Automatically Surfacing Opportunities for Improvements In Internet-Scale Applications

Vipul Harsh[1], Sayan Sinha[1,2], Henry Milner[1], Haijie Wu[1],
B Aditya Prakash[1,2], Vyas Sekar[1,3], Hui Zhang[1,3]
[1]Conviva [2]Georgia Tech [3]Carnegie Mellon University

Abstract

Modern Internet services generate massive volumes of observability data, yet identifying *opportunities* for business performance improvements remains elusive. In many cases, such insights manifest only within sub-populations defined by derived attributes that cannot be predefined, might evolve over time, and often cannot be exhaustively enumerated ahead of time. Unfortunately, existing commercial and research systems fall short in one or more aspects of generating such improvement opportunities: expressiveness, automation, and scalability. We present a vision for automatically surfacing opportunities for improvements to tackle these seemingly conflicting and intractable requirements. We highlight the early promise from a proof-of-concept system, showing evaluation on three real-world services and discuss open challenges for future work.

CCS Concepts

• **Networks** → **Network monitoring**; **Network reliability**; **Network management**.

1 Introduction

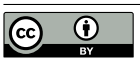
Modern Internet-scale services across many business verticals (e.g., E-commerce, OTT video platforms, ride-sharing) are increasingly getting large and complex with multiple sub-systems, geo-distributed backends, third-party services, intricate user workflows among others. To monitor such systems, operators invest significant effort in observability, product analytics, and data warehousing tools. Such services collect large volumes of telemetry (metrics, events, logs, traces) collected from multiple vantage points (client, backend, network).

However, operators and business owners continue to be frustrated in terms of the value these provide, as existing tools consistently fail to surface *opportunities* for improving Key Performance Indicators (KPIs) relating to user experience, user engagement, performance, and business outcomes. These tools are very valuable in identifying "smoking gun" issues such as outages or large deviations but fail to identify opportunities for improvements that actually matter.

To see why, consider two scenarios inspired by real-world phenomena in a large e-commerce application. In the first case, users on a specific app version who trigger a *particular sequence* of actions (e.g., Click on buy -> Experience browser error -> Timeout) cause a disproportionate number of backend database deadlocks. In another scenario, returning users, who are accessing the app for a *second or later time*, experience higher error-rates. These are not anomalies in the classical sense as they do not manifest as significant deviations in KPIs for any obvious user cohort of interest. Over a long time range, however, these types of occurrences can potentially result in significant drop in user satisfaction, and ultimately revenues.

Such nuanced "long tail" *opportunities* to improve business outcomes are the norm not the exception in Internet-scale services [13]. We define an opportunity as any discernible pattern that suggests a concrete avenue to improve some KPIs of interest (e.g., performance and user engagement). Crucially, this definition not only encompasses anomalous patterns that signal outages, but also non-anomalous patterns which nevertheless can lead to improvements. Capturing a diverse set of opportunities, such as the scenarios described earlier, requires expressing a combinatorially large set of user-journeys that are not just characterized by user metadata, but the unique sequence of actions they perform, the transient state of the backend services they hit, and the parameters associated with the requests they send.

Ideally, we want to have an **expressive**, **scalable** and **automated** mechanism to spot such opportunities to guide potential interventions and improvements. Unfortunately, simultaneously achieving all three goals is easier said than done, and even achieving just two of these may be impossible with current solutions. For example, existing OLAP-style analytics systems [1] rely on pre-computed static attributes to slice and dice data through manual query interfaces. As such, these cannot express the sequential/stateful requirements of the scenarios above. On the other hand, there are more expressive stateful analytics schemes [2, 4, 26]. However, they are expensive and require significant manual effort. While



there are efforts at automating detection and root cause analysis [5, 8, 17, 19, 20, 25, 27, 28], they too rely on a small, scoped set of precomputed attributes and cannot capture the complexities of user-journeys described above.

