Efficient Trade-Off between Speed Processing and Accuracy in Summarizing Data Streams

Nesrine Gabsi[1,2], Fabrice Clérot[1], and Georges Hébrail[2]

[1] France Télécom RD, 2, avenue P. Marzin 22307 Lannion, France
fabrice.clerot@orange-ftgroup.com
[2] Institut TELECOM, TELECOM ParisTech, 46 Rue Barrault 75013 Paris, France
{nesrine.gabsi,georges.hebrail}@enst.fr

Abstract. Data streams constitute the core of many traditional (e.g. financial) and emerging (e.g. environmental) applications. The sources of streams are ubiquitous in daily life (e.g. web clicks). One feature of these data is the high speed of their arrival. Thus, their processing entails a special constraint. Despite the exponential growth in the capacity of storage devices, it is very expensive - even impossible - to store a data stream in its entirety. Consequently, queries are evaluated only on the recent data of the stream, the old ones are expired. However, some applications need to query the whole data stream. Therefore, the inability to store a complete stream suggests the storage of a compact representation of its data, called summaries. These structures allow users to query the past without an explosion of the required storage space, to provide historical aggregated information, to perform data mining tasks or to detect anomalous behavior in computer systems. The side effect of using summaries is that queries over historical data may not return exact answers, but only approximate ones.

This paper introduces a new approach which is a trade-off between the accuracy of query results and the time consumed in building summaries.

1 Introduction

A data stream is an ordered, continuous sequence of timestamped data elements [8]. The information naturally occurs in the form of a sequence of data values; examples include sensor data, Internet traffic, financial tickers, transaction logs, etc. The potentially infinite nature of data streams and their high arrival rate imply an inability to store a data stream in its entirety and restrict queries and algorithms to process the data of a stream on the fly in a one pass fashion (i.e: without a prior data storage). In order to be processed in real time, queries must be specified before the beginning of streams. These queries run continuously over a period of time and incrementally return new results as new data arrive. The inability to store a complete stream suggests the use of approximate summary structures, referred to in the literature as synopses [9] or digests [3]. By definition, a summary is an incomplete representation of the historical data of a stream. Summarizing online data streams has been largely studied with several techniques like sketching, sampling, building histograms, and wavelets [2] [7] [11].

In this paper, we investigate using both sampling and clustering processes to make historical summaries on data streams. The sampling method is known to be fast and

M.J. Zaki et al. (Eds.): PAKDD 2010, Part II, LNAI 6119, pp. 343–353, 2010.

efficient for answering important classes of queries, but this technique does not give good results for historical queries [4]. The clustering process enables to maintain a good performance for queries applied on distant past. By taking advantages of the sampling and the clustering processes, we show in this paper that we can make a summary of the whole data stream. Through our performance measurements, we show that the data kept using our algorithm, allow us to have good results for queries that cover a very distant past.

In this paper, we are interested in querying the whole data stream specially, the historical part of the stream. We study the multi-dimensional and numerical data stream produced by a source 'F'. Each data stream can be viewed as a sequence of $< t_i, v_1, v_2, ..., v_n >$ where: $v_1, v_2, ...v_n$ represent the element values for each attribute and t_i is the time when the reading element was produced by the source 'F'. We assume that the elements come under increasing timestamps. The study of delay in the arrival of elements can be managed by our approach but it is not the purpose of this paper.

We are interested in aggregating the stream values within a time interval [6]. This means answering range queries. A range query is a pair Q $< Agg, [t_{start}, t_{end}] >$ where Agg represents an aggregate operation (sum, count, etc.) and $[t_{start}, t_{end}]$ are the bounds of the time period over which the aggregate is calculated. The aggregate operation is computed over the data stream's summary.

We present the Reservoir Hybrid (RH) approach in order to build historical summaries of data streams. It is a two stage summary: initially the summary is in the form of samples then, the elements of these samples evolve to make part of a cluster summary. This proposal offers a good compromise in terms of accuracy and run-time.

