Mining and Linking Patterns across Live Data Streams and Stream Archives *

Di Yang, Kaiyu Zhao, Hanyuan Lu, Maryam Hasan, Elke Rundensteiner and Matthew Ward
Worcester Polytechnic Institute
diyang|kaiyuzhao|hylu|mhasan|rundenst|matt@cs.wpi.edu

ABSTRACT

We will demonstrate the visual analytics system $Vistream^T$, that supports interactive mining of complex patterns within and across live data streams and stream pattern archives. Our system is equipped with both computational pattern mining and visualization techniques, which allow it to not only efficiently discover and manage patterns but also effectively convey the mining results to human analysts through visual displays. In our demonstration, we will illustrate that with $Vistream^T$, analysts can easily submit, monitor and interact with a broad range of query types for pattern mining. This includes novel strategies for extracting complex patterns from streams in real time, summarizing neighbor-based patterns using multi-resolution compression strategies, selectively pushing patterns into the stream archive, validating the popularity or rarity of stream patterns by stream archive matching, and pattern evolution tracking to link patterns across time.

1. INTRODUCTION

The mining of complex patterns such as clusters, outliers, and top-k nearest neighbors from huge volumes of streaming data has been recognized as critical for diverse application domains, ranging from moving object monitoring to stock transaction analysis. For example, various pattern mining requests, ranging from asking for intensive-transaction areas (clusters) to detecting abnormal transactions (outliers), are submitted against the transaction stream from NYSE by financial analysts every day.

Unlike traditional data mining in static environments where mining queries are submitted to static datasets for one-time mining results, stream analysis tasks tend to be continuous. That is the latter requires a mining system to build a **temporal context** for mining results for analysts. For example, financial analysts analyzing stock transaction streams may want to be continuously informed about the latest patterns, such as intensive transaction areas arising in the streams. Also, to correctly interpret the market trend implied by these transactions, she needs to learn how these patterns change or relate to each other over time. Other example inquiries include whether an interesting pattern just observed has ever appeared before in the past stream, or what are the relationships between the patterns observed now and those observed 5 minutes back, or even which patterns are likely to remain stable across time. Clearly, this capability of mining not only the present but also the past, and of establishing relationships among patterns across time provides a powerful paradigm for effective stream mining.

Unfortunately, state-of-the-art stream mining techniques tend to focus on mining the present [1, 2, 3], with recent efforts on improving efficiency for extracting patterns in real-time from high speed streams. Little progress has been made toward supporting other stream pattern mining services, such as real time pattern summarization, pattern matching, prospective pattern maintenance and pattern evolution tracking. These services significantly extend an analyst's understanding of the streams, helping them to discern trends and phenomenon of true significance.

Over the past 15 years the XMDV team at WPI, composed of database, visualization and HCI experts supported by a series of six NSF grants, has developed an open-source visual tool suite XmdvTool ($http://davis.wpi.edu/xmdv/$) to facilitate interactive data exploration. In our recent research effort, we have began to enhance this tool by providing services for interactive pattern exploration in streaming data, resulting in a streaming version of $XmdvTool$ [6], called $Vistream^T$. The ViStream^T [1] system that we propose to demonstrate here is a significant extension of our previous basic pattern detection technology. It advances our previous work, by providing a rich set of functionalities for mining and querying patterns in the past, the present and the future of streams. It is an integrated platform that includes all of our major research results on complex pattern mining since 2009 [5, 6, 8, 7, 9, 10].

In particular, $ViStream^T$ makes the following contributions:

1) Extract complex patterns from data streams for both single and several interrelated queries using scalable execution strategies.

2) Summarize extracted patterns into descriptive yet highly compact formats by employing multi-resolution compression strategies. This compression achieves compact storage and

*This work is supported under NSF grant CCF-0811510, IIS-0812027, IIS-0119276 and IIS-00414380.

[1] T stands for "Timeline", highlighting ViStream^T's new functionalities for mining the whole timeline of streams.

efficient retrieval while preserving key pattern features, such as the internal pattern structure and external shapes of complex patterns like cluster structures.