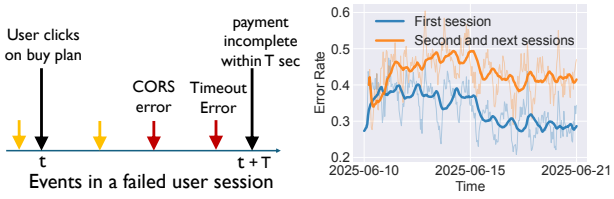
In this paper, we articulate our vision to automatically surface business-critical insights to drive continuous performance improvement in Internet-scale services. To achieve this goal, we argue for a paradigm shift, moving away from analysis based on predefined static attributes of the telemetry database, towards on-demand data slicing to generate **derived attributes** on the fly. Computing derived attributes allows us to capture the space of all possible user-journeys. We identify three major challenges in designing such a system spanning algorithms, ML, and system-design —

(1) **Hypothesis generation**: The space of hypotheses based on derived attributes is vast and potentially infinite. One needs (1) an efficient way to express hypotheses in this space using a concise language, and (2) automated methods to narrow down potentially insightful regions in this space.
(2) **Scalable attributes computation**: On-the-fly computation of derived attributes necessary for testing generated hypotheses requires efficient data processing tools capable of handling complex and stateful logic efficiently.
(3) **Opportunity finder**: The system needs to generate hypotheses from the computed derived attributes and output opportunities which are truly related to KPI degradations, using suitable ML models.

As a pragmatic first step towards solving these challenges, we envision a human AI collaborative approach where human operators can express their intents of capturing certain classes of behavioral opportunities and the system automatically surfaces the important ones from those classes. We outline an end-to-end design of a system, sketching the key components in our solution to address hypothesis generation, scalable attributes computation and opportunity finder (§ 4). We implement a proof-of-concept prototype combining simple yet expressive hypothesis generation based on derived attributes, a basic stateful attributes computation engine that is efficient, and an "ensemble of experts" approach to validate hypotheses to find opportunities. We show that even this preliminary design reveals interesting opportunities in real-world applications that would otherwise be difficult to discover with existing tools (§ 5).

2 Motivation

In this section, we present two illustrative real-world scenarios (Figure 1) from a production system. We use anonymized data from a large application-level monitoring and analytics service provider. The provider gives its customers (who run global-scale applications) fine-grained visibility into the digital experience observed by their users in the field, enabling its customers to create custom KPIs/metrics (e.g., error, conversion, timing) to track business outcomes [9, 15].



(a) Example 1 (b) Example 2

Figure 1: Illustrative examples from a production setup. (a) Example 1: Event timeline for a user session ending in a payment conversion failure: we hypothesize that the initial CORS error leads to client-side retries, culminating in the timeout_error, which ultimately leads to payment not finishing in the 10-second window. (b) Example 2: How a specific error affects a group of users differently from others. For a specific OS version, error rate is higher when a user accesses the app a second time compared to the first time.

Example 1: Errors impacting a content-platform service. Customer X runs a digital subscription service. They created a KPI metric to check how "fast" new users were able to subscribe. To this end, they created a binary metric to track whether a prospective subscriber who "clicked on buy plan" button was able to complete "payment" within a specific time. When debugging an unrelated problem, we saw that many sessions where this failure metric was flagged (i.e., payment did not occur within T seconds) showed a specific event sequence: users who clicked buy_plan, then encountered a web_cors_error, and subsequently a timeout_error, were not finishing the payment flow (Figure 1a). The initial CORS error (a browser security feature) triggered client-side retries that led to the timeout. Now, a non-trivial fraction $1 - 2\%$ users were impacted, but the overall payment failure rate was not high enough to trigger an alert.

Example 2: Figure 1b shows a different but subtle problem that we observed for a different customer Y. Here, the customer had defined a global error metric to track user-sessions that experience application errors. Once again, we observe a subtle but noteworthy pattern in the error rate — users on a specific OS version exhibit higher error rates in their second or subsequent session compared to their first visit (affecting approximately 35% of sessions on that OS). We speculate that this return pattern may trigger distinct backend workflows, such as cache accesses or follow-up onboarding steps, which could be contributing to the observed errors.

