

Engineering Resilient Reproducible Analytical Pipelines (RAP): Semantic-Based Self-Healing for High-Velocity Heterogeneous Data Streams

Tarek Clarke¹[0009–0005–7475–4766]

Tallinn University of Technology, Akadeemia tee 15a, 12619 Tallinn, Estonia

Abstract. Mission-critical telemetry systems such as clinical monitoring systems all face critical limitations in data availability, veracity and velocity. High-frequency data pipelines break easily when upstream schemas shift, sensors fail or interfaces change. Traditional pipelines rely on brittle selectors or rigid schemas. When these fail, organizations experience data blackouts, delayed decision-making and loss of situational awareness at critical points. This research proposes a self-healing Reproducible Analytical Pipeline (RAP) framework designed to autonomously mitigate schema drift without manual intervention in real time. Leveraging a container-deployed Python ecosystem and BERT Large Language Model Processing, the framework replaces static schema changes with a dynamic semantic embedding-driven reconciliation. The framework uses a cross-domain generalizable model to work in various industries. Our framework introduces a domain-agnostic ingestion interface that implements validation and normalization logic. The approach enables a unified, cross-domain resilient data ingestion, while reducing pipeline fragility and ensuring the stability of critical, high-velocity analytical workflows in mission-critical environments. To validate this framework, it has been evaluated in a motorsport telemetry environment that shows a critical amount of schema drift. It was evaluated across 9 different hardware environments to validate the technology independence of the system.

Keywords: Data engineering · Reproducible Analytical Pipelines (RAP) · Autonomous Agents · Data Provenance · Schema Drift · Self-Healing · BERT · Semantic Reconciliation · Telemetry

1 Introduction

High-velocity telemetry data streams often operate with ingestion pipelines where the upstream data schema evolves rapidly and can create instability in these pipelines. Since these pipelines are often brittle and schema-bound, even minor changes in the schema structure can break the pipeline, leading to "data blackouts". In high-stakes environments such as motorsport or ICU monitoring, data blackouts can have serious implications on the well-being of their subjects. This paper presents a framework capable of mitigating the impact of schema drift on

these environments and engineering a resilient, self-healing analytical pipeline that is domain agnostic and remains fully reproducible throughout its lifecycle. This paper presents the Resilient RAP (Reproducible Analytical Pipeline) a GPU-accelerated framework enhanced by BERT-based semantic schema reconciliation. This system was validated against AMD (ROCm/HIP), NVIDIA (CUDA), Apple Silicon and traditional x86 environments. Validated against synthetically generated data streams containing 3.6 million packets under a nominal 5% synthetic chaos profile (the selected GH200 reports recorded approximately 12% injected chaos), it maintained a 99.77% detection rate with a p95 latency as low as 0.004ms.

Applied to a Formula 1 telemetry simulation, the semantic layer achieves 94-95% accuracy, an improvement over traditional Levenshtein and regex reconciliation systems. The main contributions of this paper are as follows: a resilient, domain-agnostic analytical pipeline, GPU-accelerated BERT semantic reconciliation and validation across multiple GPU/CPU environments.

1.1 Related Work

This paper is based on the UK Government Digital Service’s definition of Reproducible Analytical Pipelines (RAP)[6], and Liu et al.’s definition of concept drift[3]. Thuel et al.’s paper on fault tolerance in real time systems[5] was used as inspiration for the application of the resilient aspect of the pipeline. For the chaos engineering portion, Basiri et al.’s paper on automating chaos experiments in production environments[1] was used as a foundation. For the semantic reconciliation portion, Zhang et. al’s paper on Schema matching using generative tags and hybrid features[7] and Klettke et al’s paper on schema extraction and structural outlier detection [2], and Reimers et al’s paper on BERT-based sentence embeddings [4] were applied to this framework.

1.2 Framework Overview

This framework employs a lightweight Bidirectional Encoder Representations from Transformers (BERT) reconciliation layer (all-MiniLM-L6-v2) that uses cosine-similarity scoring to translate drifted schema labels back to their canonical equivalents. BERT was selected over generative GPT-class models to minimize model size and latency, as the task requires semantic understanding rather than text generation. Across all reconciliation tests, the model achieved 62.5% accuracy, outperforming both Levenshtein distance (29.2%) and Regex-based pattern matching (41.7%) (categorized in Table 1). Notably, BERT solves 100% of namespace, synonym and unit drift, which are the categories that cause real-world pipeline failures. Although BERT introduces a modest latency overhead (~ 9.9 ms vs. 0.4 ms for Levenshtein and 0.06 ms for Regex), it completely eliminates the 70.8% data-loss floor inherent to character-distance approaches and demonstrates superior resilience to synonym drift (e.g., `gas_reserve_pct` $\rightarrow$

`fuel_reserve_percentage`). This ensures zero-loss sensor attribution in evolving telemetry environments and provides a robust foundation for cross-domain schema healing.

