ASSO: the Automated Schemaless Stream Overseer

Chiara Forresi
DISI - University of Bologna
Cesena, Italy
chiara.forresi@unibo.it

Matteo Francia
DISI - University of Bologna
Cesena, Italy
m.francia@unibo.it

Enrico Gallinucci
DISI - University of Bologna
Cesena, Italy
enrico.gallinucci@unibo.it

Matteo Golfarelli
DISI - University of Bologna
Cesena, Italy
matteo.golfarelli@unibo.it

ABSTRACT

In the big data era, real-time tracking and evaluation of streaming information are essential for delivering prompt insights. The schemaless nature of streaming data introduces substantial analytical challenges: since schemas are constantly changing in an unpredictable way, a continuous query issued over the stream can quickly become obsolete, losing purpose and effectiveness. In this demonstration, we present ASSO (Automated Schemaless Stream Overseer), a solution for dynamic monitoring and analysis of schemaless streams. ASSO considers the overlapping sliding window paradigm, interpreting data streams and offering an intentional view of their schemas, while dynamically adjusting continuous queries to accommodate schema evolution.

1 INTRODUCTION

Continuous monitoring and analysis of streaming data have become crucial in the big data era, where decision-makers need timely insights. Streaming data poses unique challenges, particularly in its often *schemaless* nature: each record comes with its own schema definition, which may differ significantly from the others [4, 14, 16]. This lack of structure leaves users unaware of the data flowing over time, making real-time analyses challenging. Stream data analysis typically operates within a specified time window, with the *overlapping sliding window* being one of the most widely used types [1]. A sliding window is a dynamic time interval of fixed duration subdivided into *panes* where, with each slide, the oldest pane is replaced with the most recent one. In this context, a concept of *state* is essential to store information from previous panes, enabling continuous calculations as the window slides forward.

Extracting information from a sliding window with schemaless data is challenging [13], particularly in streaming contexts where automating analysis of heterogeneous data (well-studied for static datasets [7, 10]) remains underexplored. First, it is difficult to gain a clear understanding of the data present in a window, as schemas can change over time. Therefore, it is essential to account for the unbounded nature of the input domain. Second, a continuous query that always targets the same attributes could lead to inaccurate and outdated results, as schema changes would not be reflected. Therefore, the query must dynamically adapt to the changes in the data over time. Ultimately, due to space and time constraints in processing the data within a window, it is not feasible to retain all data across the window. Instead, processing must rely on pre-aggregated data to produce outcomes representing the entire window.

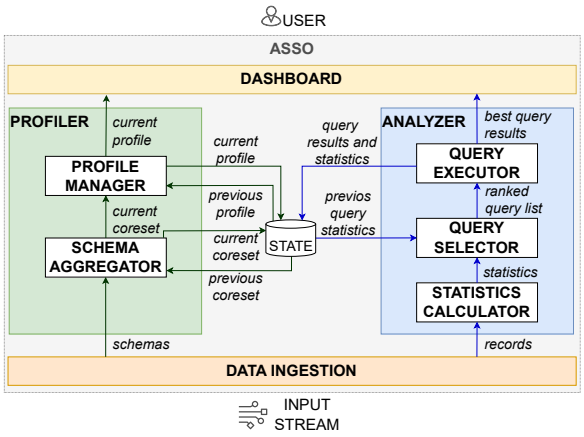

Figure 1: Architecture of ASSO.

In this demo paper, we propose ASSO as the *Automated Schemaless Stream Overseer*, a tool that enables the continuous monitoring and analysis of schemaless data streams. Given the dynamic nature of data schemas, ASSO interprets the data streams from an intentional perspective, providing a conceptual view of the schemas in the window. Additionally, ASSO executes a continuous dynamic query that adapts to the evolving data streams, delivering an extensional result while considering the functional dependencies between the data within the window. Both of these operations are performed while respecting the constraints of the streaming environment. In the demonstration, the user will have the opportunity to explore ASSO's functionalities from both intentional and extensional perspectives, experiencing how the system interprets and processes data in real-time while adapting to schema changes. A demonstration video and public access to the system are available at http://big.csr.unibo.it/asso.