Note that these incidents were not covered by any existing capabilities developed in-house by the service provider or deployed by the customer. Rather, these were uncovered serendipitously by customer-facing and research investigation teams who were trying to troubleshoot unrelated issues.

Discovering such opportunities for improvement today is either absent, or at the very least painstaking and involves significant manual effort. In many cases, the KPI drops for specific user behaviors will go unnoticed. Even if an operator notices the KPI change, they may have no clear starting point. Guided only by intuition, they might manually craft a few SQL queries to test simple hypotheses based on precomputed attributes present in the telemetry— 'Is it an issue with Android users? Or users in Germany?'.

The operator then is left to guess at more complex behavioral or temporal patterns, a process that is ad-hoc, unscalable, and entirely dependent on the operator's luck and prior experience with the system. Even if an operator were to somehow conceive a few potentially useful hypotheses, they face a significant barrier— each new hypothesis requires writing a complex, bespoke query that is computationally expensive. In Example 1, to test the behavioral hypothesis for payment failures, an operator must construct a multi-stage query involving stateful scans over the event table, following by joins with the KPI metrics table. In Example 2, an operator must first write a subquery on the sessions table, grouping the sessions by user-id and then apply a window function to identify which session was their second. A seemingly natural solution is to use event-processing systems [2, 4, 26] to pre-compute every possible attribute that could be used to validate some hypothesis and store them as regular columns in the database. However, this approach is infeasible: (1) the space of candidate attributes is combinatorially vast, even potentially innumerable, and (2) the attribute itself may evolve as user-behavior changes (e.g., frequent vs infrequent user).

Finally, we highlight another noteworthy aspect of such incidents. Given the subtle and nuanced nature of these issues, they "fly under the radar" of classical anomaly and deviation detection techniques. Over time, however, the cumulative effect of many such issues will inevitably lead to poor user-experience. Hence, we want to automatically uncover such opportunities for intervention to improve the user-experience.

Ideally, we want a system to identify such opportunities that satisfies three natural goals (Table 1): (1) **expressivity** to easily define and test a hypothesis for any interesting user journey; (2) **automation** to systematically generate and test thousands of hypotheses, moving beyond manual guesswork; and (3) **scalability** to efficiently compute several attributes simultaneously that are needed to test those hypotheses.

Table 1: Comparison of Related Work Categories Against System Goals

Solution Category	Expressivity	Scalability	Automated
Classical OLAP systems (e.g., SQL, Spark [1])	X	✓	X
Stateful event processing systems (e.g., [2, 4, 26])	✓	X	X
Automated anomaly detection/root cause analysis (e.g., [5, 8, 17, 19, 20, 25, 27, 28])	X	✓	✓
LLM-based insight detection e.g., InsightPilot [22])	X	X	✓

3 Problem Formulation

In this section, we define our requirements to understand why current solutions fall short. We begin by formalizing a general data model, applicable to many observability settings in large-scale services.

Data model: We assume that the most atomic unit of data is an *event* $e \in \mathcal{E}$, where each event is a structured object with attributes such as timestamp, user ID, request ID. In our setting, these are client-side events streaming from collection agents on end-user devices, but more generally, events could be structured objects (e.g., JSON), sensor readings, log updates, etc. We assume that events arrive as a time-ordered stream $\mathcal{S} = \{e_1, e_2, \ldots, e_n\}$.

Events are grouped into partitions $\{C_g\}$ based on some natural event attributes g, such as user ID, session ID, request ID etc. This grouping is sometimes referred to as *sessionization*. Each group C_g is internally ordered by event timestamps to reflect sequential behavior. For each group C_g, we compute a vector of metrics $\mathbf{m}(C_g) \in \mathbb{R}^d$, where each component represents a KPI such as session duration, user activity frequency, or backend response time.