Table 1. Examples of Schema Drift and Reconciliation Results

Drifted Token	Expected Target	Category	BERTLev		Regex
velocity_kph	speed	Synonym	Y	N	Y
engine_revolutions	rpm	Synonym	Y	N	Y
disc_temperature	brake_temp	Synonym	Y	N	Y
driver_pulse_bpm	heart_rate	Synonym	Y	N	Y
combustion_heat	engine_temp	Synonym	Y	N	N
rubber_inflation	tyre_pressure	Synonym	Y	N	N
speed	speed	Typo	N	Y	N
rpm	rpm	Typo	N	Y	N
spid	speed	Typo	N	Y	N
r.p.m	rpm	Typo	N	Y	N
car.telemetry.speed_kph	speed	Namespace	Y	N	Y
telemetry.oil.temp	oil_temperature_c	Namespace	Y	N	N
moteur.regime.rpm	rpm	Namespace	Y	N	Y
sig_0x3F_lat	g_force_lateral	Obfuscated	N	Y	N
ch12_analog_psi	oil_pressure_psi	Obfuscated	Y	N	N
hr	heart_rate	Abbreviation	Y	N	N
vel	speed	Abbreviation	N	N	N
prs_tyr	tyre_pressure	Abbreviation	N	N	N
speed_mph	speed	Unit Variant	Y	Y	Y
brake_temp_kelvin	brake_temp	Unit Variant	Y	Y	Y
lateral_force_g	g_force_lateral	Reorder	Y	N	Y
front_brake_pressure	brake_pressure_front	Reorder	Y	N	Y
brk_frnt_psi	brake_pressure_front	Compound	N	N	N
max_v_ms	speed	Compound	N	N	N

2 System Architecture

2.1 Ingestion

Data is ingested from a synthetic dataset using pre-determined Formula 1 schema values and sensor ranges at 100Hz across all benchmarks. Telemetry streams from three specific race weekends were selected for consistency across benchmarks.

2.2 Chaos Injection

Chaos was injected into the telemetry stream before the reconciliation stage using the runner’s runtime profile controls. The public artifact does not include

a committed deterministic JSON profile or random seed for the historical runs, so a new execution is not expected to reproduce the committed summaries bit-for-bit. A unified test profile was outlined using 15 race events from the synthetic F1 race weekend semantics (Free Practice 1-3, Qualifying and Race over three different race weekends). For each event, 240,000 packets were assigned for a total of 3,600,000 packets at 100Hz ingress for each hardware benchmark. The profile was designed to assess the efficacy of the framework on high-volume data pipelines.

2.3 Hardware Environments

The framework was assessed on 9 different hardware platforms over various environments. Testing encompassed data centre-level GPUs (NVIDIA B200 Blackwell and H200 NVL), enterprise-level GPUs (NVIDIA RTX PRO 6000 Blackwell Workstation and Max-Q variants), high-end consumer GPUs (NVIDIA 5090 and AMD 7900XT), low-end legacy consumer GPUs (NVIDIA 1660 Ti), unified CPU/GPU architectures (Apple M4 and NVIDIA GH200), and consumer x86 CPUs (Intel i5-12600k).

Relevant platform specific parameters are systematically detailed in

Table 2. Hardware Evaluation Platform Specifications

Hardware Variant	Type	Memory / VRAM	Memory Bandwidth
NVIDIA B200	GPU	192 GB HBM3e	8.0 TB/s
NVIDIA GH200	GPU	96 GB HBM3 + 480 GB LPDDR5X	4.0 TB/s + 512 GB/s
NVIDIA H200 NVL	GPU	141 GB HBM3e	4.8 TB/s
NVIDIA RTX 5090	GPU	32 GB GDDR7	1,792 GB/s
NVIDIA RTX PRO 6000 Blackwell	GPU	96 GB GDDR7	1,792 GB/s
AMD RX 7900 XT	GPU	20 GB GDDR6	800 GB/s
NVIDIA GTX 1660 Ti	GPU	6 GB GDDR6	288 GB/s
Intel 12600K	CPU	32 GB (System RAM)	Shared Bus
Apple M4 (CPU)	CPU	16 GB Unified Memory	150 GB/s

2.4 Evaluation Metrics

To evaluate the resilience and performance stability of the self-healing RAP framework across heterogeneous hardware platforms, we define the quantitative metrics detailed operationally in Table 3:

Table 3. Metric Definitions

Metric	Formula	Interpretation
SDR	$\frac{\text{recovered}}{\text{schema_drift}}$	BERT schema drift recovery rate
RC	$\frac{\text{recovered}}{\text{total_anomalies}}$	Recovery coverage
ADR	$\frac{\text{total_anomalies}}{\text{total_chaos}}$	Anomaly detection rate
PAR	$\frac{\text{accepted}}{\text{total_packets}}$	Packet acceptance under chaos