3) Identify patterns similar to a pattern of interest from the historical pattern archives via pattern matching.

4) Define and track evolution of pattern changes over time, from the past, to the present and even to the near future.

5) Provide an interactive platform to visually initiate a request and interactively display the knowledge mined above in unified visualizations.

The ViStream^T system will be released as a freeware in our next XMDV open-source release (*http : //davis.wpi.edu/xmdv/downloads/*).

2. ARCHITECTURE OF VISTREAM^T

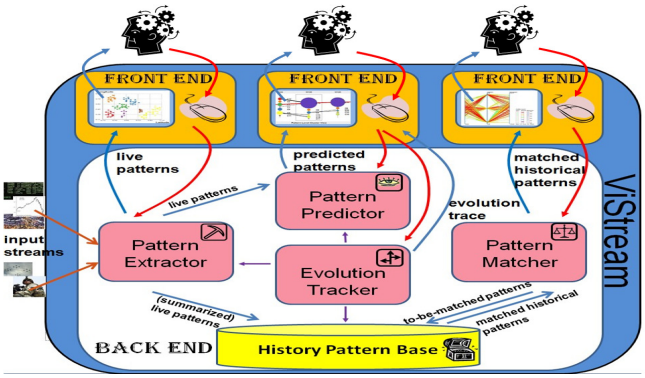

Figure 1: System Architecture of ViStream^T

The $ViStream^T$ system is composed of one computational backend and potentially multiple frontends, each serving one analyst. At the **computational backend**, the ***pattern extractor*** uses our pattern detection algorithms [9, 10, 8] to extract live patterns from the current stream window. The extracted patterns are consumed by one of three different destinations. 1) Direct feedback via visual displays to the analysts. 2) Archive in the ***history pattern base*** as part of the stream history. Analysts can specify whether to summarize the extracted patterns, and if yes, to what degree, before archiving them [7]. 3) The ***pattern predictor*** extracts the pattern that may appear in the near future by analyzing the live pattern just extracted. It is passed to the ***pattern predictor*** to be used as basis for extracting prospective patterns [5]. The ***pattern matcher*** mines the *history pattern archive* to find patterns in the stream history most similar to those specified by the users' pattern matching queries [7]. The ***evolution tracker*** monitors the interrelationship between the historical, live and prospective patterns to determine the evolution of patterns over time [5]. In each **visual frontend**, the mined knowledge is integrated into a single pattern space conveyed using the visualization pipelines [4]. A diversity of effective interaction tools are provided for analysts to explore this knowledge, visually uncovering trends and patterns across time and data sets.

3. PATTERN SUMMARIZATION, ARCHIVING AND STREAM MATCHING

3.1 Pattern Extraction

As the gateway of the backend, the *pattern extractor* handling huge workloads is a scalable fashion by first grouping the pattern extraction queries submitted by multiple analysts by their target pattern types. Then, it employs the shared pattern extraction strategies proposed in [7, 8] to achieve effective execution of possibly huge workloads.

In particular, to share computation among queries, we identify a "containment" interrelationship among patterns identified by these queries with different pattern-specific parameter settings. Then we introduce a representation that captures the patterns identified by queries within a single compact structure. For example, given two density-based clustering queries Q_i and Q_j, if $Q_i.\theta^{range} \geq Q_j.\theta^{range}$ and $Q_i.\theta^{cnt} \leq Q_j.\theta^{cnt}$ [2], then Q_i is more "relaxed" than Q_j. This indicates that the clusters identified by Q_i will be the "expansion" or "merge" of the clusters identified by Q_j or completely "new clusters" (not identified by Q_j before) [10]. Thus, we can represent the cluster structures identified by those two queries using a single hierarchical cluster representation, resulting in significant savings in both CPU and memory resources.

Second, by leveraging overlaps among sliding windows, our meta-query strategy utilizes a single query to answer multiple queries with different window-specific parameters. Both the incremental pattern representation and meta-query techniques can be applied to several complex pattern types, such as clusters, k nearest neighbors, and distance-based outliers [8]. Our system realizes efficient shared execution of multiple queries with arbitrary parameter settings.