Analytics Goal: We envision a data analytics system that generates a set of hypotheses $\mathcal{H}$, where each hypothesis guides computation of the derived attributes required to test the hypothesis e.g. a binary attribute for whether the user is frequent to test whether a KPI is worse for infrequent users. The system then evaluates the hypotheses and surfaces corresponding opportunities for operators of internet-scale services.

Static vs. Derived Attributes: The central challenge here is w.r.t. the types of hypotheses we generate and validate to surface opportunities for improvement.

Classical data analytics workflows rely on hypotheses based on *static attributes* that are predefined and stored directly in the database e.g. device_type, region, or app_version [5, 8, 17, 19, 20, 25, 27, 28]. More precisely, **static attributes**

are directly observable within individual events, such as operating system, device type, or browser family.

Recall, however, in Example 1, the errors occurred for users who were going through a particular sequence of events. Similarly, in Example 2, the errors occurred within a specific, transient phase of the user lifecycle. Both involve subtle and complex aspects of the session and user history and sequence of actions that cannot be captured by the conventional lens of static attributes as defined above.

Instead, we argue that we should focus on more expressive hypotheses that are based on **derived attributes**– defined by behavioral, temporal, or non-local context that are not present as simple database columns. These must be computed on the fly from raw event data and other data tables. These hypotheses are necessary to explain subtle shifts or degradations in computed metrics $\mathbf{m}(C_g)$. More concretely, to construct such rich hypotheses, we consider three specific types of derived attributes.

- **Indirect:** Attributes computed from static attributes using zeroth-order predicates, e.g., `night` = (`hour`$(e) < 9$) OR (`hour`$(e) > 21$).
- **Stateful:** Attributes defined over event sequences, computed via stateful queries e.g., session duration, time spent waiting in rebuffering state, or number of retries before success. These depend on the temporal structure of the groupings C_g.
- **Non-local:** Attributes computed by referencing external or global context—e.g., whether a session occurred when the backend nodes were exhibiting high CPU usage, or the result of network request accesses corresponding to top 20 URLs accessed during login across all sessions.

Unfortunately, such derived attributes pose stringent requirements for data processing efficiency, defined by scalable context-aware stateful computations. We call a computation context-aware if the data needs to be filtered or grouped based on other telemetry sources e.g. grouping user-sessions based on error-timestamps from backend logs. Context-aware processing can require complex expensive joins between multiple tables. For stateful computation, while powerful interactive analytics tools like Kusto (KQL) [4], EQL [2] and TLB [26] provide the necessary expressivity, they are not architected to compute thousands of them. Hence, they do not meet our requirements of automation and scalability. LLM-based frameworks such as InsightPilot [22] aim to automate the analysis process, but rely on tools that operate only over static attributes [14], preventing execution of expressive queries involving derived attributes.

Overall problem statement: Summarizing, we frame the problem of automatically surfacing opportunities in terms of generating and validating interesting and useful hypotheses that link some derived attributes to degradations in key performance indicators (KPIs). More precisely, given a grouping over events (e.g. sessions, users, backend-services), our objective is to automatically generate a rich set of candidate hypotheses, efficiently compute corresponding derived attributes and finally test those hypotheses.

4 Design

Figure 2 shows a high-level view of our system design with the key components. Next, we highlight how we tackle key challenges in hypotheses generation, attribute computation, and hypothesis validation for surfacing opportunities.