2 THE ASSO SYSTEM

Figure 1 presents the architecture of ASSO, illustrating how data flows throughout the system. Data from the input stream is continuously processed by the data ingestion component, which serves as a data collector for ASSO's two main components: the *profiler* and the *analyzer*. In both cases, additive data structures are exploited to pre-aggregate the data in the stream to comply with the time and space constraints inherent to stream processing [18].

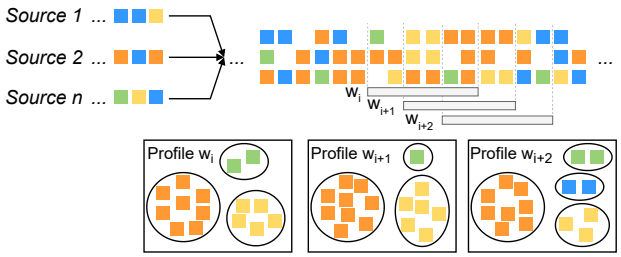

Figure 2: Intuition of schema profiling; squares represent data messages, colors are indicative of schema similarities.

- The **profiler** is responsible for providing intensional information to the dashboard. It processes the schemas from each pane, aggregates them in the schema aggregator using an additive data structure called coreset, and then passes the result to the profile manager to generate the current profile. The *profile* is a representation providing key insights about the schema of data in a high-variety context by grouping similar schemas together [9].
- The **analyzer** delivers extensional insights by processing records from each pane. It dynamically adapts a continuous query [3] to the current stream data and delivers results to the dashboard. The analyzer builds the state by selecting and executing relevant queries based on statistics collected by the statistics calculator. Finally, it outputs the best query result by aggregating all partial results.
- The **dashboard** ultimately provides the end user with a visual overview of the results generated by the profiler and analyzer components.

In ASSO's implementation, the profiler and analyzer are developed in Scala, using Kafka streams for efficient data processing; ingestion and output are managed through dedicated Kafka topics. The dashboard is a web page that connects to the output topic to deliver real-time visualizations. It also provides users with the ability to define input streams with varying frequencies and sources, as well as configure key parameters for the profiler and the analyzer, including the window size (duration and slide interval), state capacity (i.e., the percentage of input records that can be stored in the state), and the size of the group-by set considered by the analyzer.

2.1 Profiler

Figure 2 provides an intuition of the profiling process, where the schema profile of the heterogeneous stream is updated along three consecutive sliding windows. Profiles are produced by clustering the schemas extracted from the data (where a schema is the set of attributes in a record) according to the k-means algorithm. As we are considering schemaless data, the similarity between schemas is calculated with the Jaccard distance [15], which enables the comparison of two sets independently of their dimensionality. Given two schemas A_i and A_j, the Jaccard similarity is defined as $sim_J(A_i, A_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}$, and the distance is derived as $d_J(A_i, A_j) = 1 - sim_J(A_i, A_j)$. In the sliding window paradigm, running the clustering algorithm every time the window slides would be clearly inefficient (if not unfeasible) and would lack continuity (because the provided profiles would be hardly comparable across different windows). In ASSO, this is addressed with a two-phase approach [19].

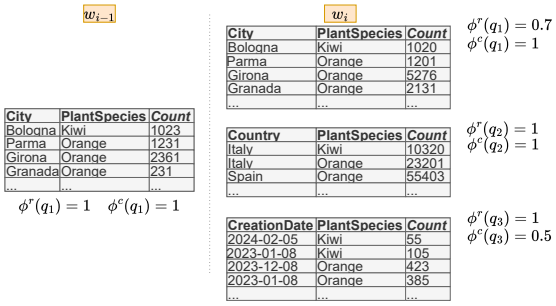

Figure 3: Examples of representativeness and continuity for possible queries in window w_i.