Pattern predictor provides analysts the capability to determine which patterns are the most likely to persist as the window slides. The extraction is made based on both the live patterns just extracted in the present stream window and the characteristics of the pattern extraction query. For example, given clustering query Qc_i which periodically extracts clusters from a sliding window (window size $win = 60$ seconds, slide size $slide = 10$ seconds) over a data stream, and the clusters extracted by Qc_i from the current window at time 00:00:00, ViStream^T can pre-compute the clusters that are likely to persist in the following output moments, 00:00:10 up to 00:00:60. This is achieved by analyzing the timestamps of the objects composing the current live patterns (clusters in this case), and by discounting the effect of expiring objects that will happen in the future to those live patterns. Although these patterns may need to be updated when new objects in the stream arrive, many properties are guaranteed to hold due to integrality and containment.

3.2 Multi-Resolution Pattern Summarization

For long-term analysis, the extracted live patterns may need to be archived for later reference. Based on the specific analytical task and available storage, an analyst may decide to summarize the extracted patterns before archiving them. The benefits of pattern summarization are manifold. They include not only savings in storage space but also highlighting of the key features that make patterns quantitively comparable (important for later pattern retrieval). For example, density-based clustering methods are capable to produce arbitrary shapes clusters, however the state-of-

[2] θ^{range} and θ^{cnt} are two input parameters that define density-based clusters

the-art representation for each cluster is simply all its cluster members (tens of thousands or even millions of objects). Obviously, such simplistic representation causes significant strains on pattern storage and retrieval processes. To solve this problem, we analyze the pattern structures of density-based clusters and identify their key features that covers both the internal structure and their external shapes. To capture these features, we employ two summarization principles, namely the graph-based and the grid-based strategies, into one multi-resolution summarization method, called Skeletal Grid Summarization (*SGS*) [7]. Figure 2 shows an example of our proposed Skeletal Grid Summarization (SGS) for a 2D cluster.

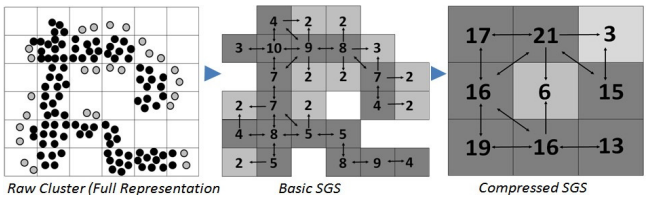

Figure 2: SGS Summarization for a 2D cluster

An important capability is the design of efficient pattern summarization algorithms. A streaming system conducting expensive pattern extraction against high speed input streams can hardly afford additional computational costs for pattern summarization. Thus, to minimize the computation costs for pattern summarization, we have developed an integrated pattern extraction + summarization approach [7]. For example, our proposed summarization method for density-based clusters, C-SGS, incrementally maintains both the full representation and the corresponding SGS summarization of the extracted clusters in an integrated manner. As our experimental studies on several data sets confirm, this results in an almost "free" cluster summarization generation process by piggy-packing the summarization process into the cluster extraction process itself [7].

3.3 Live Pattern Matching Against Stream Archives

In $ViStream^T$, the *pattern matcher* aims to understand the properties of current stream phenomena by matching observed patterns in the current stream against the past stream archive. In particular, an analyst specifies a pattern of interest P_i either from newly extracted live patterns or from the *history pattern base*. She then asks the *pattern matcher* to find patterns that are most similar to P_i in the *history pattern base* or via the live pattern match.

Organization of History Pattern Base. The *history pattern base* organizes patterns based on their key features. For example, for density-based clusters, we build two indices for the archived clusters. One is based on the position of each cluster, and the second is based on all other features of each cluster captured in SGS. The first index, the *locational feature index*, expresses the position of each cluster using its minimum bounding rectangle (MBR). We employ one of the most widely used indices for MBRs, namely the R-tree index, to organize them. The second index, called the *non-locational feature index*, organizes the clusters based on their non-locational features. We use a four-dimensional grid index to organize the clusters' SGS, with the four dimensions: the volume (number of *skeletal grid cells*), the status count (number of *core cells*), the average density and the average connectivity of each cluster.