4.1 LLM-Assisted Hypothesis Generation

The space of plausible hypotheses based on derived attributes is vast and combinatorial, hence we need to narrow down useful regions. We further observe that hypotheses generation is in fact a domain-specific problem: an anomalous signal in one domain may not be anomalous in another. For instance, a temporary spike that comes back to its usual value might not be alert-worthy for a "heart rate" metric, but would indicate a severe issue for a metric indicating blood infection rates. We tackle this problem by leveraging the world knowledge embedded in large language models (LLM) and incorporating the natural language metadata about the domain. Specifically, we design a LLM-assisted hypothesis generator that synthesizes candidate hypotheses classes given the following inputs:

- **Hypothesis templates:** A library $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ of parameterized hypotheses classes, where each template T_j defines a family of hypotheses over event-groups C_g (sessions, users and so on). Each T_j is associated with a computation DAG D_j that specifies how to compute the derived attributes required to test all hypotheses in that class. These hypotheses templates are pre-defined and are a part of the system.
- **Metadata:** Natural language artifacts such as KPI descriptions, user reviews, source code, and telemetry schema.
- **Human prompts:** Natural language instructions that guide the LLM towards generating hypothesis-class parameters, targeting potentially useful regions in the hypotheses space.

Given a prompt, such as "find patterns associated with high rebuffering," the LLM selects a subset of relevant templates $\mathcal{T}' \subseteq \mathcal{T}$ and instantiates each $T_j \in \mathcal{T}'$ by filling in its parameters to produce a corresponding set of concrete hypothesis-classes. These concrete-hypothesis classes will then be passed to the attributes computation engine (§ 4.2) for execution (see section 4.2). By leveraging human intent, system metadata, and prior context, this design enables the LLM to generate hypotheses that are semantically grounded and better aligned with domain-relevant diagnostic needs.

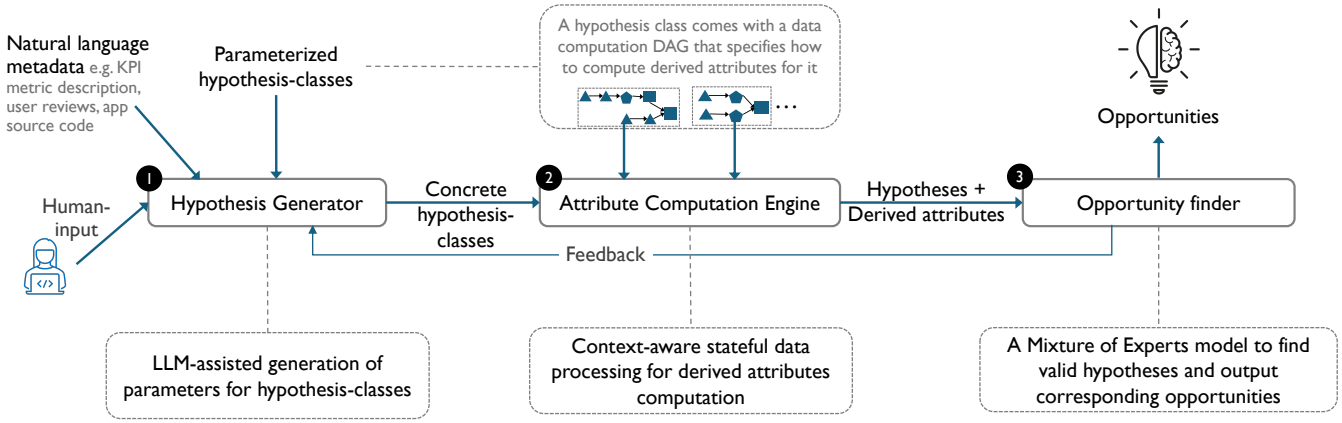


Figure 2: Our envisioned system consists of three components: (1) an LLM-assisted hypothesis generator that generates concrete hypothesis-classes based on a library of parameterized hypothesis-classes, textual metadata and human prompt (2) an attributes computation engine that performs context-aware data processing to compute derived attributes necessary to test generated hypotheses, and (3) an opportunity finder that applies ML models (e.g., regression, outlier detection) to test hypotheses and surface opportunities corresponding to the valid hypotheses.

