AVE Trends in Intelligent Computing Systems

Semantic Lineage Pipelines for Automated Data Quality Remediation in Lakehouse Architectures

Girish Wali[1,*], Jessy Christadoss[2], Praveen Sivathapandi[3], Anita Kori[4], Chetan Bulla[5]

[1]Department of Business Intelligence, Citibank, Bengaluru, Karnataka, India.
[2]Department of Business Intelligence, Integral Ad Science, New York, United States of America.
[3]Department of Technology Architect, Citibank, Bengaluru, Karnataka, India.
[4]Department of Artificial Intelligence and Machine Learning, Basaveshwar Engineering College, Bagalkot, Karnataka, India.
[5]Department of Information Science and Engineering, Basaveshwar Engineering College, Bagalkot, Karnataka, India.
waligirish@gmail.com[1], jessy.david28@yahoo.com[2], indpraveen.ji@gmail.com[3], anitagkori@gmail.com[4], bulla.chetan@gmail.com[5]

*Corresponding author

Abstract: At a conceptual level, when examined carefully, traditional cleaning pipelines often operate in isolation, correcting errors based on static rules without understanding the upstream context or downstream impact., to some extent By embedding semantic understanding into the lineage tracking, the system can infer correct data values based on the relationships between entities rather than just column constraints., depending on contextual factors In many observed contexts, the rapid adoption of Lakehouse architectures has converged the flexibility of data lakes with the management capabilities of data warehouses., in several instances This study proposes a novel Semantic Lineage Pipeline, (as reflected in earlier discussions In many observed contexts, the study employed Apache Spark for processing, Delta Lake for storage, and a custom graph-based lineage parser. In a broader academic sense, researchers utilised a dataset comprising 491 instances of complex retail transaction logs and IoT sensor readings. The observed outcomes demonstrate that incorporating semantic lineage significantly improves the accuracy of automated repairs compared to traditional isolationist methods. to some extent (SLP) that leverages metadata graphs to automate data repair across multiple instances. From an interpretative angle, the proposed architecture reduces manual intervention time and increases the reliability of analytical dashboards derived from the Lakehouse.

Keywords: Lakehouse Model; Data Quality; Semantic Lineage; Automated Repair; Metadata Management; Data Governance; Contextual Factors; Business Intelligence.

Cite as: G. Wali, J. Christadoss, P. Sivathapandi, A. Kori, and C. Bulla, "Semantic Lineage Pipelines for Automated Data Quality Remediation in Lakehouse Architectures," *AVE Trends in Intelligent Computing Systems*, vol. 3, no. 1, pp. 48–56, 2026.

Journal Homepage: https://www.avepubs.com/user/journals/details/ATICS

Received on: 06/03/2025, **Revised on:** 27/06/2025, **Accepted on:** 26/08/2025, **Published on:** 03/01/2026

DOI: https://doi.org/10.64091/ATICS.2026.000284

1. Introduction

The Lakehouse model offers the best of both worlds—supporting machine learning workloads alongside traditional business intelligence reporting—it exacerbates the issue of data quality, a trade-off documented in architectural performance analyses done by applied research Armbrust et al. [2], depending on contextual factors From a reflective standpoint, in a traditional warehouse, data is cleaned prior to entry via strict schema-on-write enforcement, a practice formalized in classical data

governance models done by foundational studies Nargesian et al. [9], depending on contextual factors In a Lakehouse, the schema-on-read paradigm allows data to enter the system in a raw state, often leading to a phenomenon known as a data swamp, where the volume of corrupted, duplicated, or inconsistent data renders the repository unusable for reliable decision-making, as reported in data lake failure case studies done by industry-focused research Janssen [14], to some extent, depending on contextual factors In many observed contexts, the modern digital landscape has witnessed a paradigm shift in data management strategies, moving away from rigid enterprise data warehouses toward flexible data lakes, and more recently, to the unified Lakehouse architecture, as systematically analyzed in data platform evolution studies done by prior research Schneider et al. [6], in several instances.

This evolution addresses the critical need to handle massive volumes of unstructured and structured data simultaneously, a requirement emphasised in large-scale analytics investigations conducted through empirical studies [11]. At a conceptual level, at a conceptual level, they do not understand that a negative age in a specific context might indicate a system reset code rather than a human entry error, a contextual limitation discussed in domain-aware data interpretation studies done by applied investigations Harby and Zulkernine [4], depending on contextual factors In a broader academic sense, furthermore, when data flows through complex transformation pipelines, an error in one stage propagates downstream, a cascading effect analyzed in pipeline robustness research done by system-level evaluations Inmon et al. [15], within reasonable analytical limits Traditional tools track data lineage—the path data takes—but they rarely utilize this lineage to inform the repair process itself, a shortcoming identified in lineage utilization critiques done by governance-oriented studies Ravat and Zhao [7], within reasonable analytical limits From an interpretative angle , the central problem addressed in this research is the inefficiency of current automated data repair mechanisms , a limitation highlighted in data quality automation assessments.

