Clustering Events on Streams using Complex Context Information

YongChul Kwon, Wing Yee Lee, Magdalena Balazinska
University of Washington
Seattle, WA
{yongchul,wingylee,magda}@cs.washington.edu

Guiping Xu
Huazhong University of Sci. & Tech.
Wuhan, China
gpxu@mail.hust.edu.cn

Abstract

Monitoring applications play an increasingly important role in many domains. They detect events in monitored systems and take actions such as invoke a program or notify an administrator. Often administrators must then manually investigate events to figure out the source of a problem. Stream processing engines (SPEs) are general purpose data management systems for monitoring applications. They provide low-latency stream processing but have limited or no support for manual event investigation. In this paper, we propose a new technique for an SPE to support event investigation by automatically classifying events on streams. Unlike previous stream clustering algorithms, our approach takes into account complex user-defined contexts for events. Our approach comprises three key components: an event context data model, a distance measure for event contexts, and an online clustering algorithm for event contexts. We evaluate our approach using synthetic data and show that complex context information can improve online event classification.

1. Introduction

Monitoring applications play an increasingly important role in many domains including network management (*e.g.*, network intrusion detection), computer system management (*e.g.*, clickstream analysis, system health monitoring), enterprise decision-making (*e.g.*, near real-time sales tracking), and sensor-based environment monitoring (*e.g.*, air quality monitoring, car traffic monitoring). As a concrete example, consider a pervasive computing system where a building is instrumented with Radio Frequency IDentification (RFID) readers. In this building, objects and people carry RFID tags allowing the infrastructure to track their movements [23]. Applications monitor streams of location information produced by the RFID infrastructure to offer a variety of services from personalized reminders ("You forgot your wallet in room 340") to sophisticated activity tracking ("Your team members ran into each other and are having an impromptu meeting") [27].

Today, a relatively easy way to implement the above monitoring applications is in the form of continuous queries running in a stream processing engine (SPE) [1, 2, 8, 19, 25]. For example, an RFID reminder system that produces an alert whenever a person is sighted away from their laptop, wallet, or other item can take the form of a two-operator continuous query. First, a join operator correlates the location of people with that of their belongings, then a filter operator produces alerts when owners and objects appear far apart from each other. With this approach, the SPE continuously processes input streams from the monitored environment as per the query specification and the application only needs to implement the logic that handles the output of the query. An SPE can thus greatly reduce the effort involved in implementing a monitoring application.

A stream processing query thus produces outputs when interesting events, such as meetings between groups of friends, occur. To further support monitoring applications, an online stream clustering process (*e.g.*, [12]) can automatically group these events into distinct categories. For example, it could automatically group lectures, department meetings, and social events into three distinct clusters of events. Similarly, the clustering process could categorize network anomalies into link failures, denial-of-service attacks, or false alarms. To perform such classifications, however, online stream clustering algorithms today rely only on event attributes: events in streams take the form of relational tuples whose attributes define a multidimensional space used during clustering. A meeting event, for example, could be characterized by the time, location, and number of attendees. Such event attributes, however, offer only limited information to help in the clustering process, thus circumscribing the quality of the resulting clusters.

In this paper, we present an approach that enhances the online clustering and thus event classification capabilities of an SPE. In our approach, users specify complex context information for events. For example, in the RFID tracking application, the context information for a meeting event

could include not only the location and time of the meeting but also the exact list of people attending the meeting and the list of objects they brought with them. The system uses this context information during online event clustering and to classify each newly detected event by identifying its top-k most similar clusters. Clearly, context information has the potential to improve the accuracy of the clustering process: *e.g.*, knowing the list of people attending a meeting could help the system distinguish impromptu group meetings of similar-sized but different teams. Using such information, however, is challenging because event contexts cannot simply be captured by extending the list of attributes of an event.

More specifically, clustering events in a streaming manner using their context information raises three important challenges. First, we need a data model to represent event contexts and a query language to specify them. Second, we need a distance measure to compare event context instances. Third, we need an online event context clustering algorithm based on the new distance measure.

In this paper, we present a new approach that addresses all three challenges. We model event contexts in the form of sets of relations (*i.e.*, tables) and use standard stream processing queries to specify them (Section 2).[1] For example, the context of a meeting event could include one relation listing meeting attendees, another listing all the items brought to the meeting, and a third indicating simply the location, time, and meeting duration. Our model is thus simple yet flexible and powerful. To compare event contexts, we propose a new distance computation framework called *Context Distanced Measure* (CDM). CDM is based on the Earth Mover's Distance [24] and compares event contexts by taking both their values and structures into account (Section 3). Finally, we propose a new online clustering algorithm, MC-Stream, for event contexts (Section 4). MC-Stream is based on CluStream [3] but is designed to cluster complex events rather than tuples. We evaluate our approach using synthetic RFID meeting event data and, through these preliminary results, show that our approach has the potential to significantly improve precision and recall when classifying new events on streams compared to using only event attributes.

2. Events and Context

In an SPE, a query takes the form of a loop-free, directed graph of operators each processing data arriving on its input streams and producing data on its output stream. Data items on streams are typically relational-style tuples: *i.e.*, each tuple in a stream has the same set of pre-defined attributes.

