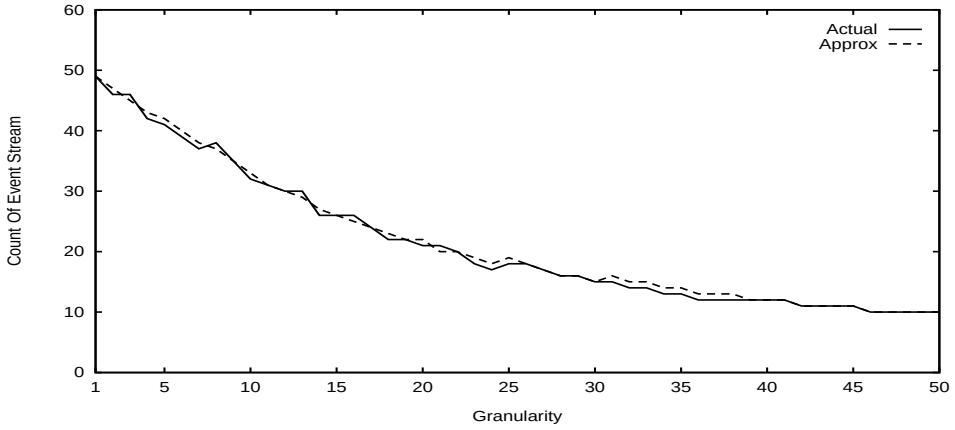

Figure 5.12: Count Estimation of a Single Event Stream at Multiple Time Granularities

g	Actual	Approx.	Abs_Err	Rel_Err (%)	g	Actual	Approx	Abs_Err	Rel_Err (%)
1	49	49	0	0	26	18	18	0	0
2	46	47	1	2,17	27	17	17	0	0
3	46	45	-1	-2,17	28	16	16	0	0
4	42	43	1	2,38	29	16	16	0	0
5	41	42	1	2,44	30	15	15	0	0
6	39	40	1	2,56	31	15	16	1	6,67
7	37	38	1	2,7	32	14	15	1	7,14
8	38	37	-1	-2,63	33	14	15	1	7,14
9	35	35	0	0	34	13	14	1	7,69
10	32	33	1	3,12	35	13	14	1	7,69
11	31	31	0	0	36	12	13	1	8,33
12	30	30	0	0	37	12	13	1	8,33
13	30	29	-1	-3,33	38	12	13	1	8,33
14	26	27	1	3,85	39	12	12	0	0
15	26	26	0	0	40	12	12	0	0
16	26	25	-1	-3,85	41	12	12	0	0
17	24	24	0	0	42	11	11	0	0
18	22	23	1	4,55	43	11	11	0	0
19	22	22	0	0	44	11	11	0	0
20	21	22	1	4,76	45	11	11	0	0
21	21	20	-1	-4,76	46	10	10	0	0
22	20	20	0	0	47	10	10	0	0
23	18	19	1	5,56	48	10	10	0	0
24	17	18	1	5,88	49	10	10	0	0
25	18	19	1	5,56	50	10	10	0	0

Table 5.5: Summary of the experiments conducted using a single event

5.2 Granularity

The experiments of this section were conducted with varying values of the *granularity* parameter. For each granularity value, using our approximation algorithm we estimated the eventsets that are frequent in the event stream.

Table 5.6 reports the experimental results. For each granularity, the second column gives the number of *actual* frequent eventsets, and the third column presents the number of *estimated* eventsets. The last two columns report the number of *under* and *over*

estimated eventsets, respectively. An under-estimated eventset is one that is in the set of actual frequent eventsets but not found by the approximation algorithm. On the other hand, an over-estimated eventset is one that is found to be a frequent eventset but is not really frequent.

As the granularity increases, the total number of frequent eventsets decreases. We used absolute support threshold values rather than relative ones. Since the support threshold is held constant and the count of a particular event decreases at coarser granularities, the number of frequent eventsets of length 1 (C_1) decreases as well. The candidates of length 2 are generated by the combinations of frequent eventsets of length 1. Thus, a constant decrease in C_1 yields an exponential reduction in the total candidate eventsets of length 2, which in turn yields a reduction in the total number of frequent eventsets of length 2. This is similar for coarser granularities and does explain the pattern in Figure 5.13. Note that the reduction does not follow an exact pattern and is fully dependent on the dataset.

Granularity	Actual	Approx.	Under	Over
2	445	443	15	13
3	309	318	6	15
4	204	207	11	14
5	124	122	10	8
6	75	77	1	3
7	49	50	2	3
8	11	9	4	2
9	1	0	1	0
10	0	0	0	0

Table 5.6: Summary of the experiments conducted for varying granularity values

The absolute errors of over/under estimations fluctuate around a linearly decreasing pattern. Figure 5.14 plots the absolute errors at different granularities and clearly shows the fluctuating pattern. Figures 5.15 and 5.16 show the regressions of over-estimation