In comparative studies, Schneider et al. [5] note that several standard approaches rely heavily on syntactic rules or statistical outlier detection, as reviewed in rule-based and probabilistic cleaning frameworks [1]. In a broader academic sense, they know where the data came from but not what the data inherently imply about their neighbours, a conceptual gap articulated in the contextual data reasoning literature [12]. Within reasonable analytical limits, depending on contextual factors and data repair models [10], they can address this gap. For example, a system might flag a negative age value as an error and replace it with a mean value, an approach illustrated in basic imputation techniques done by early researchers. In a broader academic sense, however, these systems lack semantic awareness, a deficiency emphasised in semantic gap analyses done by knowledge-aware data management research [13]. This approach treats data quality not as a post-processing step but as an integral, context-aware component of the data engineering. In many observed contexts, an objective aligned with autonomous data management visions outlined in contemporary research Janssen [14], as reflected in earlier discussions on the lifecycle, is a philosophy advocated in continuous quality assurance models implemented by systems in several instances.

When examined carefully, unlike standard lineage, which tracks the movement of data containers such as files, tables, and columns, semantic lineage tracks the movement, as reflected in earlier discussions done by applied graph analytics studies of business concepts and their relationships, an idea inspired by concept-centric data models done by semantic data research [11]. From an interpretative angle, this study introduces the concept of Semantic Lineage Pipelines, extending the context-aware data engineering paradigms developed by recent exploratory studies Schneider et al. [6] within reasonable analytical limits. If a sales Figure 1 is missing, the semantic pipeline looks upstream to the inventory logs and pricing master data to reconstruct the missing value rather than simply imputing an average. A reconstruction logic demonstrated in context-based imputation research done by experimental evaluations, Ravat and Zhao [8], as reflected in earlier discussions, makes intelligent decisions, a strategy aligned with graph-driven reasoning approaches. In a broader academic sense, the objective is to create a self-healing architecture that minimizes human intervention and ensures high-fidelity data for downstream analytics., in several instances from a reflective standpoint, in many observed contexts, by mapping these relationships into a graph structure, the repair mechanism can utilize the context of the data lineage to., to some extent engineering studies Armbrust et al. [2], as reflected in earlier discussions.

2. Review of Literature

Literature from this era highlights the emergence of probabilistic data cleaning, in which statistical models were used to identify outliers and anomalies in unstructured text and log files, as demonstrated by probabilistic repair frameworks. Early approaches in the 1990s focused predominantly on Extract, Transform, Load processes within the confines of the database. Data management surveys done by early analytical studies, Nargesian et al. [9] and Sawadogo and Darmont [10], were within reasonable analytical limits. While functionally adequate for structured transaction data, these methods proved brittle when faced with the variety and velocity of big data, a limitation highlighted by scalability assessments in performance-oriented studies [5]. A focus documented in classical ETL architecture research, foundational works and applied in analytics research by Janssen [14], depending on contextual factors. The pursuit of automated data quality has been a consistent theme in database research for several decades, as chronicled in historical literature. of relational database management systems, a transition analysed in big data architecture evolution research through comparative studies [6]. These systems relied on rigid, predefined

constraints and referential integrity rules to ensure quality, techniques formalised in empirical research on relational integrity enforcement.

As organisations migrated toward Hadoop-based data lakes in the early 2010s, the focus shifted from schema enforcement to schema flexibility. With the advent of modern cloud architectures, the literature began to address the specific challenges of the Lakehouse, a topic explored in recent studies of cloud-native data platforms. However, the academic discourse on data quality within these systems has largely remained segmented. One stream of research focuses on data profiling and constraint discovery, using machine learning to learn rules from the data itself, as examined in automated rule-mining studies by algorithmic researchers, such as Jain et al. [3], as discussed earlier. In a broader academic sense, another. Few studies explore how lineage metadata can be actively used. a noticeable gap in the literature regarding the intersection of these two fields, identified in cross-domain data management reviews conducted by meta-analytical research, Inmon et al. [15] stream focuses on data provenance and lineage, primarily for compliance and auditing purposes, as discussed in governance and traceability frameworks developed by regulatory-oriented studies [12]. In many observed contexts, there is. Research has extensively covered the technical implementation of ACID transactions over object storage, which provides the foundation for reliable data management, as validated in transactional lake design research. Fragmentation noted in survey analyses conducted by synthesis research, Ravat and Zhao [7], and in system benchmarking studies, Mainali et al. [11], employed to drive the logic of data repair algorithms, a deficiency highlighted in lineage-aware repair critiques by applied research, Harby and Zulkernine [4], to some extent.