2 Related Work

In this section, we briefly describe the CluStream and StreamSamp approaches which are used to create the RH approach for summarizing data streams.

CluStream [1] is based on clustering of quantitative data but it can provide a structure particularly suited for the data stream summary. The objective is to build a summary as evolutionary micro-clusters including snapshots which are stored regularly. Aggarwal was inspired by the BIRCH algorithm [12] using the CFV structure (Cluster Feature Vector) to represent clusters and expands on this concept by adding temporal features. The CFV structure maintains statistical data that summarize all the elements of micro-clusters.

CluStream proceeds in two steps: the first one is the building of a data summary after the initialization of k micro-clusters (using the k-means algorithm). This step consists on the micro-clusters creation and maintenance. When a new stream element enters the Clustream process, its distance to the centroid of each micro-cluster is calculated and then assigned to the nearest micro-cluster. The CFV of this cluster is updated without storing the belonging of this element to the cluster. The second step is characterized by post-analysis which can be applied to the stored snapshots. At each clock time, the algorithm takes a snapshot and saves it according to a pyramidal time frame[1]. This

[1] Snapshots are stored at different levels of granularity. This structure favors recent time frames to older ones.

consists of storing on disk the CFV of all micro-clusters. For post-analysis purposes CFVs allow the subtraction of two snapshots. Approximation of results for statistical queries can be computed over the pre-specified time horizon.

CluStream keeps representative snapshots even for old stream elements. This allows the monitoring of the data stream over time. However, one weakness of the algorithm is that the process of distance calculations is expensive. A second limitation consists of the high number of parameters which depend on the nature of the stream and the arrival speed of elements.

StreamSamp [5] is based on sampling a data stream. It was developed to overcome the limitations of CluStream. It combines a memory-based reduction method which is random sampling and a time-based reduction system which is tilted windows [2]. The algorithm proceeds in two steps: the first one is the building of data summary by sampling and re-sampling stream elements. Upon arrival, the data are sampled in a purely random way with a fixed sampling rate, α, and placed in samples of size T. When T is reached StreamSamp, stores on the disk the sample elements and its starting and ending timestamps of constitution. The weight 1 is attributed to this sample. As it is impossible to keep all samples, when L samples of size T are filled, StreamSamp merges the two oldest samples following the tilted windows system. The weight of the resultant sample (created by random re-sampling) is multiplied by 2. It is a recursive operation. The second step allows the exploitation and analysis of the created summary. It consists of the exploration of the summary retained for a given period. The algorithm starts by constituting the final sample by concatenating elements having different weights to restore the flow of that period. Each element in this period is associated to its weight.

StreamSamp has the advantage of being fast on designing the data stream summary. However, its performance degrades over time because old elements increase in weight for a given sample size. Therefore, if a sample contains recent elements (much lower weight) and some old elements, the latter will increase the errors in the results of query answers.

3 Reservoir Hybrid Approach

The RH approach is presented as a trade-off between processing speed and accuracy in summarized data streams. The basic idea of this algorithm is that the elements from the data stream are first sent into StreamSamp to be processed. When the samples are no longer representative in terms of the two criteria detailed below, the sample elements are sent to CluStream. Depending on the chronological order, these elements are sent to the *reservoir* before sending to CluStream. Since it involves CluStream, only numerical data can be considered.

3.1 Representativeness of Samples

The representativeness of a sample is associated to two criteria: (i) the variance criterion and, (ii) the position criterion of the centroids. These criteria are based on inherent

[2] Summaries with a constant size are maintained covering time periods of varying sizes, shorter for the present and longer for the distant past.

features of the two processes: the random sampling for StreamSamp and the updating evolutionary micro-clusters for CluStream. In order to preserve a summary with good quality, we have to maintain a good quality for these two processes. The first criterion is checked for each attribute while the second one is checked considering all attributes together. The main idea is to transit from the StreamSamp process to CluStream based on a simultaneous check of these two criteria.