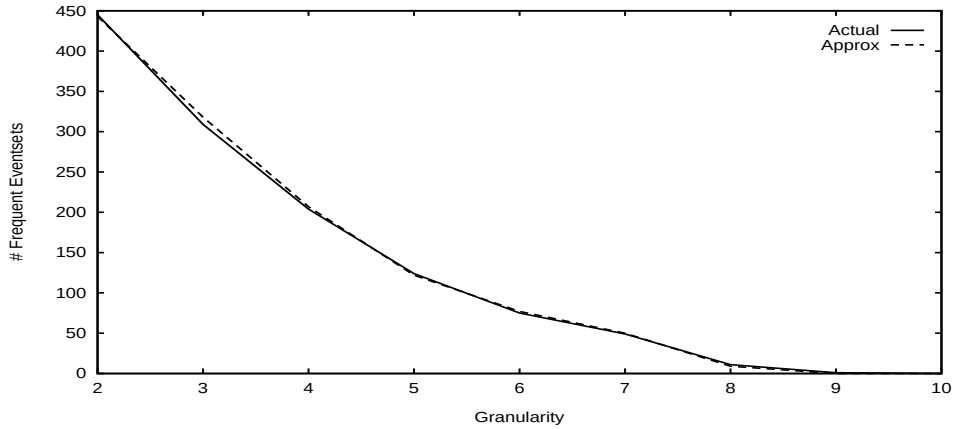

Figure 5.13: Frequent Eventset Counts vs. Granularity

and under-estimation errors, respectively, and are given to make the overall linear pattern more clear. The local fluctuations arise from the distance distributions of the streams in the dataset.

The relative errors (RE), given in Equations 9 and 10, are plotted in Figure 5.17. While RE_{Over} gives the ratio of the total estimated eventsets that are indeed infrequent, RE_{Under} gives the ratio of the total actual frequent eventsets that are not estimated by the method as frequent. As Figure 5.17 shows clearly, the relative errors stay below 8% except for the granularities at which the total number of frequent eventsets is very small, which gives higher relative errors for small absolute errors. The sharp increase in the Figure 5.17, for example, is a good example of such a situation, where even a small absolute error gives high relative error because of very small frequent eventset count.