4.2 Stateful and Expressive Computation

The attribute computation engine receives a set of concrete hypothesis-classes from the hypothesis generator. It takes the associated computation DAG for each hypothesis-class that specifies how to compute its corresponding derived attributes and compiles all of them into one giant computation DAG. It reuses computation whenever possible, merging duplicate nodes across the individual DAGs for efficiency. Note that for attributes involving temporal or behavioral context, some nodes in the DAG would have to be computed using stateful operators over event streams. Conceptually, the attribute computation engine will create new columns in the database that correspond to the derived attributes.

4.3 Mixture-of-Experts Opportunity Finder

The Opportunity Finder receives from the attributes computation engine, derived attributes (e.g. frequency of use in the last week for each user), corresponding to the hypotheses classes generated by the Hypothesis generator.

To narrow down the relevant hypotheses from the hypothesis-classes that may be leading to KPI degradation, the Opportunity Finder applies a mixture of "experts" approach i.e., suite of statistical and machine learning pipelines; to assess the relevance of computed hypotheses. These can include trend analysis, outlier detection, change point detection, similarity analysis between cohorts (e.g., treatment vs. control) etc. If the validator detects a statistically significant effect (e.g., uplift or drop beyond a configured threshold), the corresponding hypothesis h_i is marked as an opportunity and passed on for visualization or downstream action.

Feedback loop: We envision a feedback loop where the hypothesis generator receives feedback from the Opportunity Finder to refine its generation policy. After each hypothesis is assessed, i.e., quantified by statistical metrics such as effect size or correlation strength, the feedback is used to bias the LLM towards more informative hypotheses subclasses.

5 Proof of concept

We implement a proof of concept and evaluate it on production data. We show that even our preliminary design can uncover novel opportunities in real-world applications.

5.1 Implementation

Hypothesis generation: We implemented hypotheses based on two types of derived attributes:

- Event patterns: This generates attributes related to sequence of user events that may correlate with failures in a KPI metric (e.g., login success/failure). As part of the input, the user can specify the length of the pattern sequence k, an event-to-token mapping function M, and a filtering function F that specifies event-types to ignore. The system generates all relevant unique event sequences of length k that appear in the data.
- User lifecycle patterns: This generates attributes related to patterns in user engagement across sessions (e.g., first-time vs. repeat usage). As a part of the input, inactivity threshold f used to define sparse users (e.g., no activity for $>$ 30 days), and a session boundary k that defines lifecycle-based attributes can be specified. Users with session indices $\leq k$ (e.g., first, second etc.) are treated as new

users in contrast to users with session indices $> k$ (long-term users). We used Llama 3.2 [18] for generating the hypothesis class parameters.

Attribute computation: we use a tailored, efficient strategy for stateful processing to compute the above attributes:

- For 1, we use an approximation route– it first takes a random sample of all sessions within a time window and then employs the Aho-Corasick string-matching algorithm [6] to efficiently compute the occurrence counts for all n-gram sequences within the sampled sessions, separately for faulty and normal sessions.
- For 2, we use a partition-based analysis approach– it first counts the number of occurrences of each user ID in the retrieved data, then computes the given attribute based on the count.

Opportunity Finder: Given the attributes, we use a heuristic validation layer to surface opportunities for improvement:

- Expert 1: An event-sequence pattern is deemed a significant opportunity if it occurs in at least 10% of faulty sessions and its rate of occurrence in faulty sessions is at least 2x higher than its rate in normal sessions.
- Expert 2: We deem the binary attribute-hypothesis as an opportunity, if (1) there are outliers in the KPI metric breakdown for user-cohorts induced by the binary derived attributes (e.g. frequency of use, first-time) or (2) if the absolute difference in the KPI metric between the derived user-cohort and all users is $> 25\%$

The final output of our system is a unified list of validated opportunities from all experts.

5.2 Evaluation on real-world data

We ran the above workflow for one week of production data for 3 digital services for various KPIs. To prune the analysis, we first find user-cohorts based on static attributes with significant number of users. For each such user-cohort, we report a subset of the opportunities identified by our prototype in table 2, those we (manually) deemed to be insightful.