[1]We already introduced the concepts of an event and its context in our previous work [6] and simply review their definitions in Section 2.

eid	time	duration	location
1	3:20pm	10min	Atrium
2	3:25pm	47min	Room 303
3	4:05pm	23min	Breakout

Figure 1: Example of meeting events: `eid` is the event identifier; `time` and `duration` are the time and duration of the event; `location` identifies where the event occurred in the building.

ASPECT 1: Participants

eid	person	type	departed from
1	Alice	Student	Room 506
1	Bob	Student	Room 387

ASPECT 2: Objects

eid	object id	type	owner
1	132	Laptop	Alice
1	273	Mug	Alice
1	920	Book	Bob
1	946	Book	Charlie

Figure 2: Sample context for a meeting event with `eid` $= 1$. The context has two aspects. Each aspect is a relation.

In our model, an event is a tuple in a stream that takes the form: (`eid, time,` a_1`, ...,` a_n), where `eid` is an attribute that uniquely identifies the event, `time` is the time when the event occurred, and a_1, ..., a_n are the value attributes of the event. Our definition of an event is thus broad. Almost any tuple in any stream can be considered an event as long as it takes the form specified above. Figure 1 shows examples of "meeting events" from an RFID deployment. Each tuple in the table corresponds to one event. Events are output by continuous queries running in an SPE (we call these queries *event-queries* [6]). Events from Figure 1 could be produced, for example, by a trivial query grouping RFID tag identifiers sighted by different readers and producing an output every time a group of tags remains within the vicinity of a reader for some period of time. More sophisticated event queries are possible for RFID and other applications [22, 27].

The attributes of an event describe its main properties and suffice to uniquely identify the event. In our example, the `time`, `duration`, and `location` suffice to distinguish between different meeting events in the building. Understanding an event, however, can require looking at significantly more information surrounding the event. We call this extra information the *context of the event*. For example, when a meeting is detected, important information about the meeting includes the identity and occupation of people attending the meeting, the items they brought with them (*e.g.*, coffee mugs or laptops), and even the last few places each person visited before the meeting. Such extra information provides more details about the event (*e.g.*, "Right after class, Alice bumped into Bob who just came out of his office to get some coffee" rather than "Two people briefly passed each other in the hallway"). Unlike simple attributes such as event times and locations, such complex context information often can not be represented as a single tuple.

To represent the context of an event, we organize its information into *aspects*. Each aspect is a subset of the event context information and is represented as a standard relation (or table). Figure 2 illustrates a sample context of a meeting event. The context comprises two aspects: the participants and objects present during the meeting. Such aspects deliver more information about the meeting than simply the meeting time, duration, and location attributes.

To specify the context of an event, the user executes additional continuous queries that join the event stream with other streams. We call these queries *context-queries* to distinguish them from event-queries [6]. For each event, the result of each context-query is a relation that describes one aspect of the event context. Each aspect is thus a relation and the set of all aspects forms the context of an event.

Although we define the notion of event and event context using a concrete RFID application as example, these concepts are generally applicable.

3. Context Distance Measure Framework

In order to cluster events into meaningful categories using their context information, we need an effective technique for comparing event contexts.

Comparing the contexts of two events requires comparing two sets of relations. Because no existing measure is designed for comparing such nested sets of multidimensional objects, we cannot apply an existing measure directly. In previous work [6], we proposed to flatten event contexts and view them simply as documents containing bags of words. This approach, however, was unsatisfactory because it ignored the rich information captured by the structure of event contexts. Instead, we thus define a distance evaluation framework, CDM, that systematically computes the distance between contexts taking their structure into account. CDM hierarchically evaluates the distance between event contexts in a bottom-up manner. At each level, CDM computes the distance between two entities by aggregating distances of the entities at the lower level. Figure 3 illustrates the overall architecture of the CDM.

Distance between Attributes: At the lowest level, CDM needs a way to compare individual attribute values in tuples. By default, CDM employs commonly used metrics including the Euclidean distance for numeric attributes (normalized to fall in the range $[0, 1]$) and equality matches for categorical attributes. However, because CDM applies these metrics as black boxes, more sophisticated measures could also be used.

Distance between Tuples: CDM treats a tuple t with $|t|$ attributes as a $|t|$-dimensional vector and uses Euclidean distance to compute distances between tuples. This approach is standard for comparing tuples in a database.

Distance between Aspects: An aspect is a set of tuples.

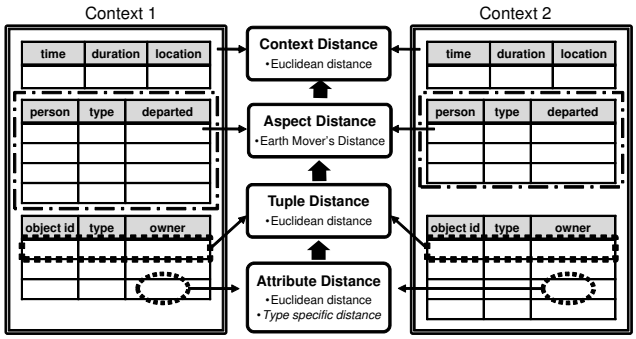


Figure 3: Distance computation between the contexts on the left and right sides using CDM.