Recovery Metrics Let n be the number of execution sessions within a standardized evaluation run. For each session i , let D_i represent the detected anomalies and R_i represent the successfully resolved anomalies. The *Recovery Coverage* (C_{rec}), which measures the operational fraction of autonomously healed drift events, is defined as:

$$C_{rec} = \frac{\sum_{i=1}^n R_i}{\sum_{i=1}^n D_i} \quad (1)$$

The corresponding *Recovery Deficit* (*Deficit*), representing total unmitigated anomalies that would otherwise induce a pipeline blackout, is computed via:

$$Deficit = \sum_{i=1}^n D_i - \sum_{i=1}^n R_i \quad (2)$$

The stability and performance consistency of C_{rec} across R discrete experimental runs per platform are evaluated using the sample standard deviation (σ) and the Coefficient of Variation ($CV = \frac{\sigma}{C_{rec}} \times 100\%$).

3 Benchmarks and Results

The Resilient RAP framework was verified across 9 platforms spanning 4 distinct Instruction Set Architectures (ISA). Testing evaluated the pipeline against 3.6 million telemetry packets under the nominal 5% cross-modal synthetic fault profile. Experimental outcomes detailing aggregate platform performance means are consolidated within Table 4. Complete per-run summary metrics from the selected reports are itemized systematically inside Table 5. While the GH200 had the lowest Packet Acceptance Rate (PAR), it was actually the best performing platform, as evidenced by its significantly higher recovery coverage (C_{rec}), calculated by equation 1, at 26.34% compared to the 15-17% range observed over all other platforms. Due to its unified CPU/GPU architecture, it is able to ingest larger amounts of data than a discrete GPU or CPU without throttling. Since other platforms experienced throttling, a higher ADR actually implies that packets are being dropped by the platform before the reconciliation stage, not that the platform is less accurate.

Table 4. Resilience Metrics Under 5% Chaos Injection (Mean Across 3 Runs)

Platform	Type	SDR	RC	ADR	PAR
NVIDIA GH200	GPU	1.0000	0.2634	0.5433	0.8979
NVIDIA RTX 5090	GPU	1.0000	0.1742	0.8233	0.9575
NVIDIA GTX 1660 Ti	GPU	1.0000	0.1740	0.8240	0.9575
NVIDIA RTX PRO 6000 Blackwell	GPU	1.0000	0.1733	0.8228	0.9575
NVIDIA H200 NVL	GPU	1.0000	0.1732	0.8249	0.9575
NVIDIA B200	GPU	1.0000	0.1728	0.8261	0.9576
AMD RX 7900 XT	GPU	1.0000	0.1723	0.8257	0.9575
Intel 12600K	CPU	1.0000	0.1511	0.9486	0.9576
Apple M4 (CPU)	CPU	1.0000	0.1509	0.9459	0.9564

Table 5. Per-Run Metrics (All Runs)

Platform	Run	SDR	RC	ADR	PAR
NVIDIA B200	Run 1	1.0000	0.1735	0.8261	0.9574
NVIDIA B200	Run 2	1.0000	0.1726	0.8254	0.9576
NVIDIA B200	Run 3	1.0000	0.1723	0.8268	0.9576
NVIDIA H200 NVL	Run 1	1.0000	0.1738	0.8228	0.9575
NVIDIA H200 NVL	Run 2	1.0000	0.1735	0.8262	0.9574
NVIDIA H200 NVL	Run 3	1.0000	0.1722	0.8255	0.9575
NVIDIA RTX 5090	Run 1	1.0000	0.1750	0.8240	0.9576
NVIDIA RTX 5090	Run 2	1.0000	0.1739	0.8202	0.9573
NVIDIA RTX 5090	Run 3	1.0000	0.1736	0.8258	0.9577
NVIDIA RTX PRO 6000 Blackwell	Run 1	1.0000	0.1723	0.8247	0.9574
NVIDIA RTX PRO 6000 Blackwell	Run 2	1.0000	0.1743	0.8225	0.9577
NVIDIA RTX PRO 6000 Blackwell	Run 3	1.0000	0.1733	0.8211	0.9574
AMD RX 7900 XT	Run 1	1.0000	0.1732	0.8221	0.9575
AMD RX 7900 XT	Run 2	1.0000	0.1718	0.8299	0.9576
AMD RX 7900 XT	Run 3	1.0000	0.1718	0.8252	0.9574
Intel 12600K (CPU)	Run 1	1.0000	0.1517	0.9479	0.9576
Intel 12600K (CPU)	Run 2	1.0000	0.1511	0.9507	0.9576
Intel 12600K (CPU)	Run 3	1.0000	0.1506	0.9473	0.9576
Apple M4 (CPU)	Run 1	1.0000	0.1506	0.9468	0.9575
Apple M4 (CPU)	Run 2	1.0000	0.1491	0.9476	0.9542
Apple M4 (CPU)	Run 3	1.0000	0.1531	0.9434	0.9576
NVIDIA GTX 1660 Ti	Run 1	1.0000	0.1727	0.8258	0.9577
NVIDIA GTX 1660 Ti	Run 2	1.0000	0.1755	0.8237	0.9575
NVIDIA GTX 1660 Ti	Run 3	1.0000	0.1737	0.8226	0.9574
NVIDIA GH200	Run 1	1.0000	0.2631	0.5437	0.8977
NVIDIA GH200	Run 2	1.0000	0.2630	0.5432	0.8978
NVIDIA GH200	Run 3	1.0000	0.2642	0.5430	0.8982