Overall, expert 1 surfaced opportunities for 18 scenarios across all services, producing the top-5 ranked hypotheses, for each scenario. Using manual inspection, we deemed opportunities in 5 out of 18 scenarios to be useful. Expert 2 surfaced opportunities for 20 scenarios for each service, out of which 8 were useful on average. While the false positive "rate" may seem high at first, it imposes a very reasonable cognitive burden; i.e., the effort to manually inspect ≈ 20 opportunities for improvements over the course of one week for a large service provider is quite reasonable. Further, operators with domain knowledge can quickly discard false leads.

KPI Impacted	Opportunity Lead
Login success rate	High error rate in Step 1 of Login (OTP verification)
Buy plan click to payment complete conversion rate	User sequence: Buy plan → CORS error → Timeout error leads to conversion failure
Session crash rate	API to retrieve user location fails
Subscription start to finish conversion rate	Subscription API returns failure (unexpected response)
Subscription viewed to payment conversion rate	New UI version of subscription page leads to conversion failure
Global error rate	High error rate in second and later sessions
User-induced error rate	Higher error rate for first-time users
Login error rate	Gradual increase in error rate for first-time users on new app version

Table 2: Opportunity leads identified by our prototype as potential areas for improvement.

Scalability: Expert 1 processed and analyzed > 1000 hypotheses in less than 10 minutes. Expert 2 processed and analyzed a batch of 250 hypotheses in less than 3 minutes per KPI metric on average. While these numbers are preliminary and there's a lot of room for optimization, they still suggest that our proposed system will be scalable.

6 Discussion and future work

Realizing the full potential of opportunities-generation requires significant research in several key areas:

LLM-assisted feature generation: A key next step is to leverage more capable LLMs for more effective exploration of the hypothesis space. Related approaches exist in other domains—for example, RcaCopilot [12] uses an ML model to select RCA handlers based on incident type. In our setting, effective use of system metadata may be enabled by hierarchical [11] and multi-modal LLMs [29].

Rigorous validation and causal inference: To move beyond correlation, integrating formal causal inference methods like synthetic controls [7] or doubly robust estimation [16] can be helpful to reduce false positive rates.

More complex event patterns: Future work should focus on using an expressive declarative language for defining hypotheses. This includes incorporating temporal logic (e.g., "event A occurs until event B") and richer predicates to correlate different datasets such as client-side user-experience metric and backend load. Some earlier works have developed primitives to express specific kinds of data slicing e.g., PivotTracing [23] employs a happens-before primitive, [21] uses primitives to query the web.

User interfaces: A new class of UIs is needed to move beyond manual query writing similar to [3] but for writing new attribute templates. UIs can also be very helpful for visual exploration of hypotheses based on derived attributes, as [24] does for static attributes.

Scalable data processing: Computing thousands of derived attributes requires new data processing systems designed for high-throughput, stateful, context-aware and scalable query execution, focusing on approximation algorithms, query planning and resource management techniques.

Opportunity finder for Agentic systems: Our framework is naturally suited for troubleshooting agentic AI systems to automatically discover complex failure modes in systems whose behavior is defined by high-dimensional data like prompts and generated plans [10].

Privacy-Preserving Community Insights: One avenue of future work is to create a framework for sharing abstract hypothesis-class templates or signatures across organizations. This would allow a community to crowdsource a library of effective patterns for discovering long-tail issues without sharing sensitive raw data.

Incorporating auxiliary datasets: The system's expressivity can be enhanced by joining client-side telemetry with auxiliary datasets, such as performance and health metrics for third-party services, backend telemetry, etc.

References

[1] Apache Spark. https://spark.apache.org/.

[2] Event Query Language (EQL). https://www.elastic.co/docs/explore-analyze/query-filter/languages/eql.

[3] GoFlow. https://go-flow.co/.

[4] Kusto Query Language overview. https://learn.microsoft.com/en-us/kusto/query/?view=microsoft-fabric.