To compute the distance between two aspects, CDM could employ one of many set similarity measures including Jaccard's coefficient or Hausdorff distance [30]. CDM uses the Earth Mover's Distance(EMD) [24]. EMD is a metric originally introduced for content-based image retrieval and successfully used in multimedia databases. EMD works as follows: assuming that there is a mound of earth for each tuple in one aspect and a hole in the ground for each tuple in the other aspect, EMD computes the minimum total amount of work to completely level the ground by filling the holes with the mounds of earth. The distance is then the average amount of work per unit of earth moved to perform this leveling. Sizes of mounds of earth and depths of holes represent tuple weights. By default, all tuples have the same weight, equal to $\frac{1}{\text{cardinality of aspect}}$. As an aspect distance measure, EMD has two attractive properties. First, EMD is a metric if all base distances are metric (which they are in the default settings of CDM). This enables us to leverage standard clustering algorithms defined in metric space. Second, EMD is more precise when comparing complex objects than several alternatives: unlike Jaccard that only considers exact matches between tuples, EMD takes the actual distances between tuples into account. With EMD, unlike with Hausdorff, all tuples contribute to the distance computation. In Section 5, we compare the performance of EMD as the aspect distance measure to these alternative techniques in more detail.

Distance between Contexts: Finally, CDM aggregates all distances among corresponding aspects by treating each aspect as a different dimension and computing again, the Euclidean distance:

$$\text{CDM}(A, B) = \sqrt{\sum_i \text{EMD}(A_i, B_i)^2}$$

where A_i and B_i are i^{th} aspects in event contexts A and B. Although different ways of combining distances between aspects into a context distance are possible, treating each aspect as a different dimension is a natural technique.

In summary, CDM computes the distance between two sets of relations in a bottom up fashion by aggregating distances at lower levels. CDM is also highly configurable because it permits domain specific distances at each level.

4. Online Event Clustering

Recently, data stream clustering has received significant attention [12]. Clustering streams of event contexts rather than streams of tuples, however, raises two new challenges. First, we need a medoid- or prototype-based algorithm rather than a centroid- or grid-based one because it is not clear how to define a meaningful centroid for a set of relations. Second, computing the distance between two contexts is an expensive operation (as we show later in Figure 12) and we need a technique that keeps the number of such computations low.

To address these challenges, we propose a new online event clustering algorithm, called *MC-Stream*. MC-Stream is based on CluStream [3], an online clustering algorithm that uses micro clusters. CluStream defines a micro cluster as a set of items within predefined maximum distance of each oter in feature space. The advantage of using micro clusters rather than a "batch" algorithm that accumulates events and clusters them periodically [21] is that micro clusters always reflect the most recent trends and event classification can be done directly as part of cluster maintenance.

We first briefly describe the baseline algorithm, CluStream, then elaborate MC-Stream.

4.1. CluStream

CluStream represents a micro cluster with a Cluster Feature Vector (CFV). For each attribute dimension, the CFV maintains the sum and sum of squares of the attribute values for all tuples in the cluster. The CFV also maintains the sum and sum of squares for the time attribute. In other words, CFV captures simple statistics(mean and variance) of the micro cluster in each feature dimension and time. Finally, the CFV includes the number of tuples in the micro cluster.

CluStream works as follows: it first runs k-means over a small batch of tuples to create an initial set of micro clusters, much larger than the number of clusters in the data. Then, every time a new tuples arrives, if the tuple falls within some pre-defined distance from a micro cluster centroid, it adds the tuple to the micro cluster. If not, the tuple serves to create a new micro cluster. Offline, a user can request that the system produce a smaller number of macro clusters.

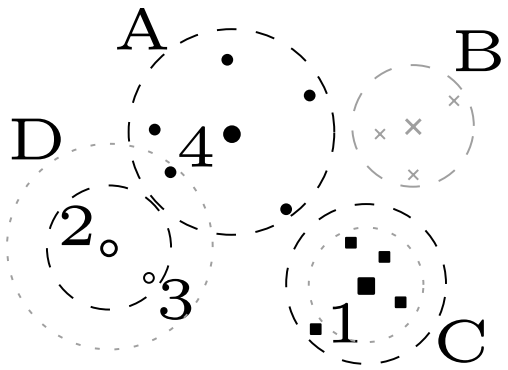


Figure 4: MC-Stream in action. After the initial clustering, three micro clusters (A, B, and C) are created. A new data item, 1, is added to cluster C, and results in slight expanding the cluster radius. When data item 2 arrives, it forms a new micro cluster D and micro cluster B is evicted assuming that it is stale. Subsequent data items 3 and 4 are added to D and A respectively. Note that 3 shrinks the boundary of D by replacing default radius of the singleton micro cluster. Black colors show the final state of the micro-clusters.