$$RE_{Over} = \frac{\#Over\ Estimations}{\#EstimatedEventsets} \tag{9}$$

$$RE_{Under} = \frac{\#Under\ Estimations}{\#ActualFrequentEventsets} \tag{10}$$

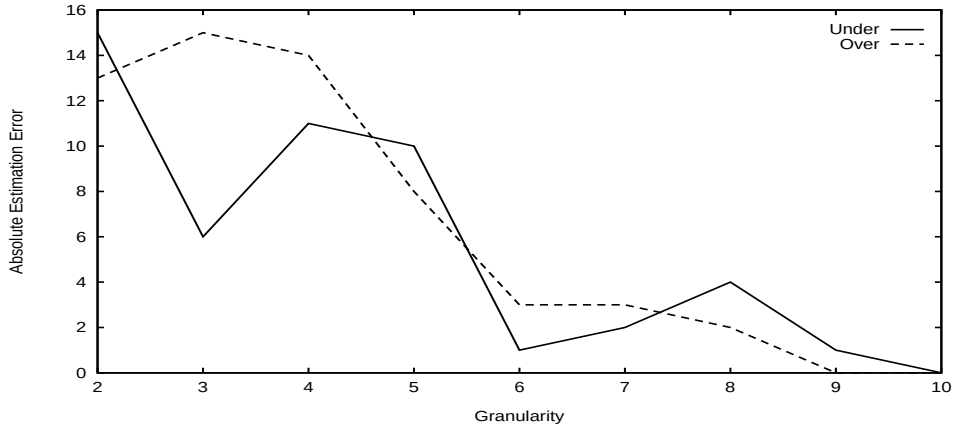

Figure 5.14: Absolute Estimation Errors vs. Granularity

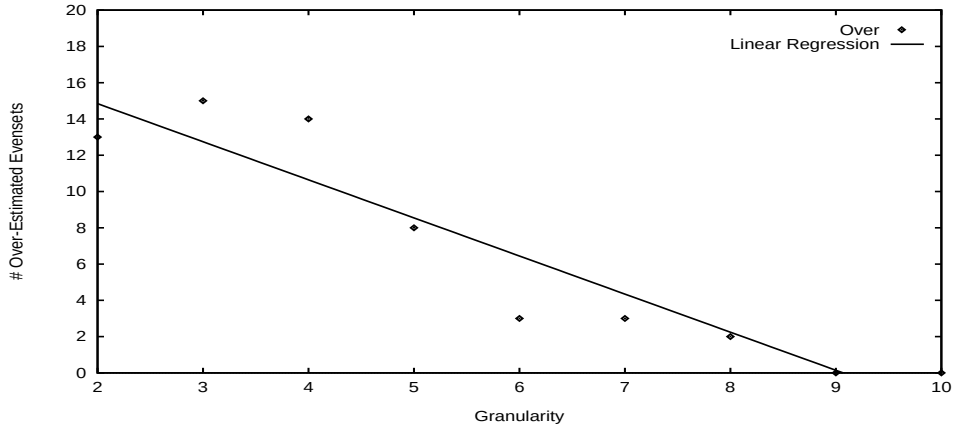

Figure 5.15: Linear Regression of Over-Estimation Errors vs. Granularity

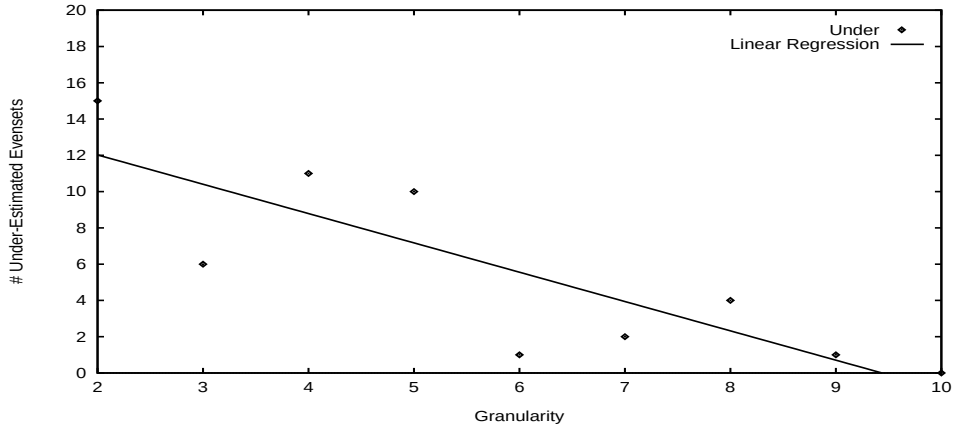

Figure 5.16: Linear Regression of Under-Estimation Errors vs. Granularity

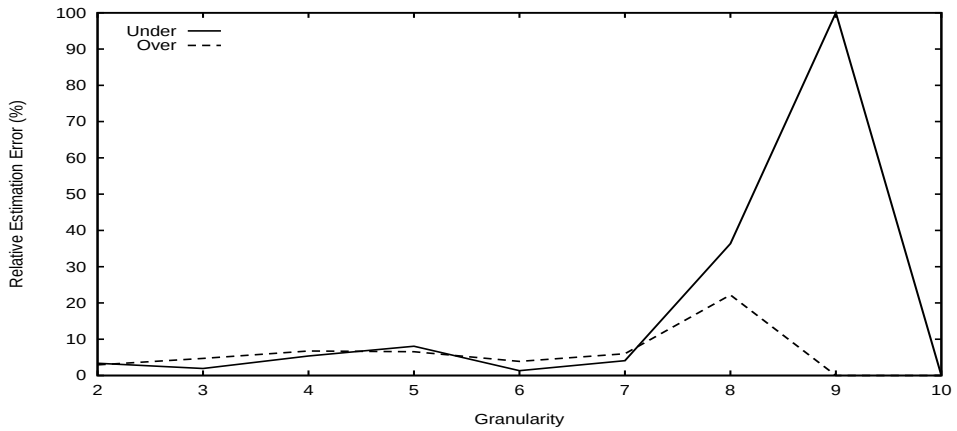

Figure 5.17: Relative Estimation Errors vs. Granularity

5.3 Support Threshold

We conducted several experiments under varying values of the *support threshold*. One typical experiment is summarized in Table 5.7. As the support threshold value increases, the number of frequent eventsets of length 1 decreases. This yields a reduction in candidate eventset count, which in turn causes a reduction in the total number of frequent eventsets. The experiments conducted produced similar patterns for total number of frequent eventsets, and the results of one of these experiments are depicted in Figure 5.18.

The errors of over/under estimations follow the same pattern (Figure 5.19) as in experiments conducted at different granularities and given in the previous subsection. The absolute errors fluctuate around a linearly decreasing pattern (Figures 5.20 and 5.21), which is again due to the distance distributions of the dataset. However, the relative errors, as shown in Figure 5.22 stay below 10% except for the support threshold values where the total number of frequent eventsets is very small.

Support	Actual	Approx.	Under	Over
35	1061	1081	27	47
40	683	704	23	44
45	383	399	25	41
50	172	190	10	28
55	66	74	10	18
60	8	8	2	2
65	0	0	0	0
70	0	0	0	0