[5] Root Cause Analysis with DoWhy, an Open Source Python Library for Causal Machine Learning. https://www.pywhy.org/dowhy/v0.12/.

[6] A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[7] M. Amjad, D. Shah, and D. Shen. Robust synthetic control. *Journal of Machine Learning Research*, 19(22):1–51, 2018.

[8] R. Bhagwan, R. Kumar, R. Ramjee, G. Varghese, S. Mohapatra, H. Manoharan, and P. Shah. Adtributor: Revenue debugging in advertising systems. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 43–55, 2014.

[9] P. Carbone et al. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society TCDE*, 36(4), 2015.

[10] M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, K. Keutzer, A. Parameswaran, D. Klein, K. Ramchandran, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

[11] S. Chakraborty, J. W. Stokes, L. Xiao, D. Zhou, M. Marinescu, and A. Thomas. Hierarchical learning for automated malware classification. In *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pages 23–28. IEEE, 2017.

[12] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, et al. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 674–688, 2024.

[13] J. Dean and L. A. Barroso. The tail at scale. *Communications of the ACM*, 56:74–80, 2013.

[14] R. Ding, S. Han, Y. Xu, H. Zhang, and D. Zhang. Quickinsights: Quick and automatic discovery of insights from multi-dimensional data. In *Proceedings of the 2019 international conference on management of data*, pages 317–332, 2019.

[15] O. Etzion et al. Event-driven architectures and complex event processing. In *IEEE SCC'06*, 2006.

[16] M. J. Funk, D. Westreich, C. Wiesen, T. Stürmer, M. A. Brookhart, and M. Davidian. Doubly robust estimation of causal effects. *American journal of epidemiology*, 173(7):761–767, 2011.

[17] Y. Gan, M. Liang, S. Dev, D. Lo, and C. Delimitrou. Sage: practical and scalable ml-driven performance debugging in microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 135–151, 2021.

[18] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[19] V. Harsh, W. Zhou, S. Ashok, R. N. Mysore, B. Godfrey, and S. Banerjee. Murphy: Performance diagnosis of distributed cloud applications. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 438–451, 2023.

[20] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '04, page 219–230, New York, NY, USA, 2004. Association for Computing Machinery.

[21] L. V. Lakshmanan, F. Sadri, and I. N. Subramanian. A declarative language for querying and restructuring the web. In *Proceedings RIDE'96. Sixth International Workshop on Research Issues in Data Engineering*, pages 12–21. IEEE, 1996.

[22] P. Ma, R. Ding, S. Wang, S. Han, and D. Zhang. Insightpilot: An llm-empowered automated data exploration system. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 346–352, 2023.

[23] J. Mace, R. Roelke, and R. Fonseca. Pivot tracing: Dynamic causal monitoring for distributed systems. *ACM Transactions on Computer Systems (TOCS)*, 35(4):1–28, 2018.

[24] S. Malik, F. Du, M. Monroe, E. Onukwugha, C. Plaisant, and B. Shneiderman. Cohort comparison of event sequences with balanced integration of visual analytics and statistics. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pages 38–49, 2015.

[25] A. Manousis, H. Shah, H. Milner, Y. Li, H. Zhang, and V. Sekar. The shape of view: an alert system for video viewership anomalies. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 245–260, 2021.

[26] H. Milner, Y. Cheng, J. Zhan, H. Zhang, V. Sekar, J. Jiang, and I. Stoica. Raising the level of abstraction for time-state analytics with the timeline framework. In *CIDR*, 2023.

[27] G. Somashekar, A. Dutt, M. Adak, T. Lorido Botran, and A. Gandhi. Gamma: Graph neural network-based multi-bottleneck localization for microservices applications. In *Proceedings of the ACM Web Conference 2024*, pages 3085–3095, 2024.

[28] S. J. Taylor and B. Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[29] S. Yin, C. Fu, S. Zhao, K. Li, X. Sun, T. Xu, and E. Chen. A survey on multimodal large language models. *National Science Review*, 11(12), 2024.