4.2. MC-Stream

MC-Stream is based on CluStream. To cluster contexts rather than tuples, MC-Stream represents a micro cluster $\mathcal{MC}$ using a *Medoid-Radius-Time Vector* (MRTV) that we define as:

$$MRTV = (\mathrm{m}, SR^2, SR^1, ST^2, ST^1, n)$$

where m is the medoid of the micro-cluster, $\mathcal{MC}$. SR^2 is the sum of squared CDM distances between m and all other events in the cluster, *i.e.*, $\sum_{i \in \mathcal{MC}} CDM(\mathrm{m}, i)^2$, and SR^1 is simply the sum of the CDM distances, *i.e.*, $\sum_{i \in \mathcal{MC}} CDM(\mathrm{m}, i)$. ST^2 and ST^1 are defined similarly but on the timestamp attributes of events in $\mathcal{MC}$. Like CluStream, ST values are used to check the freshness of a micro cluster. Finally, n is the number of events in $\mathcal{MC}$. Thus, an MRTV is analogous to the CFV of CluStream except it captures statistics in terms of distance to the medoid rather than statistics of all attributes.

MC-Stream works very similarly to CluStream: MC-Stream first runs k-medoid over a small batch of events to an create initial set of micro clusters. Upon detecting a new event, MC-Stream classifies it by linearly scanning all micro-clusters and returning the k closest micro clusters (MC-Stream does not use the offline macro clustering phase). The new event is then inserted into the closest micro-cluster if it is within a predefined distance of the cluster medoid. If it is beyond the limit, a new singleton micro cluster is created (we use the same heuristic when checking the distance against a singleton micro cluster) and

replaces the least recently updated micro cluster. This cluster maintenance mechanism is different from CluStream, which merges the two closest micro clusters to make room for a new one when it can not find a sufficiently stale cluster (*i.e.*, a cluster not updated for a long time). Indeed, merging two MRTVes would require that the vector be either recomputed from the raw events, which would be overly expensive, or be approximated from the two existing vectors, which would not yield tight clusters. We thus leave this feature for future work. The radius threshold is set in a similar way as in CluStream: we use a given parameter t and multiply it by the standard deviation of the radius of the micro cluster. Evicted micro clusters are migrated to disk. We currently do not use these evicted micro clusters. An interesting area of future work lies in exploiting these old micro clusters to produce more accurate classification results when requested by a user.

5. Evaluation

In this section, we first evaluate the benefits of using context information and the performance of CDM for offline event clustering. We then evaluate the precision and recall of using MC-Stream to identify the top-k most likely categories for each newly detected event. Finally, we measure the runtime performance of our approach.

To the best of our knowledge, there is no real data set available that contains complex context information. We thus use synthetic data. We assume the RFID application used as our running example and generate a realistic set of meeting events using our department's academic calendar for Spring quarter 2008. We describe events using event templates where we specify event parameters(time, location, participants, etc.) and the degree of randomness in these parameters. From each template, we generate a set of event instances. We generate two aspects for each event context: `Where` (when and where the meeting was held) and `People` (the set of participants and their occupations). Meetings are either *regular* such as classes or seminars or *ad-hoc* such as coffee breaks or project meetings. We use a machine with a dual 2.66GHz quad core, 16GB RAM running 32bit Linux 2.6.23 kernel.

5.1. CDM Clustering Quality

The goal of our first set of experiments is to answer two questions: (1) does context information improve clustering quality compared with using only event attributes? (2) does EMD outperform other standard measures as the distance function for context aspects? All experiments in this section focus on offline event clustering. We use the standard k-medoid clustering algorithm because our online clustering algorithm uses medoids as well.

We generate two data sets: one with 1000 regular meeting events (65 distinct event types) and the other with 1000 ad-hoc meeting events (107 distinct event types). We cluster each data set using k-medoids and the CDM. We compare the performance when using either Jaccard similarity[2], Hausdorff distance, or the Earth Mover's Distance (EMD) to compare event aspects. We also compare the performance when using only the `Where`, the `People`, or both aspects at the same time and for different ratios between k, the number of clusters created, and the true number of clusters in the data (65 or 107). Figure 5 shows the results. The figure shows both the average cluster purity(expected fraction of major label in a cluster) [26] and the sum of square error(sum of squared distance to cluster center).