(1) First, the input data's volume is reduced by the **schema aggregator** in Figure 1, which pre-aggregates similar schemas into coresets, i.e., data structures providing compact representations of similar schemas that are grouped together. Coresets are updated dynamically by incorporating new data and removing expired elements.
(2) Second, rather than clustering the coreset from scratch for each window, the **profile manager** incrementally updates the clustering result (i.e., the profile) to reflect changes in the coreset. This enables the algorithm to adapt to evolving clusters dynamically, including adjustments to the number of clusters (also supporting merges or splits of clusters).

Since a cluster represents a set of schemas, the cluster representative consists of a schema where each attribute is associated with its support, defined as the percentage of schemas in the cluster in which each attribute is present. Therefore, the outcome of this module includes all the clusters that compose the profile, along with their similarity (calculated as the Weighted Jaccard similarity [6]) and the support of each attribute within the window.

2.2 Analyzer

The intrinsic variability in a schemaless stream requires constant adaptation of the continuous query to the intensional features of data in the current window. This entails two challenges: understanding when and how the query should change, and ensuring that new queries can be immediately executed over the whole window. Let us consider here multidimensional queries identified by a group-by set of k categorical attributes, where k is user-defined.

To address the first challenge, in every iteration the analyzer ranks the $\binom{n}{k}$ possible queries (where n is the number of available categorical attributes) based on the following desiderata.

(1) *Representativeness*: the query should cover as much data as possible within a window.
(2) *Continuity*: the query should ensure consistent results across consecutive windows, helping users track the evolution of the data over time.

The representativeness of a query q, $\phi^r(q) \in [0, 1]$, is calculated as its *support*, i.e., the percentage of data in which at least one attribute of the group-by set is non-null. The continuity of a query q, $\phi^c(q) \in [0, 1]$, is calculated as the average strength of *approximate functional dependencies* (AFDs) between its group-by set attributes and those in the query executed in the previous iteration [11]. The AFD from attribute a_i to a_j in a recordset R

is measured as $card(R, \{a_i\})/card(R, \{a_i, a_j\})$, where $card(R, A)$ denotes the number of distinct values of the set of attributes A in R (values closer to 1 indicate strong dependencies) [2].

Figure 3 gives an example of representativeness and continuity values for three possible queries for window w_i: q_1 is the same as the one executed in w_{i-1}, thus continuity is maximum, but representativeness has dropped (assuming the City attribute is losing support); q_2 and q_3 replace City with attributes with higher support (Country and CreationDate, respectively), with the first one achieving higher continuity thanks to the geographical functional dependency. Representativeness and continuity are uniformed into a single score, $score(q) = \alpha \cdot \phi^r(q) + (1-\alpha) \cdot \phi^c(q)$, where $\alpha \in [0, 1]$ is a parameter enabling user-based customized balancing of the two desiderata. The score is finally used to rank the set of possible queries and select the *best* one.

The second challenge is due to the fact that a new query with a different group-by set can be computed over the previous panes of the window only if partial results of the query have already been computed and saved in the state (given that saving the raw stream data into the state is not feasible in most streaming scenarios [5]). Computing and saving all possible queries is clearly inconvenient; thus, a strategy is needed to anticipate query evolution by finding the most promising queries among the possible alternatives, so that the state can be filled with the respective partial results. To this end, we employ a multidimensional knapsack approach [12] that builds on the above-discussed score to identify the ideal subset of queries that can be executed, given the space constraints of the state and the limited available time. Indeed, by maximizing the score of the pre-computed queries, we increase the chance that the new best queries can provide results over the whole window.

The process is outlined in Figure 1, where for each new pane, the **statistics calculator** module computes AFDs and attribute support using sampling-based techniques applied to the pane's records [17]. The **query selector** uses these statistics to calculate the score of potential queries and implements the multidimensional knapsack. At the end of each window, the **query executor** runs the selected queries, saves the results into the state (discarding those belonging to the expired pane), and returns the results of the best query. Notice that, when the analyzer is first started, the query initially selected and run is the one maximizing representativeness (as continuity cannot be computed in the absence of a previously executed query).