Variance Criterion. One of the steps in the StreamSamp process is random re-sampling. In this step, two samples E_1 and E_2, both of size N and covering respectively the time $[t_1 - t_2]$ and $[t_3 - t_4]$, are merged into a new sample E_3. The E_3 sample is created by randomly drawing N elements from E_1 and E_2 and covering the period $[t_1 - t_4]$. This step leads to a degradation of the summary. The variance criterion monitors random re-sampling and measures the 'quality' of the resulting sample.

Definition 1. A sample has a good quality if it can generate, from stored data, an approximate response to aggregate queries such as mean, variance, etc.

We aim to control the quality of the sample resulting from the merger of independent samples E_1 and E_2. We note that these samples have the same weight. The sample's quality is checked by controlling the accuracy of aggregate estimators. In this paper, we check the accuracy of the mean estimator noted $\overline{x}$. However, we could make this criterion more severe by controlling the quality of all used aggregates (mean, median, sum, etc). The goal of this criterion is to set a statistical bound on the mean estimator. Since a sample and random sampling are used, we know that with a confidence of 95%, we have the inequality:

$$\left|\overline{x} - \widehat{\overline{x}}(E_1 \cup E_2)\right| \leq 1.96\sqrt{Var(\widehat{\overline{x}}(E_1 \cup E_2))}. \tag{1}$$

Were E_1 and E_2 are the two samples that have to be merged and $\widehat{\overline{x}}$ is the estimator of the mean.

The variance $Var(\widehat{\overline{x}}(E_1 \cup E_2))$ is estimated according to the following formula :

$$Var(\widehat{\overline{x}}(E_1 \cup E_2)) = (1 - \frac{2n}{N})(\frac{1}{2n})[\frac{1}{2n-1}\sum_{k \in E_1 \cup E_2}(x_k - \overline{x})^2]. \tag{2}$$

Where n is the sample size and N is the size of the involved population

To ensure that merger quality is satisfied, we define a threshold B (user defined parameter) that the error estimator must not exceed. The criterion is expressed using the inequality :

$$\frac{1.96\sqrt{Var(\widehat{\overline{x}}(E_1 \cup E_2))}}{\widehat{\overline{x}}(E_1 \cup E_2)} \leq B \tag{3}$$

However, even if the criterion is met, we do not decide to merge, unless the second criterion, about the position of the centroids is checked.

Position Criterion of the Centroids. While the first criterion concerns the StreamSamp process, the position criterion of the centroids is related to the CluStream process. The random re-sampling process leads to a deterioration on the quality of the built summary.

This may cause a considerable change on the position of the centroids which will be calculated on the remaining samples. Consequently, the accuracy on the position of the centroids deteriorates. Therefore, like the first criterion, a minimum precision on the centroids position must be maintained by establishing a threshold over the distance between the centroids.

Unlike the classical approach of CluStream in which the algorithm processes the whole stream, in our CluStream version, the algorithm will only processes a sampled stream.

In order to maintain this accuracy, we check the centroid's precision at each re-sampling step. We calculate the distance between the centroid (G) (calculated from the samples to be merged (E_1 and E_2)) and the centroid ($\overline{G}$) (calculated from the estimated sample (E_3)). This distance must be below a threshold D. Otherwise, the required precision is no longer respected.

$$D = \epsilon \times \sum_{E_1 \cup E_2} (d^2(\overline{x}, x_i)) \tag{4}$$

Where ϵ is a user defined parameter fixed following the evolution of the centroids, and $\sum_{E_1 \cup E_2} (d^2(\overline{x}, x_i))$ is the intra-cluster inertia of the sample made up from $(E_1 \cup E_2)$

If one of these two criteria is no longer respected, the two corresponding samples will be handled by CluStream.

3.2 Insertion of the Samples in CluStream

The insertion of elements in CluStream depends on two parameters: (1) the weight of elements and, (2) the chronological insertion.