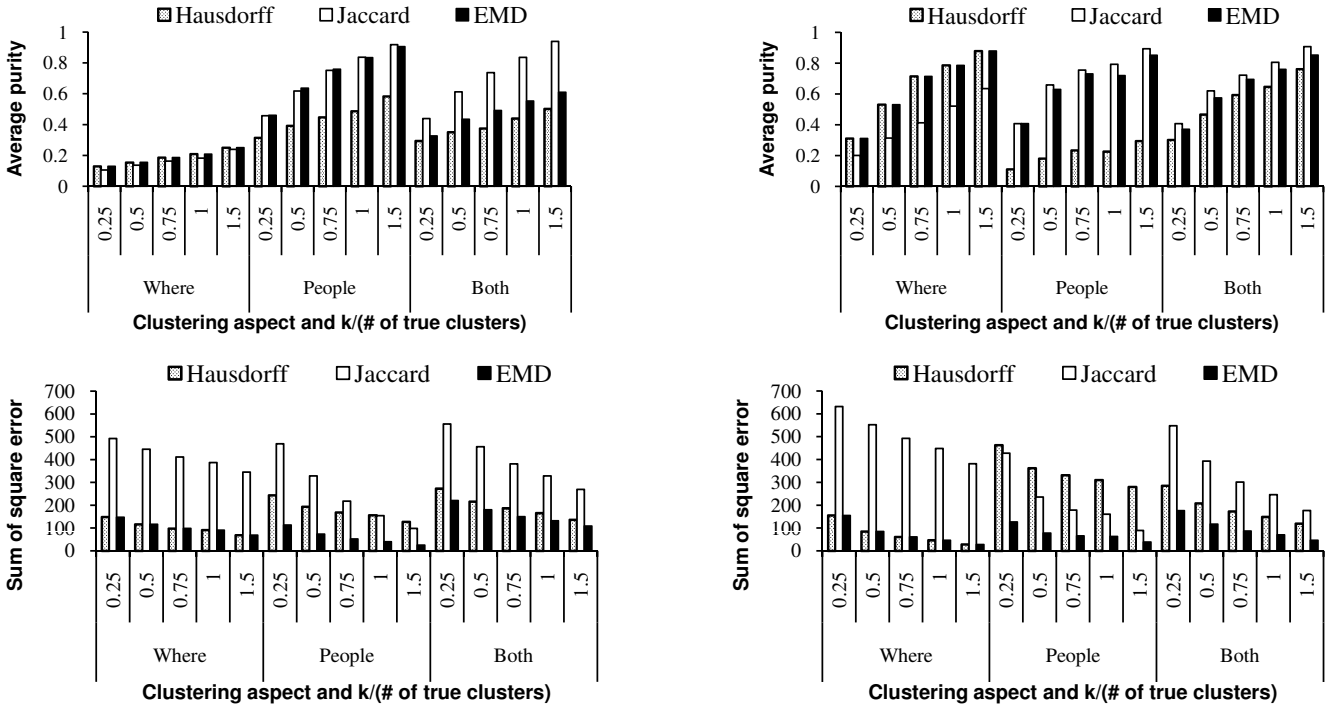
As the figure shows, for ad-hoc meetings and EMD, using the `People` aspect yields significantly higher average cluster purity than using only event attributes (*i.e.*, `Where` aspect). Using both aspects also outperforms event attributes alone, although it loses to using only `People`, the better of the two aspects. Overall, however, using complex context information such as the list of people attending a meeting, significantly improves the clustering quality compared to using only event attributes. The differences are less significant for regular meetings because either aspect alone characterizes regular meeting types quite accurately. For the sum of square errors results, selecting the best rather than combining aspects yields the lowest errors. Overall, however, these preliminary results show that using contexts to automatically classify events is a promising technique.

In all cases, EMD outperforms Hausdorff distance because the latter does not capture detailed differences between aspects. For example, with the `People` aspect, Hausdorff ends up with only one of three values: 0 (when the two aspects are equal), 0.25 (when two aspects share some people), and 1.0 (when two aspects share nothing). Jaccard similarity performs best on the `People` aspect because the data in this aspect is categorical. Jaccard, however, does not handle numerical data well as shown in the results for the `Where` aspect. Additionally, Jaccard similarity yields worse sum of square errors in all cases. EMD is thus the most flexible technique as it can handle both types of data well and yields both high purity and low sum of square error values.

5.2. MC-Stream Performance

In this section, we compare the performance of MC-Stream to that of the original CluStream [3]. MC-stream leverages both aspects in the event contexts, while CluStream can only use the `Where` aspect. Because the goal of our approach is to automatically classify events on streams, we evaluate the techniques by measuring the precision and

[2]We subtract the similarity from 1 to get a distance.



(a) Ad-hoc meetings **(b) Regular meetings**

Figure 5: Clustering quality when using k-medoid with different measures for comparing event aspects and for different ratios between the number of clusters created and the real number of clusters in the data. Left column: ad-hoc meeting events such as coffee breaks. Right column: regular meeting events, such as classes.

recall of their classifications. For both techniques, we use the top-k closest micro-clusters as possible categories for each newly detected event. For this experiment, we define a 3-level (concrete, semi-abstract, and abstract) hierarchy for event categories as shown in Figure 6 (concrete instances such as the "graduate data mining class" are omitted in the figure). The precision at level l in the hierarchy is the fraction of events in the top-k event clusters that belong to the same level-l category as the newly detected event. The recall at level l is the ratio between the number of events in the top-k micro clusters that are in the same level-l category as the newly detected event and the total number of such events in all micro-clusters. We vary l to observe how well the algorithms handle events at different levels of abstraction.

We also vary the number of in-memory micro clusters to evaluate the performance of the algorithm when the total amount of memory is smaller than the total number of clusters in the data. Because we need a larger data set for this experiment, we generate 13000 events (from 1446 distinct event types) at 100 events per timestep. The first 3000 events serve for the initial clustering. The timeout threshold to drop old micro-clusters in CluStream is set to 25

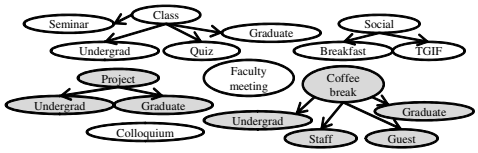


Figure 6: Hierarchy of meeting events. Grey events are ad-hoc events. Concrete event types were omitted in the figure.

timesteps and the maximum boundary threshold for each cluster is set to 2 standard deviations for both techniques. In theory, the two parameters may affect to precision and recall curves due to changes in the dynamics of micro cluster maintanence. However, we observed that the results were not sensitive to either parameter for this data set.