2.3 Dashboard

The dashboard presents the outcomes of the profiler and analyzer described above, providing a comprehensive view of the results; a snapshot is shown in Figure 4.

The output of the **profiler** is displayed on the left, showing attribute support in the current window and highlighting changes from the previous window to quickly identify variations in attribute occurrence. Moreover, the dashboard shows the current profile, providing a continuous view of the schemas within the window. It allows users to observe active schemas and their similarities, and with a click on a schema, it links the schema to the attributes it comprises, while also tracking structural schema changes over time.

The output related to the extensional information provided by the **analyzer** is in the central part, starting from the graph of approximate functional dependencies that illustrates the relationships between categorical attributes. This helps users identify dependencies within the data and enhances exploratory analysis. The feature allows users to adjust the level of approximation in the dependencies, offering a flexible view of attribute relationships tailored to their specific analysis needs. Additionally, the dashboard displays the outcome of the best query for the current window, aligning real-time data insights with the desiderata and user-defined preferences selected from the attribute's view. This ensures that the displayed query results are relevant to the user's analytical goals.

The dashboard is completed by two additional modules. One is the stream statistics display, which shows the frequency of records within each window pane, providing users with preliminary statistics to quickly assess the stream's volume and pace. The second module is the timeline, offering a graphical representation of the current window. Users can click on the timeline to navigate to previous panes, and the results in other modules will reflect that historical view. A second click returns the dashboard to real-time mode, resuming continuous monitoring.

3 DEMO PROPOSAL

In this demonstration, we examine real-world sensor data streams from a project in the precision agriculture domain, where the objective is to create a digital twin of the plants within orchards [8]; the system is designed to work with JSON data, but can be easily adapted to any other form of semi-structured data. In particular, we focus on three continuously monitored data streams: soil properties, plant health, and atmospheric conditions, which together generate an accurate digital representation of the orchard's plants. Variability arises within and across these sources due to differences in data collection methods and environmental conditions.

The first scenario provides an introductory experience, helping the user gain an initial understanding of the data flowing through the system. The three data sources exhibit varying degrees of similarity in their schemas over time, and the user can visually explore these variations. By analyzing the current window, the user can identify the schema profile, assess the degree of similarity between schemas, observe the appearance or disappearance of attributes, and track shifts in the functional dependencies between them. This scenario allows users to intuitively engage with the system and build a foundational understanding of the data.

The second scenario provides a more interactive and analytical experience, supporting the user in gaining deeper insights from an extensional perspective. The user is tasked with understanding and interacting with the results of the continuous query that best represents the data at any given moment. The user can observe how this query evolves and is encouraged to actively engage with ASSO. Users can suggest adjustments such as hiding certain attributes in the next window or marking others as mandatory. This interactive feedback loop offers users greater control over the analysis process, enhancing their ability to guide the system's behavior.

Both scenarios are designed to be dynamic, allowing users to adjust various parameters related to the input streams. Users can selectively enable specific data streams to focus on particular subsets of the data, as well as modify the frequency at which the data is received. The window size and state capacity can also be adjusted enabling the exploration of different temporal spans and governing the spatial constraint on both the analyzer and profiler, thus opening the possibility of varying analytical outcomes (as different parameter values influence the selection