(1) To maintain the representativeness of the elements, the weight must be taken into consideration. Two strategies can be applied for the insertion of elements into CluStream: (i) each element i is inserted w_i times (w_i is the weight of the element i) or (ii) each element is multiplied by its weight and inserted once in the nearest micro-cluster. CluStream uses the Euclidean distance. It is not based on the correlation between attributes (i.e. Mahanalobis distance [10]). The two strategies offer the same results. Consequently, each element is inserted once because the first strategy needs $O(p * \tau * w_i)$ times to insert an element i into the nearest micro-cluster (with p the number of micro-clusters and τ the time needed to calculate the distance between an element and a micro-cluster).

(2) In CluStream, Aggarwal added a temporal extension to the classical form of CFV. In order to reflect the evolution of the stream data over a time period, it is necessary to insert elements in chronological order. Thus, during their move from StreamSamp to CluStream, data elements have to be processed according to the order of their respective timestamps: samples with higher weights must be moved away before the lower weighted samples.

Following the chronological order some samples have to be moved to CluStream only because they are older than another samples, although they still respect the merge criteria. The early transmission of samples from StreamSamp to CluStream leads to the following important drawbacks:

- Unnecessary waste of accuracy, especially for classification tasks.
- Unnecessary waste of time as the clustering process of CluStream is slower than sampling.
- StreamSamp empties quickly and once emptied we lose the performance of this algorithm (fast processing data stream, fast building summaries, good accuracy of the recent period, etc.).

To overcome these drawbacks, a buffer (referred to in the sequel as the *reservoir*) is introduced between StreamSamp and CluStream. As StreamSamp processes data faster than CluStream, the basic idea behind our proposal is to keep data in the StreamSamp process as long as possible, *i.e:* as long as the representativeness criteria are respected. The reservoir structure is filled with samples that: (i) do not satisfy the representativeness criteria and hence can not remain in the StreamSamp process; and (ii) can not be moved yet to the CluStream process because there are older samples which are still in the StreamSamp process. These samples are moved from the reservoir to CluStream when the storage space allocated is reached. The data transfer between StreamSamp, the reservoir and CluStream is based on two rules.

Rule for transmission to the reservoir. Let E_1, E_2 be the two oldest samples of a weight i in the StreamSamp process. Assume that L (the maximum number of samples of weight i) is reached and that E_1 and E_2 cannot be merged. In such case, we check an eventual merger between E_2 and E_3 (the third oldest sample of weight i). If this merger is possible, only E_1 is sent to the reservoir, and the samples E_2 and E_3 are merged. Otherwise, samples E_1 and E_2 are sent to the reservoir. As we cannot indefinitely send samples to the reservoir, we need to vacuum it dynamically by sending elements to CluStream.

Rule for transmission to the CluStream process. Once the reservoir is filled, the δ oldest samples are sent to the CluStream process (δ is a user defined parameter). These δ samples are jointly extracted from the StreamSamp process and from the reservoir. They are sent to CluStream in chronological order from the oldest to the newest. The storage space allocated for the reservoir is not predefined. Rather, as illustrated by the following formula, a global space is shared between the StreamSamp summary and the dynamic reservoir.

$$Size(Res) = Size(HSpace) - Size(SSamples) \tag{5}$$

Res: Reservoir, *HSpace*: total space allowed for the hybrid approach, *SSamples*: Samples in StreamSamp. Such a sharing mechanism allows a flexible management of the storage space. This is an important feature of our approach as the storage space required by StreamSamp from one hand and the reservoir from the other hand, highly data-dependent on the quality of the built samples. Indeed, if the merger's criteria are often satisfied, StreamSamp needs more space than the reservoir as few samples are sent to the reservoir. In the opposite case, the reservoir needs more storage space.

4 Empirical Results

We aim at assessing the performance of our algorithm and comparing it with CluStream and StreamSamp used on their own. To make the comparison fair, all algorithms use the same amount of memory to store their summaries. The algorithm parameters are presented in table 1. Real data sets KDD98[3] and CoverType[4] are used to evaluate the performance of the algorithms. In these evaluations, we are interested in the robustness and efficiency of algorithms for estimating queries which are evaluated over a time period that grows old over time. Thereby, we study the aging period [0-10000] at different timestamps ($t_{10000}, t_{20000}, etc.$). We repeat the StreamSamp and the RH approach 100 times because they include the sampling step. The result corresponds to the mean of these drawings.