Figure 7 through 10 show the results. In the figures, **C** and **M** denote CluStream and MC-Stream respectively. The number that follows the letter represents the ratio of in-memory micro clusters to the number of distinct event types in the data set (*e.g.*, M 2.0 indicates that twice as many micro-clusters are in memory as there are distinct event types).

Figure 7 shows the average precision with respect to top

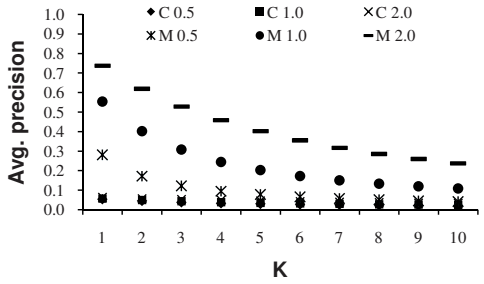


Figure 7: Average precision for concrete events w.r.t. number of top K micro clusters. MC-Stream yields better precision than CluStream.

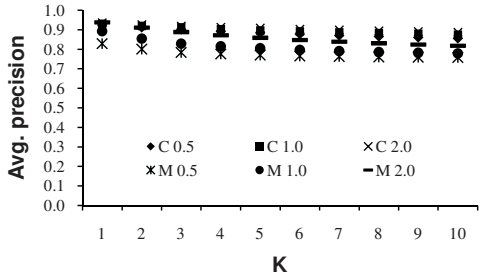


Figure 8: Average precision for semi-abstract events w.r.t. number of top K micro clusters. CluStream yields slightly better precision than MC-Stream.

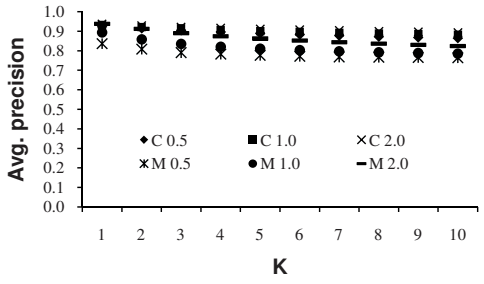


Figure 9: Average precision for abstract events w.r.t. number of top K micro clusters. CluStream yields slightly better precision than MC-Stream.

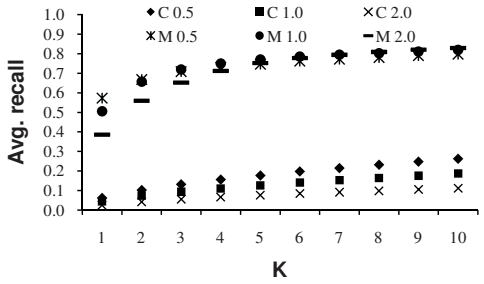


Figure 10: Average recall at concrete level w.r.t. number of top K micro clusters. MC-Stream significantly outperforms CluStream.

k micro clusters using concrete events in the event hierarchy. With enough micro clusters, 70% of events in the top micro cluster produced by MC-Stream are of the same type as the newly detected event, while this number is only 30% for CluStream. At higher levels of abstraction, however, both techniques yield comparable performance as shown in Figure 8 and 9.

Figure 10 shows that MC-Stream outperforms CluStream even more significantly in terms of recall. By exploiting the extra `People` aspect, MC-Stream's medoid algorithm clusters the same type of events more closely than CluStream's centroid version. Recall for higher levels of abstraction are very low because the total number of matching abstract events vastly outnumbers that in the top-k micro clusters.

In summary, our preliminary results show that taking context information into account when clustering events on streams can yield better precision and recall than solely using event attributes.

5.3. Cost of CDM evaluation

In this experiment, we examine whether we can run MC-Stream at streaming speed. For this, we measure the cost of CDM distance computations. Because the total evaluation time of CDM is linear in the number of aspects in the event context, we only evaluate the performance of EMD computation.

We use the EMD implementation by Rubner et. al compiled using GCC 4.1.2 with -O3 -march=pentium4 options. All values in aspects are randomly generated floating point numbers. We use Euclidean distance to compare tuples.

Figures 11 and 12 show the average time in milliseconds from 200 aspect comparisons as we vary the number of attributes and the number of rows in an aspect (standard deviations are within 2% of the averages). The results clearly show that the number of rows affects the evaluation time more than the number of attributes. However, using more complex attribute distance measures could also impose a high overhead. Both figures confirm that CDM evaluations should be avoided or cached whenever possible; even in the fastest scenario, we can run only a couple of thousands evaluations per second. This means that MC-Stream could be significantly slower than original CluStream which uses standard distance metric between two vectors. Thus, we have to organize micro clusters in a structured way to reduce the number of distance evaluations rather than simply scan the whole list of micro clusters. We discuss this optimization next.

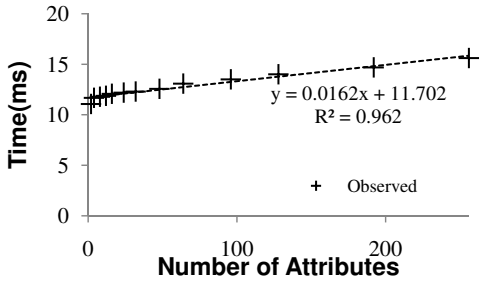


Figure 11: EMD evaluation time w.r.t. number of attributes when the number of rows is 100. The evaluation time increases linearly with the number of attributes in an aspect.