Table 5.7: Summary of the experiments conducted for varying support thresholds

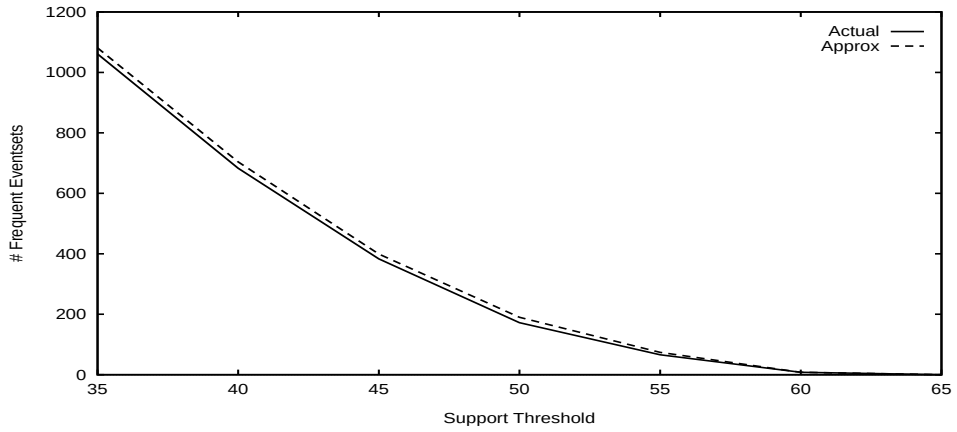

Figure 5.18: Frequent Eventset Counts vs. Support Threshold

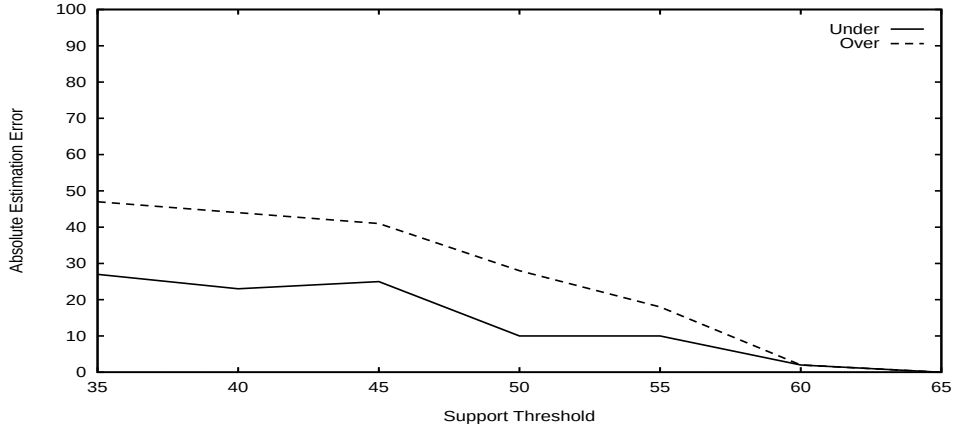

Figure 5.19: Absolute Estimation Errors vs. Support Threshold

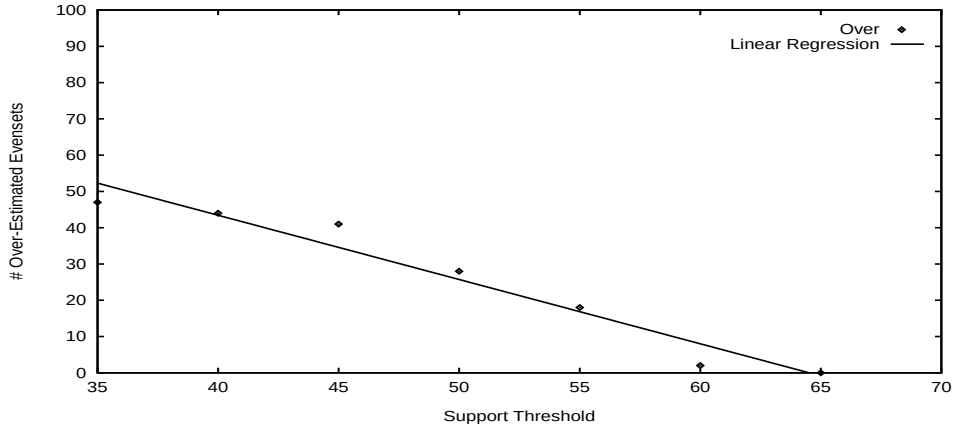

Figure 5.20: Linear Regression of Over-Estimation Errors vs. Support Threshold

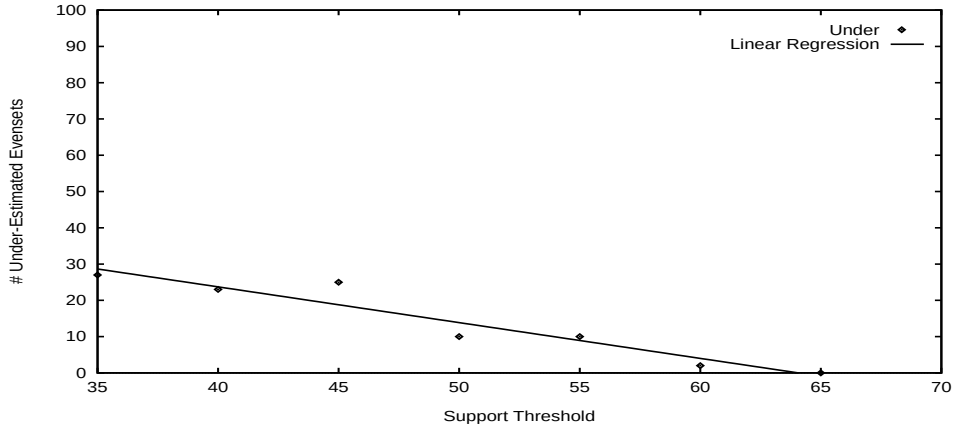

Figure 5.21: Linear Regression of Under-Estimation Errors vs. Support Threshold

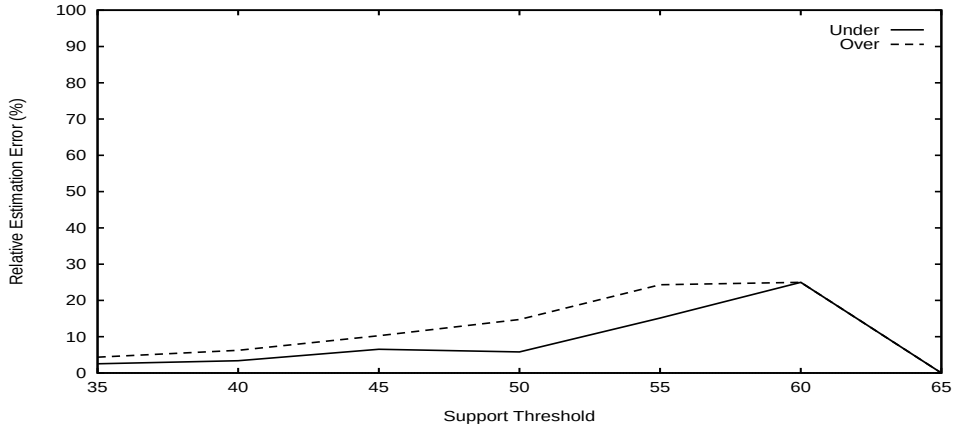

Figure 5.22: Relative Estimation Errors vs. Support Threshold

5.4 Number of Events

The last set of experiments was conducted under varying values of *event counts*. We increased the number of events by incrementally adding new event streams to the event set. A typical experiment is summarized in Table 5.8.