Pattern Matching Process. The strategy for executing *Pattern Matching Queries* is composed of the coarse-grained **candidate search** and the fine-grained **detailed match**. Our system first searches for the potential match candidates in the pattern archive. In the position-sensitive case, it searches the *locational feature index* for potential candidate patterns and calculates their non-locational distance with the to-be-matched patterns. In the non-position-sensitive case, the Pattern Matcher directly searches the non-locational feature index for the candidates.

Given the to-be-matched pattern and a matching candidate pattern, it compares their features in corresponding sub-regions and aggregates the results to assign a match score. In the position-sensitive case, a single scan on the *skeletal grid cells* in two clusters is sufficient to calculate their match score. In the non-position-sensitive case, our system uses an A* style anytime search algorithm to search for the best alignment. Figure 3 shows an example of the pattern matching view in our $ViStream^T$ with the to-be-matched cluster on the left most side (specified by analyst) and three matched clusters found in the stream history. In general, analysts can select any live pattern detected by ViStream miner of interest, and commit it to the *pattern matcher*. A pattern matching view will be generated. The analysts can gain an insight into the popularity of the to-be-matched pattern, request matched patterns under some similarity thresholds, or simply require the top k similar patterns to be returned. Analysts will be offered an overview by the pattern (upper) level of the displays, while learning more details about the patterns through the tuple (lower) level displays.

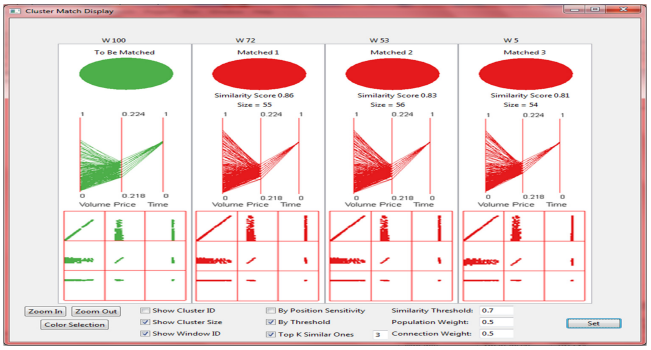

Figure 3: Pattern Matching View with leftmost to-be-matched live pattern and three matches from pattern archive

4. EVOLUTION TRACKING: LINK PAST, PRESENT AND FUTURE

$ViStream^T$ also provides evolution tracking functionalities for linking mining results through the time line of the data stream [5]. To track the evolution of the patterns over time, we analyze the characteristics of the pattern structure for each pattern type and design an evolution model (semantics) that describes the pattern changes for the given pattern type. Taking density-based clusters as example, we

present the first evolution model for density-based clusters in sliding windows [5]. This evolution model not only covers statistical changes of individual clusters, such as the size or the centroid changes, but also classifies structural cluster changes, including splitting and merging of clusters.

Based on the proposed evolution models, we have designed efficient evolution tracking algorithms. Clearly, the two-phase solution of pattern extraction first followed by evolution tracking is not efficient. We observe that this two-step solution suffers from a significant performance overhead, because when conducted independently, the evolution tracking process can be as expensive as the pattern extraction process itself. Thus, we propose a more efficient approach by integrating evolution tracking within the pattern tracking process itself [5]. By using this integrated strategy, we achieve almost "free" evolution tracking along with the pattern extraction process.