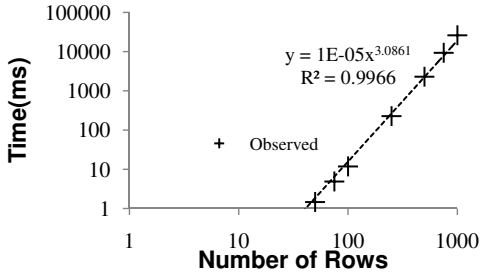


Figure 12: EMD evaluation time w.r.t. number of rows when the number of attributes is 8. The evaluation time increases approximately with the cube of the number of rows in an aspect.

6. Discussion

As shown in Section 5.3, CDM evaluations can be expensive: *e.g.*, even with only 1ms per distance computation, it takes 3s to compare a context against 3000 micro clusters. Instead of linearly scanning all micro clusters, however, we can index micro clusters in a tree as in BIRCH [31] or ClusterTree [29]. Such an index would save unnecessary CDM evaluations. Figure 13 shows the number of CDM computations (analytically derived) to reach a leaf node in a tree with 10K or 1M micro clusters and for different fanouts. Clearly, smaller fanouts save more computations reducing them from millions to less than 100. However, larger fanouts yield better data partitions since small numbers of clusters at intermediate nodes cause these clusters to grow in size. Thus, with a smaller fanout, when retrieving k closest clusters, we may end up scanning a more significant portion of the tree due to huge overlaps at intermediate nodes. Finding an optimal fanout and structure to minimize the k closest clusters retrieval cost is an interesting area of futre work.

The CDM evaluation cost grows with the cube of the number of rows in an aspect. Thus, reducing the data complexity of an aspect by filtering rows, reducing dimensionality, or applying summary techniques such as sketches can boost performance and is thus interesting future work.

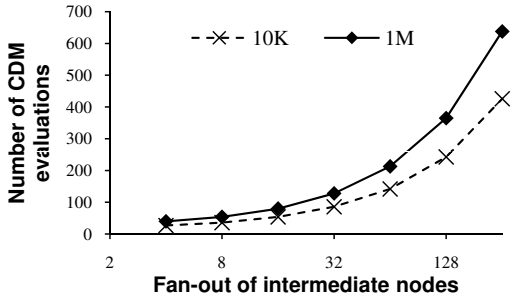


Figure 13: Number of CDM evaluations to reach the leaf node in a tree containing 10K or 1M micro clusters. The x-axis shows the number of medoids in each node in the tree.

Finally, event contexts can be very complex with many aspects. As we showed, in some cases, adding an extra aspect may actually hurt rather than improve performance. An interesting problem is thus how to select the most significant aspects to use in order to get good clustering performance at a low cost.

7. Related Work

There exists significant work on data stream management systems [13]. These systems provide effective near real-time event detection, but offer limited support for the subsequent investigation of events. In previous work [6], we proposed an overall system architecture for combining live and historical stream data and for extracting similar past events to newly detected ones. In this paper, we go further and investigate detailed measures for comparing event contexts and clustering events using context information.

Similarity queries (*e.g.*, [14, 30]) have been extensively investigated in the database literature resulting in a multitude of similarity measures. For example, metrics based on the Cosine Similarity [5] are most appropriate for data objects comprising categorical values and have been successfully used in document searching and ranking of database query results [4]. CDM currently focuses on numerical attributes and uses only exact matches for categorical attributes, but it could be extended with such more sophisticated measures. Minkowsky Distance and Quadratic Distance are suitable to process numerical attributes in vector metric space models. They are thus applicable to image and multimedia similarity searching [30], but they can not directly be used to compare objects like database tuples with both categorical and numerical attributes. Finally, as we discussed above, there exist several measures for set comparisons (*e.g.*, Jaccard's coefficient, Hausdorff distance, and EMD) but EMD has the most suitable combination of properties for our problem.

Clustering algorithms for static data have been extensively researched in many domains [7, 15]. Traditional al-

gorithms are key to bootstrap MC-Stream by populating the initial micro clusters. Since it is hard to define a mean for a set of event contexts, MC-Stream uses a medoid and radius representation for its clusters. Thus, any medoid-based clustering algorithm [16, 20] can be used for this step. If CDM is configured with distance functions that are not metric and is thus no longer a metric, relational clustering algorithm [7, 10] can be used to get the initial clusters.

Multi-relational data clustering [17, 18, 28] is very relevant to clustering event contexts because an event context is a special case of multi-relational data with a star schema. The RBDC [17, 18] algorithm hierarchically clusters multi-relational data based on a similarity measure proposed in RIBL [11]. CrossClus [28] clusters data stored in multiple relational tables based on user hints and multi-relational features. It also represents each cluster with a medoid. Both algorithms can be used to cluster event contexts. However, they are not designed for data streams. When computing similarity, CrossClus considers only one column in a joined relation while RDBC and the CDM framework consider entire columns. Both RIBL and the CDM framework recursively compute similarity (or dissimilarity) measures. However, they aggregate the measures of multiple tuples differently. The relevant feature-search technique proposed in CrossClus is orthogonal to our approach.