The absolute and relative errors again showed similar behaviors as in the previous experiment sets. The number of absolute errors increases linearly as the event count increases, and the percentage of relative errors stays under $5 - 6\%$ except for very small event counts, where small frequent eventset counts yield high relative errors for reasonable absolute errors.

Figure 5.23 plots both the actual and estimated numbers of frequent eventsets for varying numbers of event streams. Figure 5.24 shows the counts of over-estimated and under-estimated eventsets, which are also plotted in Figure 5.25 and Figure 5.26, respectively, along with their corresponding linear regressions. These figures are provided just to verify the linear patterns observed in the previous experiments. Finally, Figure 5.27 presents the relative estimation errors.

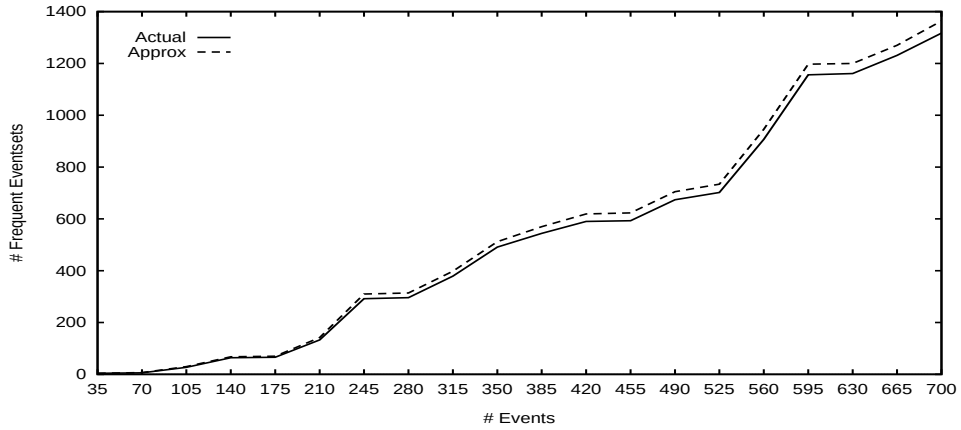

Figure 5.23: Estimation Error Counts vs. Number of Events

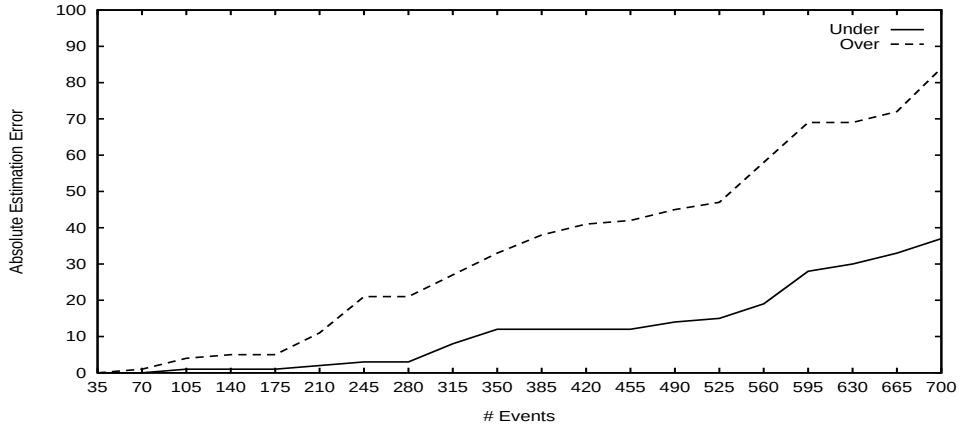

Figure 5.24: Absolute Estimation Errors vs. Number of Events

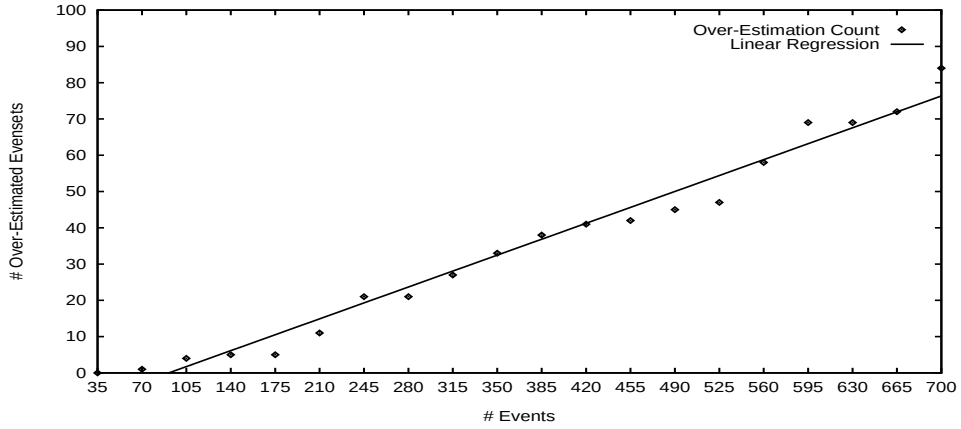

Figure 5.25: Linear Regression of Over-Estimation Errors vs. Number of Events

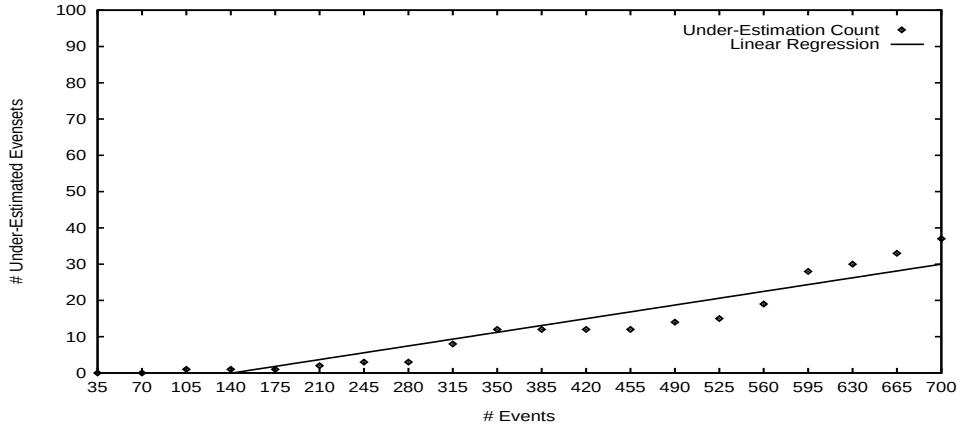

Figure 5.26: Linear Regression of Under-Estimation Errors vs. Number of Events

# Events	Actual	Approx.	Under	Over
35	4	4	0	0
70	6	7	0	1
105	27	30	1	4
140	64	68	1	5
175	66	70	1	5
210	133	142	2	11
245	292	310	3	21
280	296	314	3	21
315	379	398	8	27
350	491	512	12	33
385	544	570	12	38
420	590	619	12	41
455	593	623	12	42
490	674	705	14	45
525	702	734	15	47
560	907	946	19	58
595	1156	1197	28	69
630	1161	1200	30	69
665	1231	1270	33	72