Table 1. Parameters of Algorithms

StreamSamp	CluStream	RH approach
$\alpha = 1$	Nb of clusters = 50	$B = 0.25$
T = 200 elements by sample	Nb of snapshots by order = 32	$D = 5.10^{-4}$
L = 8 samples by order		$\delta = 2$

For reasons of limited space, we just present in this section the results for median as querying task and classification as data mining task. Furthermore, other kinds of analysis tasks have been applied (e.g. Mean, clustering) and present good results for the Reservoir Hybrid approach. Note that we also compared these algorithms with the classical Hybrid Approach (without using a reservoir), in order to study the impact of reservoir in the construction of the summary.

4.1 Median Evaluation

We study the performances of the different approaches on median estimation. The estimated error is calculated according to its ranking values:$error = \frac{|EstimatedRank - RealRank|}{WindowSize}$
The *Real Rank* is calculated over the original dataset (5000 in our case) while, the *Estimated Rank* is calculated over the resulted summary and the *Window Size* represents the number of elements studied for the median calculation(10000 in our case). The value of the estimated rank depends on the algorithm used to design the summary:

1. With StreamSamp, the estimated rank is easily calculated because the sampling process preserves the structure of elements. Firstly, all elements included on [0-10000] are extracted and sorted according to the attribute value. We choose the

[3] The dataset contains 95412 records and 481 attributes of information about people who have made charitable donations. After examining the stationarity of the stream, we use only 4 numerical attributes (from 54).

[4] The data contains 581012 elements and is defined by 54 variables of different types. Each element belongs to a class from 7 target classes. The goal is to predict the forest cover type from these variables.

element which divides the distribution into two equal parts. The estimated rank corresponds to the rank of this element in the original data set.

2. With the CluStream algorithm, stream elements are absorbed inside the micro-clusters. For that, on period [0-10000], we use the centroids of micro-clusters as values of elements and the weight corresponds to the number of elements in the micro-cluster. We extract the rank of the median value from the original data set. While the value may not be found (micro-cluster centroid), we search the rank of the nearest lowest value and the rank of the nearest highest value from the original data set. The estimated rank is the mean of these borders.
3. Using the RH approach or the classical Hybrid approach, it is possible to have the StreamSamp and the CluStream processes running in parallel. In this case, we extract from StreamSamp's summary and the reservoir, all elements included on [0-10000]. For CluStream, we search the closest snapshots kept between 0 and 10000 to extract the statistics. We merge the elements from StreamSamp's summary with the centroids of micro-clusters. Then, we calculate the estimated rank on this new set of data.

As shown in figure 1(a), the relative median error calculated on StreamSamp increases with the aging period [0-10000]. This error is calculated once on CluStream given that it keeps two snapshots. The RH approach adopts a similar behavior to StreamSamp for the recent periods. Then, when the quality of summary deteriorates it converges to an accuracy close to the CluStream behavior. This approach provides better performance than either summarizing approaches can provide separately and provides greater accuracy in estimating the median than the classical Hybrid approach.