Data stream clustering algorithms have also extensively been researched. CluStream [3] performs micro clustering online and macro clustering offline upon user requests. Micro clusters are represented with a feature vector similar to BIRCH. In this paper, we extended CluStream to handle a complex data type such as an event context. Similarly, D-Stream [9] combines grid and density based clustering but it is unclear how it could be adapted to cluster event contexts. STREAM [21] proposed a k-median data stream clustering algorithm. This algorithm, however, does not offer significant advantages over the micro clustering approach because the clustering and classification processes are separated.

8. Conclusion

General-purpose stream processing engines are well-suited for near real-time monitoring tasks, but have limited support for the investigation of new events using historical data. Algorithms for automatically classifying events on streams through clustering exist but use only event attributes. In this paper, we described and evaluated the Context Distance Measure (CDM) framework for computing distances between events on streams using complex, user-defined context information. We also proposed MC-Stream, a new online clustering algorithm that efficiently groups events on streams using CDM to take their contexts into account. When extracting the top-k most similar clusters for a newly detected event, we showed that using context information and MC-Stream improves precision and recall compared to using existing techniques based solely on event attributes. Overall, supporting automatic classification and forensic analysis of events in SPEs is an important task and we view this work as an important step in this direction.

9. Acknowledgments

We thank James Lee for suggesting the Earth Mover's Distance and the anonymous reviewers for helpful comments on drafts of this paper. This work was partially supported by NSF Grants IIS-0713123, IIS-0454425, and a gift from Cisco Systems Inc.

References

[1] Abadi et. al. Aurora: A new model and architecture for data stream management. *VLDB Journal*, 12(2), Sept. 2003.

[2] Abadi et. al. The design of the Borealis stream processing engine. In *Proc. of the Second CIDR Conf.*, Jan. 2005.

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proc. of the 29th VLDB Conf.*, 2003.

[4] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *Proc. of the First CIDR Conf.*, Jan. 2003.

[5] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[6] M. Balazinska, Y. Kwon, N. Kuchta, and D. Lee. Moirae: History-enhanced monitoring. In *Proc. of the Third CIDR Conf.*, Jan. 2007.

[7] P. Berkhin. A survey of clustering data mining techniques. pages 25–71, 2006.

[8] Chandrasekaran et al. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *Proc. of the First CIDR Conf.*, Jan. 2003.

[9] Y. Chen and L. Tu. Density-based clustering for real-time stream data. In *Proc. of the ACM SIGKDD 2007 Conf.*, 2007.

[10] J. V. de Oliveira and W. Pedrycz. *Advances in Fuzzy Clustering and its Applications*. John Wiley & Sons, Inc., New York, NY, USA, 2007.

[11] W. Emde and D. Wettschereck. Relational instance-based learning. In *ICML*, pages 122–130, 1996.

[12] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. *SIGMOD Record*, 34(2):18–26, 2005.

[13] L. Golab and T. M. Özsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, June 2003.

[14] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces (survey article). *ACM Trans. Database Syst.*, 28(4):517–580, 2003.

[15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Survey*, 31(3):264–323, 1999.

[16] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, New York, 1990.

[17] M. Kirsten and S. Wrobel. Relational distance-based clustering. In *ILP '98: Proceedings of the 8th International Workshop on Inductive Logic Programming*, pages 261–270, London, UK, 1998. Springer-Verlag.

[18] M. Kirsten and S. Wrobel. Extending k-means clustering to first-order representations. In *ILP '00: Proceedings of the 10th International Conference on Inductive Logic Programming*, pages 112–129, London, UK, 2000. Springer-Verlag.

[19] Motwani et. al. Query processing, approximation, and resource management in a data stream management system. In *Proc. of the First CIDR Conf.*, Jan. 2003.

[20] R. T. Ng and J. Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002.

[21] L. O'Callaghan, A. Meyerson, R. Motwani, N. Mishra, and S. Guha. Streaming-data algorithms for high-quality clustering. In *Proc. of the 16th ICDE Conf.*, volume 00, 2002.

[22] C. Ré, J. Letchner, M. Balazinksa, and D. Suciu. Event queries on correlated probabilistic streams. In *Proc. of the 2008 SIGMOD Conf.*, pages 715–728, New York, NY, USA, 2008. ACM.

[23] The RFID Ecosystem. `http://data.cs.washington.edu/RFID/`, 2006.

[24] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2), 2000.

[25] Streambase. `http://www.streambase.com/`.

[26] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[27] E. Welbourne, N. Khoussainova, J. Letchner, Y. Li, M. Balazinska, G. Borriello, and D. Suciu. Cascadia: A system for specifying, detecting, and managing RFID events. In *Proc of the Sixth MobiSys Conf.*, 2008.

[28] X. Yin, J. Han, and P. S. Yu. Crossclus: user-guided multi-relational clustering. *Data Mining and Knowledge Discovery*, 15(3):321–348, 2007.

[29] D. Yu and A. Zhang. Clustertree: Integration of cluster representation and nearest-neighbor search for large data sets with high dimensions. *IEEE TKDE*, 15(5):1316–1337, 2003.

[30] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search The Metric Space Approach*. Springer Science+Business Media, Inc., 2006.

[31] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *Proc. of the 1996 SIGMOD Conf.*, 1996.