700	1317	1364	37	84

Table 5.8: Summary of the experiments conducted for varying number of event streams

The experiments discussed above and many others[1] conducted for different parameter values proved the accuracy of our method in estimating the count of a stream at coarser granularities. While the number of absolute errors decreases linearly, the percentage of relative errors stays under reasonably small values except for the points where frequent eventset counts are small. The experiment results show that the ratio of relative errors rarely exceeds 10% and most of the time does not exceed 5% if the number of frequent eventsets is large enough.

[1]The results are not presented due to space limitations.

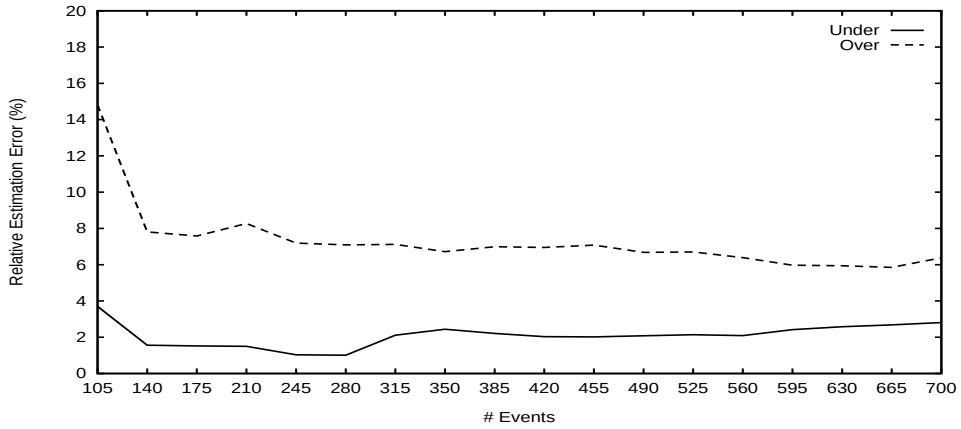

Figure 5.27: Relative Estimation Errors vs. Number of Events

6 Prediction

The statistical information collected about the frequency and distribution of the event occurrences can also be used for estimation of the event at future time ticks or at previous time ticks at which the data is missing. This can be done at the base granularity or any other coarser time granularities with the help of corresponding distance vectors. For any time tick t, let s_t be the distance from that time tick to the last occurrence of the event in the interval $[0, t]$. Then, we have $s_0 = 0$, and the state $s_t = n$ can be followed only by the states $s_{t+1} = 0$ if the event occurs at time $t + 1$, or $s_{t+1} = n + 1$ otherwise. This process satisfies the Markov Property and is therefore a Markov Chain. The state transition diagram of the system is given in Figure 6.28, where the real transition probabilities p