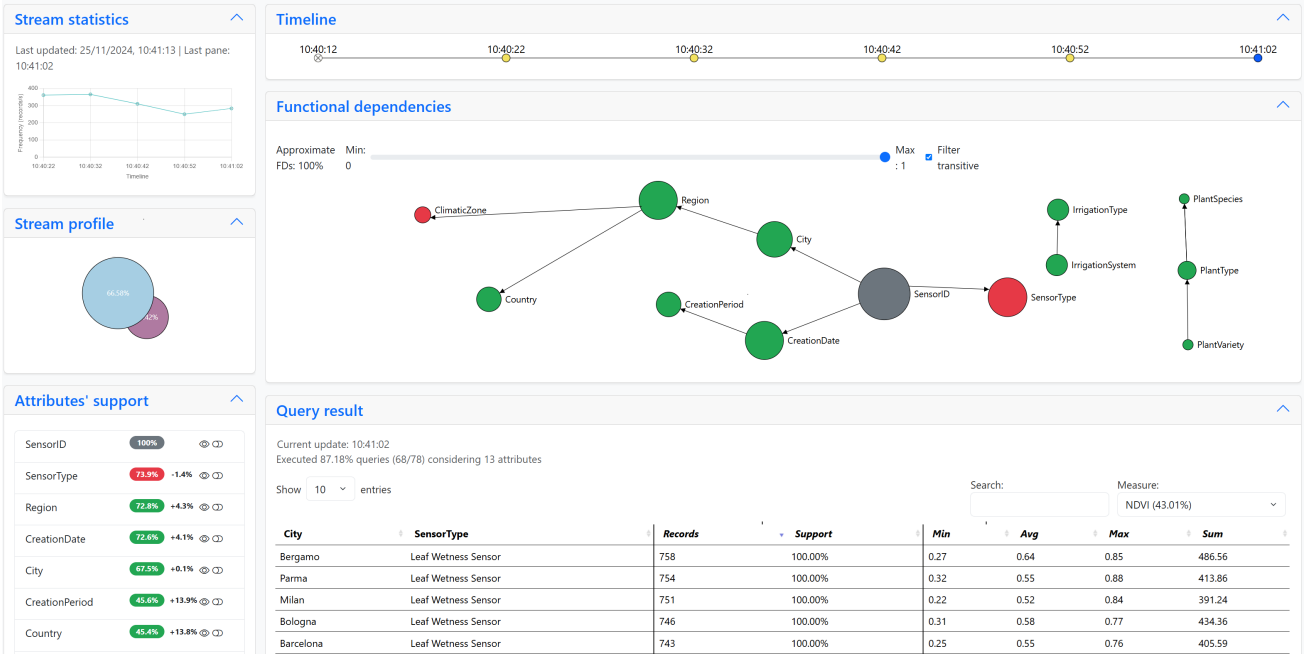

Figure 4: A snapshot of ASSO's dashboard.

of feasible queries). Finally, varying the parameter α allows users to influence the choice of the best query, prioritizing either representativeness or continuity.

ACKNOWLEDGMENTS

This study was carried out within the Agritech National Research Center and received funding from the European Union Next-Generation EU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1032 17/06/2022, CN00000022). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

REFERENCES

[1] Tyler Akidau, Slava Chernyak, and Reuven Lax. 2018. *Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing* (1st ed.). O'Reilly Media, Inc.

[2] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. 2014. Similarity measures for OLAP sessions. *Knowl. Inf. Syst.* 39, 2 (2014), 463–489. https://doi.org/10.1007/S10115-013-0614-1

[3] Arvind Arasu, Shivnath Babu, and Jennifer Widom. 2006. The CQL continuous query language: semantic foundations and query execution. *VLDB J.* 15, 2 (2006), 121–142. https://doi.org/10.1007/S00778-004-0147-Z

[4] David Corral-Plaza, Inmaculada Medina-Bulo, Guadalupe Ortiz, and Juan Boubeta-Puig. 2020. A stream processing architecture for heterogeneous data sources in the Internet of Things. *Comput. Stand. Interfaces* 70 (2020), 103426. https://doi.org/10.1016/J.CSI.2020.103426

[5] Camil Demetrescu and Irene Finocchi. 2007. CHAPTER 8 Algorithms for Data Streams. https://api.semanticscholar.org/CorpusID:16145668