4 Limitations

Limitations of the framework include the application of a synthetic telemetry simulation over a live API. While live benchmarks have worked in practice, a full dataset has not been produced. These benchmarks require significant dependencies, including torch, sentence-transformers and ROCm/CUDA wheels. When running the framework for the first time on a hardware platform, there is significant startup time. When running the framework on AMD GPUs, there is degraded performance on Windows environments, with occasional fallback to CPU. AMD has 100% ROCm support in Linux environments. This framework also maximizes GPU/CPU usage, limiting the use of the hardware for unrelated processes concurrently.

5 Code and Data Policy

The code and synthetic data supporting this paper are available at the GitHub ngise artifact repository: github.com/tarek-clarke/resilient-rap-framework/tree/ngise

6 Future Work

There are multiple planned expansions on this framework. In order to assess the viability of the framework in a wider variety of packet events, we will train and assess an adversarial Large Language Model (LLM) for synthetic telemetry chaos compared to static JSON schemas. Leveraging dynamic Gemma 4-based (gemma4-e4b-it and gemma4-31b-it-GGUF) we will generate chaos scenarios with more aggressive parameters, such as variable merges, splits and a larger variety of semantic drift possibilities. Additionally, an assessment of Generative Pre-Trained (GPT) based LLMs of varying weights on the decoding stage of the semantic reconciliation will be conducted to validate the intersection of model weight, accuracy and latency. This assessment will be conducted on a variety of domains, including aerospace (SpaceX), financial (Finnhub) and meteorological (OpenMeteo) and a live F1 (OpenF1) telemetry streams. The use of smaller and quantized models will allow the framework to be run on a variety of hardware platforms while limiting latency and maximizing portability. Additionally, this framework will apply a tiered approach to reconciliation to reduce latency. BERT-reconciled drift events can be added to a known database of schema drift events to avoid future invocations of the model if it has already been done. By extension, a more robust GPT based model can be implemented in the event that BERT reconciliation has low confidence.

7 Conclusion

In conclusion, this framework demonstrates that BERT-based semantic translation can significantly improve telemetry streams in which semantic schema drift

can occur. With a 99.77% detection rate in 3.6 million packet runs, BERT is a viable solution for such situations. While current evaluations use synthetic data based on F1 telemetry semantics, future work includes the application of the framework to multiple telemetry streams across multiple domains, as well as an expansion of the application of the chaos engineering with the use of more advanced language models. By benchmarking over 9 distinct hardware platforms spanning ROCm, CUDA, Apple Silicon, x86 CPU, as well as consumer, data centre, and legacy consumer hardware, it has been established that this framework can be used in mission critical pipelines on a variety of environments and domains.

References

1. Basiri, A., Hochstein, L., Jones, N., Tucker, H.: Automating chaos experiments in production. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). pp. 1–10. IEEE (2019). <https://doi.org/10.1109/ICSE-SEIP.2019.00012>
2. Klettke, M., Störl, U., Scherzinger, S.: Schema extraction and structural outlier detection for json-based nosql data stores. In: Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services. pp. 81–90. ACM (2015)
3. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering* **31**(12), 2346–2363 (2019). <https://doi.org/10.1109/TKDE.2018.2876857>
4. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. pp. 3982–3992. ACL (2019)
5. Thuel, S.R., Strosnider, J.K.: Enhancing fault tolerance of real-time systems through time redundancy. In: Foundations of Dependable Computing: System Implementation, pp. 265–308. Springer (1994)
6. UK Government Digital Service: Reproducible analytical pipelines: A guide. <https://gss.civilservice.gov.uk/reproducible-analytical-pipelines/> (2020)
7. Zhang, Y., Mei, D., Luo, H., Xu, C., Tsai, R.T.H.: Smutf: Schema matching using generative tags and hybrid features. *Information Systems* **128**, 102570 (2025). <https://doi.org/10.1016/j.is.2025.102570>