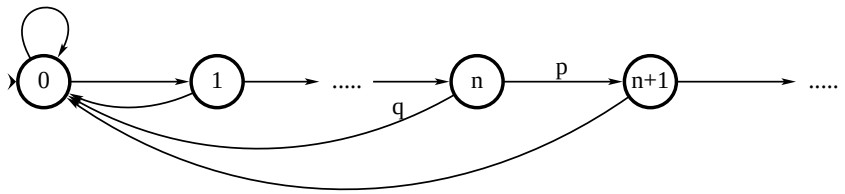

Figure 6.28: State Diagram of the Markov Chain

and q can be estimated using the distance histogram that stores the numbers of distance

values. Observing a distance $d \geq n+1$ is equivalent to starting from state 0, making a rightwards transition at each time tick until we reach the state $s = d$, and finally jumping back to state 0 in our Markov Chain given in Figure 6.28. Then, whenever we have a distance $d > n$, we are guaranteed to make the transition $n \rightarrow n+1$. Similarly, whenever we have a distance $d = n$, we will definitely make the transition $n \rightarrow 0$. Then, the state $s = n$ is visited for all distances $d \geq n$. While the exact values of p and q are not known, they can be approximated using the number of transitions observed through the event series in concern so far. p can be approximated by the ratio of the total number of transitions $n \rightarrow n+1$ to the total number of visits to the state $s = n$. Similarly, q can be approximated by the ratio of the total number of transitions $n \rightarrow 0$ to the total number of visits to the state $s = n$. Since the transition $n \rightarrow n+1$ is made for all distances $d > n$, the total number of times this transition is made equals to the summation $\sum_{i>n} D_g[i]$. Similarly, the total number of times the transition $n \rightarrow 0$ is made equals $D_g[n]$, and the total number of visits to the state $s = n$ equals to the summation $\sum_{i \geq n} D_g[i]$. Then, we have

$$p = \frac{\sum_{i>n} D_g[i]}{\sum_{i \geq n} D_g[i]} \tag{11}$$

and

$$q = \frac{D_g[n]}{\sum_{i \geq n} D_g[i]} \tag{12}$$

Now, suppose that the number of time ticks after the last occurrence of the event is equal to n, $n \geq 0$, and we want to predict the behavior of the event in the next time tick. The probability of having a 1 in the next tick is equivalent to the probability of the transition from state n to 0, which is simply q. That is, q gives the probability that the event occurs in the next time tick.

For various reasons, some of the values of the stream might not have been recorded.

As mentioned above, the same idea can be applied to predict the missing information in the past time ticks.

7 Conclusion

We introduced a probabilistic approach to answer count queries for 0/1 event streams at arbitrary time granularities. We examined the distance distribution of an event at base granularity, used the probabilities of the distance transformations to approximate the distance distribution of the event at any coarser time granularity, and used this approximation to estimate the count of the event at the granularity in concern.

The experiments conducted on real-life data proved that most of the time our approach gives reasonably good estimations with error rates less than 5%. Our method runs in $O(n)$ time and uses $O(n)$ space, where n is the length of the base event stream. The results of the experiments conducted on different real-life data prove the accuracy of our method for count estimation at multiple time granularities.

The data structure we used is a histogram that stores the possible distance values and the corresponding distance counts in the base event stream. A future research issue that we are planning to investigate is the use of samples of the base event stream to construct an approximate distance histogram, which improves the runtime while decreasing the accuracy of the estimations. The tradeoff between speed and accuracy can be examined in detail.

Another future research direction is to study different histogram classes to find the best one for storing the distance distribution. One possible scheme is to store the distance values that have the same frequencies in the same bucket, and others in individual buckets. Another method can be to store the distance values with high and low frequencies in individual buckets and the remaining ones in a single bucket. In each case, the tradeoff between space and accuracy should be analyzed carefully.

References

[1] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, Proceedings of the ACM SIGMOD Conference on Management of Data(1993) 207-216.

[2] M. Atallah, R. Gwadera, W. Szpankowski, Detection of Significant Sets of Episodes in Event Sequences: Algorithms, Analysis and Experiments, Proceedings of the 4^{th} IEEE International Conference Data Mining (2004) 3-10.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and Issues in Data Streams, Proceedings of the ACM PODS Symposium on Principles of Database Systems (2002), 1-16.

[4] C. Bettini, C. Dyreson, W. Evans, R. Snodgrass, X. Wang, A Glossary of Time Granularity Concepts, in: Temporal Databases: Research and Practice, Lecture Notes in Computer Science 1399, O. Etzion, S. Jajodia, S. Sripada (Ed.), Springer-Verlag, 1998, pp. 406-411.

[5] C. Bettini, S. Jajodia, J. Lin, Discovering frequent event patterns with multiple granularities in time sequences, IEEE Transactions on Knowledge and Data Engineering 10(2) (1998) 222-237.

[6] J.F. Boulicaut, A. Bykowski, C. Rigotti, Free-Sets: A Condensed Representation of Boolean Data for the Approximation of Frequency Queries, Data Mining and Knowledge Discovery 7(1) (2003) 5-22.

[7] S. Chaudhuri, R. Motwani, V. Narasayya, Random sampling for histogram construction: How much is enough? Proceedings of ACM SIGMOD International Conference on Management of Data (1998) 436-447.

[8] G. Das, K-I Lin, H. Mannila, G. Ranganathan, P. Smyth, Rule discovery from time series, Proceedings of the 4^{th} International Conference on Knowledge Discovery and Data Mining (1998) 16-22.

[9] A. Dobra, M. Garofalakis, J. Gherke, R. Rastogi, Processing Complex Aggregate Queries over Data Streams, Proceedings of the ACM SIGMOD Conference on Managament of Data (2002) 61-72.