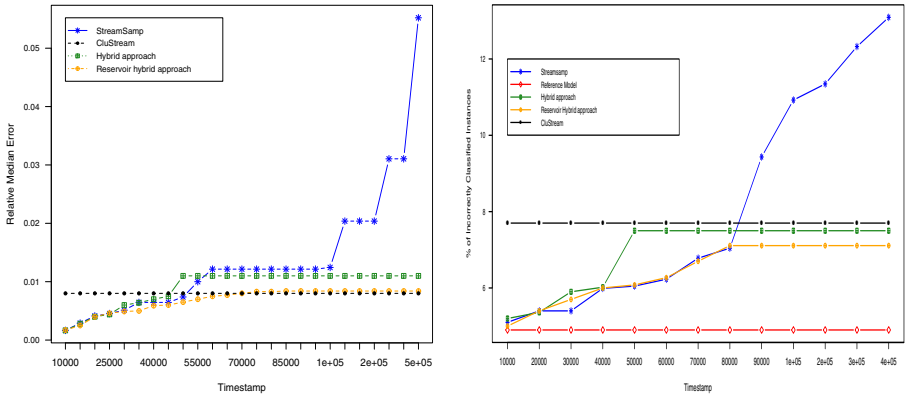

Fig. 1. a) Median Evaluation, b) Classification Evaluation

4.2 Classification Evaluation

We evaluate the performances of the generated models using the different summarizing algorithms. The models are constructed over the fixed period [0-10000] using the CoverType dataset. This dataset contains 7 labels, however, predicting 7 labels is

difficult. To achieve this, we transform the data set to 2 labels: most frequent label (a majority), and all others. We compare the performance of the generated models with the reference which is constructed on the original dataset ([0-10000] in this task). The better algorithm had to develop a prediction model close to the reference. The classification evaluation is done on three steps:

1. Extraction of the summary on the period [0-10000]:
   - From StreamSamp built summary, we extract all the elements between timestamp 0 and 10000. The summary designed by StreamSamp has the advantage of retaining the same structure as the original data set.
   - The summary generated by CluStream only contains statistical data information, therefore, values and labels are absorbed in the micro-cluster. A preprocessing stage of data becomes necessary: (i) generating element values, (ii) adding label attribute.
     (i) For the first process, we use the information kept in micro-clusters to generate n_i elements (n_i is the number of elements in the micro-cluster i), following a Gaussian distribution. This operation is repeated 100 times because of randomly generation process. The result corresponds to the mean of these different drawings.
     (ii) For the second process, we have to associate each generated element to one label. To distinguish element belonging their labels, we use a binary coding in order to transform the 'n' labels in 'n' binary attributes to the dataset. Thereby, for each element, the value of the variable to predict is replaced by a binary value: '1' if value equals the variable and '0' otherwise. Due to this technique, we know for each micro-cluster the labels of the absorbed elements. The generated elements are associated according to these labels.
   - To extract the required part of summary built by the RH approach and the classical Hybrid approach, we pick up all samples from StreamSamp. Then, we concatenate this set of data by the elements generated from the CluStream micro-clusters as described above.
2. Construction of the model: The C4.5 algorithm is used on the extracted summaries in order to construct the models. However, other algorithms like CART or SVM can be applied.
3. Evaluation of the model: Models are evaluated using the training/test method.

As shown in figure 1(b), we compared the derived models constructed by algorithms to the reference model (without summarizing operations). StreamSamp built the closest model for recent periods because it used the real data unlike CluStream which uses data generated from micro-clusters. The classical Hybrid approach and the RH approach present better results in more recent periods since they used data from StreamSamp. For very distant past periods, the RH approach performances stabilize over time and presents the best results while, the StreamSamp performances continue to degrade.

4.3 Runtime Evaluation

In a data stream context, the runtime execution is a very important feature of processing stream data. We take account the global elapsed time for the data stream processing. We

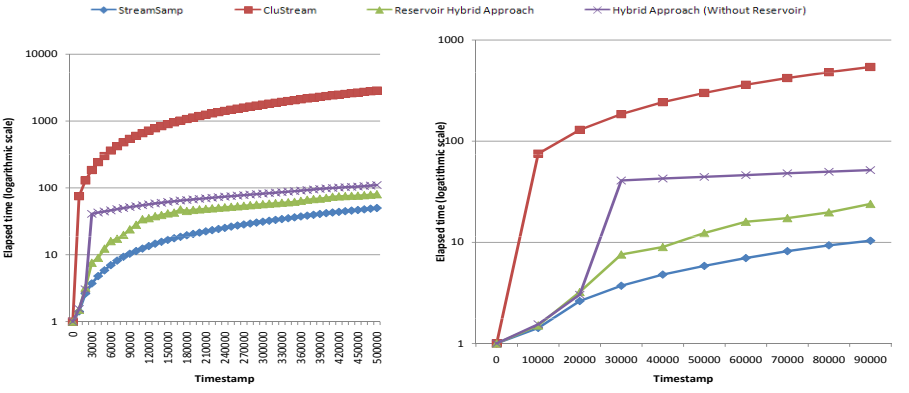

Fig. 2. a) Runtime Evaluation (logarithmic scale). b) Zoom on period [0-90000].