[6] Chiara Forresi, Matteo Francia, Enrico Gallinucci, and Matteo Golfarelli. 2023. Streaming Approach to Schema Profiling. In *New Trends in Database and Information Systems - ADBIS 2023 Short Papers, Doctoral Consortium and Workshops: AIDMA, DOING, K-Gals, MADEISD, PeRS, Barcelona, Spain, September 4-7, 2023, Proceedings (Communications in Computer and Information Science)*, Alberto Abelló, Panos Vassiliadis, Oscar Romero, Robert Wrembel, Francesca Bugiotti, Johann Gamper, Genoveva Vargas-Solar, and Ester Zumpano (Eds.), Vol. 1850. Springer, 211–220. https://doi.org/10.1007/978-3-031-42941-5_19

[7] Chiara Forresi, Enrico Gallinucci, Matteo Golfarelli, and Hamdi Ben Hamadou. 2021. A dataspace-based framework for OLAP analyses in a high-variety multistore. *VLDB J.* 30, 6 (2021), 1017–1040. https://doi.org/10.1007/S00778-021-00682-5

[8] Matteo Francia, Joseph Giovanelli, and Matteo Golfarelli. 2022. Multi-sensor profiling for precision soil-moisture monitoring. *Comput. Electron. Agric.* 197 (2022), 106924. https://doi.org/10.1016/J.COMPAG.2022.106924

[9] Enrico Gallinucci, Matteo Golfarelli, and Stefano Rizzi. 2018. Schema profiling of document-oriented databases. *Inf. Syst.* 75 (2018), 13–25. https://doi.org/10.1016/J.IS.2018.02.007

[10] Enrico Gallinucci, Matteo Golfarelli, and Stefano Rizzi. 2019. Approximate OLAP of document-oriented databases: A variety-aware approach. *Inf. Syst.* 85 (2019), 114–130. https://doi.org/10.1016/J.IS.2019.02.004

[11] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Comput. J.* 42, 2 (1999), 100–111. https://doi.org/10.1093/COMJNL/42.2.100

[12] Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. *Multidimensional Knapsack Problems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 235–283. https://doi.org/10.1007/978-3-540-24777-7_9

[13] Taiwo Kolajo, Olawande J. Daramola, and Ayodele Ariyo Adebiyi. 2019. Big data stream analysis: a systematic literature review. *J. Big Data* 6 (2019), 47. https://doi.org/10.1186/S40537-019-0210-7

[14] Jana Kosticová. 2024. Low-Latency Streaming Evaluation of JSONPath Queries. In *Proceedings of the 24th Conference Information Technologies - Applications and Theory (ITAT 2024), Drienica, Slovakia, September 20-24, 2024 (CEUR Workshop Proceedings)*, Lucie Ciencialová, Martin Holena, Róbert Jajcay, Tatiana Jajcayová, Martin Macaj, Frantisek Mráz, Richard Ostertág, Dana Pardubská, Martin Plátek, and Martin Stanek (Eds.), Vol. 3792. CEUR-WS.org, 43–51. https://ceur-ws.org/Vol-3792/paper5.pdf

[15] Michael Levandowsky and David Winter. 1971. Distance between Sets. *Nature* 234, 5323 (November 1971), 34–35. https://doi.org/10.1038/234034a0

[16] Anton Popov and Radoslav Dimitrov. 2022. The Role of NoSQL Databases in Supporting Real-Time Analytics in Cloud Computing Platforms: Performance and Efficiency Considerations. *Emerging Trends in Machine Intelligence and Big Data* 14, 4 (Apr. 2022), 52–60. https://orientreview.com/index.php/etmibd-journal/article/view/91

[17] A Shlosser. 1981. On estimation of the size of the dictionary of a long text on the basis of a sample. *Engineering Cybernetics* 19, 1 (1981), 97–102.

[18] Jonghem Youn, Junho Shim, and Sang-goo Lee. 2018. Efficient Data Stream Clustering With Sliding Windows Based on Locality-Sensitive Hashing. *IEEE Access* 6 (2018), 63757–63776. https://doi.org/10.1109/ACCESS.2018.2877138

[19] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, H. V. Jagadish and Inderpal Singh Mumick (Eds.). ACM Press, 103–114. https://doi.org/10.1145/233269.233324