[10] D. Gao, J.A.G. Gendrano, B. Moon, R.T. Snodgrass, M. Park, B.C. Huang, J.M. Rodrigue, Main Memory-Based Algorithms for Efficient Parallel Aggregation for Temporal Databases, Distributed and Parallel Databases Journal 16(2) (2004) 123-163.

[11] M. Garofalakis, J. Gehrke, R. Rastogi, Querying and Mining Data Streams: You Only Get One Look, Tutorial in ACM SIGMOD Conference (2002) 635-635.

[12] J. Gendrano, B. Huang, J. Rodrigue, B. Moon, R. Snodgrass, Parallel Algorithms for Computing Temporal Aggregates, Proceedings of the 15^{th} International Conference on Data Engineering (1999)418-427.

[13] P.B. Gibbons, Y. Matias, V. Poosala, Fast incremental maintenance of approximate histograms, Proceedings of the 23^{rd} Conference on Very Large Databases (1997) 466-475.

[14] S. Govindarajan, P. Agarwal, L. Arge, CRBTree: An Efficient Indexing Scheme for Range Aggregate Queries, Proceedings of the 9^{th} International Conference on Database Theory (2003) 143-157.

[15] R. Gwadera, M. Atallah, W. Szpankowski, Reliable detection of episodes in event sequences, Proceedings of the 3^{rd} IEEE International Conference Data Mining (2003) 67-74.

[16] P.J. Haas, J.F. Naughton, S. Seshadri, L. Stokes, Sampling-based estimation of the number of distinct values of an attribute, Proceedings of the 21^{st} Conference on Very Large Databases (1995) 311-322.

[17] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, Proceedings of ACM-SIGMOD International Conference on Management of Data (2000)1-12.

[18] Y. Ioannidis, V. Poosola, Balancing Histogram Optimality and Practicality for Query Result Size Estimation, Proceedings of ACM SIGMOD International Conference on the Management of Data (1995) 233-244.

[19] R.P. Kooi, The optimization of queries in relational databases, PhD thesis, Case Western Reserve University, September 1980.

[20] I.F.V. Lopez, R.T. Snodgrass, B. Moon, Spatiotemporal Aggregate Computation: A Survey, IEEE Transactions on Knowledge and Data Engineering 17(2) (2005) 271-286.

[21] H. Mannila, P. Smyth, Approximate query answering using frequent sets and maximum entropy, Proceedings of the 16^{th} International Conference on Data Engineering (2000) 309.

[22] H. Mannila, H. Toivonen, Discovering generalized episodes using minimal occurrences, Proceedings of the 2^{nd} International Conference on Knowledge Discovery and Data Mining (1996) 146-151.

[23] H. Mannila, H. Toivonen, A.I. Verkamo, Discovering Frequent Episodes in Sequences, Proceedings of the 1^{st} International Conference on Knowledge Discovery and Data Mining (1995) 210-215.

[24] B. Moon, I. Lopez, V. Immanuel, Scalable Algorithms for Large Temporal Aggregation, Proceedings of the 16^{th} International Conference on Data Engineering (2000), 145-154.

[25] B. Özden, S. Ramaswamy, A. Silberschatz, Cyclic Association Rules, Proceedings of the 40^{th} International Conference on Data Engineering (1998) 412-421.

[26] D. Pavlov, H. Mannila, P. Smyth, Beyond Independence: Probabilistic Models for Query Approximation on Binary Transaction Data, IEEE Transactions on Knowledge and Data Engineering 15(6) (2003) 1409-1421.

[27] G. Piatetsky-Shapiro, C. Connell, Accurate estimation of the number of tuples satisfying a condition, Proceedings of ACM SIGMOD International Conference on the Management of Data (1984) 256-276.

[28] V. Poosola, Y. Ioannidis, P. Haas, E. Shekita, Improved histograms for selectivity estimation of range predicates, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data(1996) 294-305.

[29] Y. Saygın, Ö. Ulusoy, Exploiting Data Mining Techniques for Broadcasting Data in Mobile Computing Environments, IEEE Transactions on Knowledge and Data Engineering 14(6) (2002) 1387-1399.

[30] Y. Tao, D. Papadias, C. Faloutsos, Approximate Temporal Aggregation, Proceedings of the 20^{th} International Conference on Data Engineering (2004) 190-201.

[31] A. Ünal, Y. Saygın, Ö. Ulusoy, Processing Count Queries over Event Streams at Multiple Time Granularities, Bilkent University Technical Report BU-CE-0504. Available at http://www.cs.bilkent.edu.tr/tech-reports/2005/BU-CE-0504.pdf.

[32] J. Yang, J. Widom, Incremental Computation and Maintenance of Temporal Aggregates, Proceedings of the 17^{th} International Conference on Data Engineering (2001) 51-60.

[33] D. Zhang, D. Gunopulos, V.J. Tsotras, B. Seeger, Temporal and Spatio-Temporal Aggregations over Data Streams Using Multiple Time Granularities, Information Systems 28(1-2) (2003) 61-84.

[34] D. Zhang, A. Markowetz, V.J. Tsotras, D. Gunopulos, B. Seeger, Efficient Computation of Temporal Aggregates with Range Predicates, Proceedings of the ACM PODS Symposium on Principles of Database Systems (2001) 237-245.