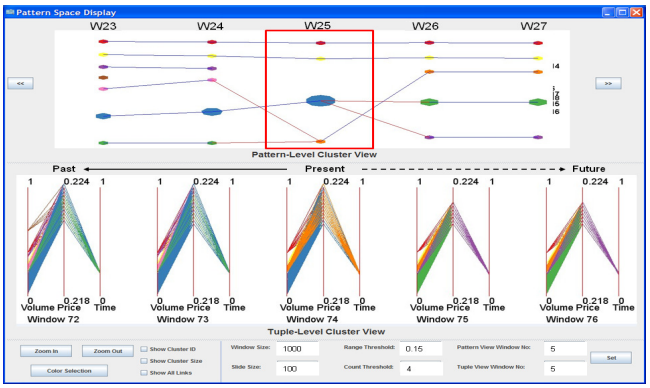

Figure 4: Visualized Pattern Space of ViStream^T

We organize the detected patterns and their evolution information into a multi-dimensional pattern space, with one dimension representing the pattern changes over time and the other representing patterns in different abstraction levels. We have designed a rich set of visualization and interaction techniques to enable analysts to easily navigate through the proposed pattern space for reviewing and monitoring pattern changes over time at different levels of abstraction. Figure 4, shows an example of the evolution from the past (W_{23} and W_{24}) to the present (W_{25}) and the near future (W_{26} and W_{27}) as applied to clusters. Clusters that persist across multiple windows are indicated via the same color. The links between any two clusters indicate that the later cluster contains the cluster members of the previous one.

5. VISTREAM ^{T} DEMONSTRATION

In our demonstration, we will let the audience first hand experience the use of ViStream^T for the exploration of complex patterns in data streams. The demonstration will be based on a live data stream $TwinCityTraf$ reporting the real-time traffic conditions in the Twin Cities area (MN). We will also work with two archived real data streams, namely, the STT data recording stock transactions from NYSE and the GMTI data recording information about moving objects. We will let the audience experience the following functionalities of the ViStream^T system:

1) Submit and monitor multiple pattern extraction queries that mine the present stream [10]. Users can compare the patterns extracted based on different parameter settings through various visualization techniques in ViStream^T, such as different display methods, including parallel coordinates, scatterplots matrix and star glyphs, and different layout strategies, such as juxtaposed and integrated layouts [4].

2) Submit pattern matching queries that mine the stream past by examining the similarities between the to-be-matched patterns and the matched patterns previously found through the visual exploration (see Figure 3). If users are not fully satisfied with the matched patterns found so far, they can adjust query parameters for matching, such as the distance function and similarity threshold.

3) Submit prospective pattern extraction queries that aim to identify which patterns in present window are the most likely to persist. The live patterns most likely to persist across time will be visualized to help users understand the potential trend of the stream characteristics changes. Users can adjust the maximum numbers of future windows based on their specific analytical tasks.

4) Turn on evolution tracking that analyses and highlights the interrelationships among the historical, live and prospective patterns (see Figure 3). Users can observe the evolution processes through our proposed pattern evolution view in ViStream^T, and learn how the patterns change over time.

6. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.

[2] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In *CIKM*, pages 811–820, 2007.

[3] Y. Chen and L. Tu. Density-based clustering for real-time stream data. In *KDD*, pages 133–142, 2007.

[4] Z. Xie, M. O. Ward, and E. A. Rundensteiner. Exploring multivariate data streams using windowing and sampling strategies. *Interacting with temporal data workshop, CHI*, 2009.

[5] D. Yang, Z. Guo, E. A. Rundensteiner, and M. O. Ward. Clues: a unified framework supporting interactive exploration of density-based clusters in streams. In *CIKM*, pages 815–824, 2011.

[6] D. Yang, Z. Guo, Z. Xie, E. A. Rundensteiner, and M. O. Ward. Interactive visual exploration of neighbor-based patterns in data streams. In *SIGMOD Conference*, pages 1151–1154, 2010.

[7] D. Yang, E. Rundensteiner, and M. Ward. Summarization and matching of density-based clusters in steaming environments. In *PVLDB 5(2)*, pages 121–132, 2011.

[8] D. Yang, E. Rundensteiner, and M. Ward. Shared execution strategy for neighbor-based pattern mining requests over streaming windows. In *TODS*, 2012. to appear.

[9] D. Yang, E. A. Rundensteiner, and M. O. Ward. Neighbor-based pattern detection for windows over streaming data. In *EDBT*, pages 529–540, 2009.

[10] D. Yang, E. A. Rundensteiner, and M. O. Ward. A shared execution strategy for multiple pattern mining requests over streaming data. *PVLDB*, 2(1):874–885, 2009.