are not interested in the aging period [0-10000] but rather by the cumulative time for processing the whole data stream. As shown in figure 2(a) (logarithmic scale), StreamSamp provides the best performance and CluStream the worst one. The runtime performance of the RH approach remained between those of StreamSamp and CluStream. They are close to the StreamSamp's performance but still much faster than CluStream (more than 10 times faster than CluStream). StreamSamp algorithm uses only merging and sampling tasks. CluStream is the slowest because of updating operations of the CFV structures and the distance calculation between centroids.

The use of the reservoir provides a benefit in speed processing. The performance of the RH approach are better than the classical Hybrid approach and much better than CluStream. Furthermore, using the reservoir strategy, we avoid the heavy initialization step (Runtime evaluation Zoom in figure 2(b)).

4.4 Conclusion

Summarizing data streams is a difficult problem as we need to take into account two antagonistic problems: (i) the representativeness off kept data and hence, the accuracy of queries results; and (ii) the speed processing which is crucial in a data stream context. In this paper, we have developed an efficient method called *Reservoir Hybrid Approach* (RH approach) for summarizing data stream. We present the results for median and classification task, however, other kinds of queries (e.g. mean), and data mining tasks (e.g. clustering) was evaluated. All the evaluation results show that the RH approach solves the two antagonistic problems and provides best results. It provides a better speed-accuracy trade-off than existing approaches. The use of a reservoir makes summary building speed close to StreamSamp performances with an accuracy close to CluStream for distant past periods.

Future work includes the design of a query language allowing the exact querying of current data as well as the approximate querying of historical data.

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB, pp. 81–92 (2003)
2. Cormode, G., Garofalakis, M.N.: Sketching probabilistic data streams. In: SIGMOD Conference, pp. 281–292 (2007)
3. Cormode, G., Korn, F., Tirthapura, S.: Exponentially decayed aggregates on data streams. In: International Conference on Data Engineering, pp. 1379–1381 (2008)
4. Csernel, B.: Résumé généraliste de flux de données. PhD thesis, Ecole Nationale Supérieur des Télécommunications (February 2008)
5. Csernel, B., Clérot, F., Hébrail, G.: Streamsamp: Datastream clustering over tilted windows through sampling. In: ECML PKDD 2006 Workshop on Kbowledge Discovery from Data Streams (2006)
6. Cuzzocrea, A., Furfaro, F., Mazzeo, G.M., Sacca, D.: A grid framework for approximate aggregate query answering on summarized sensor network readings. In: Proc. of the 1st International Workshop on Grid Computing and its Application to Data Analysis (2004)
7. Gemulla, R., Lehner, W.: Sampling time-based sliding windows in bounded space. In: SIGMOD Conference, pp. 379–392 (2008)
8. Golab, L., Tamer Özsu, M.: Issues in data stream management. SIGMOD Record 32, 5–14 (2003)
9. Guha, S., Shim, K.: Offline and data stream algorithms for efficient computation of synopsis structures. In: VLDB '05, p. 1364. VLDB Endowment (2005)
10. Mahalanobis, P.C.: On the generalised distance in statistics. In: Proceedings National Institute of Science, India, April 1936, vol. 2, pp. 49–55 (1936)
11. Rueda, L.: An efficient algorithm for optimal multilevel thresholding of irregularly sampled histograms. In: SSPR/SPR, pp. 602–611 (2008)
12. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. SIGMOD Rec. 25(2), 103–114 (1996)