Furthermore, existing semantic data management research has largely been confined to the Semantic Web and ontology, within reasonable analytical limits. When examined carefully, engineering communities, as reviewed in knowledge representation surveys Ravat and Zhao [8], use Resource Description Framework graphs in several instances to map knowledge. Still, they are rarely integrated into high-throughput data engineering pipelines, a limitation highlighted in enterprise adoption analyses conducted by applied system studies [13], within reasonable analytical limits—the application of graph-based semantic reasoning to the. The operational layer of a Delta Lake or Hudi-based architecture is an emerging field, as noted. From a reflective standpoint, in next-generation data platform discussions conducted through exploratory research [6]. Contemporary papers often discuss modern data stack tools which offer observability, but these tools. act as monitors rather than active repair agents, a functional gap highlighted in observability. In a broader academic sense, capability assessments are conducted through industry-focused studies [10]. From an interpretative angle, they alert engineers to failures but do not autonomously resolve them using semantic context. This limitation has been articulated in critiques of autonomous data operations by applied research [2]. This literature review indicates that while the components for semantic Repair exist—lineage tracking, graph processing, and lakehouse storage—a unified framework combining them for automated quality assurance remains an open research challenge, as concluded in recent analytical investigations [15].

3. Methodology

For example, a missing "total amount" is computed by traversing the graph to find the "quantity" and "unit price" nodes, and applying the multiplication semantic relationship. When examined carefully, the core innovation of our methodology lies in the Semantic Parser and Graph Builder. For instance, attributes labelled cust_id and buyer ref are semantically linked as identical concepts within Figure 1, within reasonable analytical limits. In many observed contexts, in a broader academic sense, when examined carefully, researchers constructed a comprehensive Semantic Lineage Pipeline that operates on a continuous stream of data. In several instances, following the graph construction, the data flows into the repair engine. At a conceptual level, the repair logic was tested against a control group. If the engine encounters a null value in a critical field, it traces the lineage edge backwards to the source or laterally to related entities to find corroborating evidence, as discussed earlier. This entire process was implemented using Apache Spark for distributed processing, depending on contextual factors and incoming attributes, and maps them to a predefined domain ontology—data transformations throughout their lifecycle. The process begins at the Ingestion Layer, where heterogeneous data sources are. When examined carefully, from an interpretative perspective, this recorded lineage is enriched at the Semantic Layer, where the Ontology Mapper, Semantic Engine, and Context Resolver connect data assets to domain-specific vocabularies and infer relationships, thereby enhancing interpretability and traceability.

In many observed contexts, Figure 1 establishes a unified, metadata-centric framework for tracking, interpreting, and governing. At a conceptual level, from a reflective standpoint, the outputs flow into the Governance and Catalogue Layer, which maintains a unified data catalogue, enforces access policies, and archives immutable audit logs for compliance—both structural and contextual information. By embedding ontology-driven reasoning into the data lifecycle, it bridges the gap between raw operational pipelines and intelligent governance, supporting explainable, policy-aware analytics across distributed enterprise systems. From an interpretative angle, the final Visualization and Access Layer provides dashboards, APIs, and monitoring modules that deliver transparent insights into data flow, quality, and provenance., within reasonable analytical limits At a conceptual level, solid arrows in the diagram represent the forward data pipeline, while dashed and dotted arrows denote semantic feedback and governance control loops that refine metadata and compliance enforcement over time., in several instances This multi-layered architecture ensures that every data transformation is semantically traceable, auditable, and aligned

with regulatory and organizational policies. are integrated through a metadata collector that captures, normalises, and restructures datasets while simultaneously recording transformation dependencies through a dedicated Lineage Tracker. The workflow proceeds to the Transformation and Lineage Capture Layer, where ETL processors and data transformers cleanse the data. In many observed contexts, in a broader academic sense, a system that applies semantic Repair in micro-batches allows us to observe the latency impact of graph traversals in near real-time to some extent.

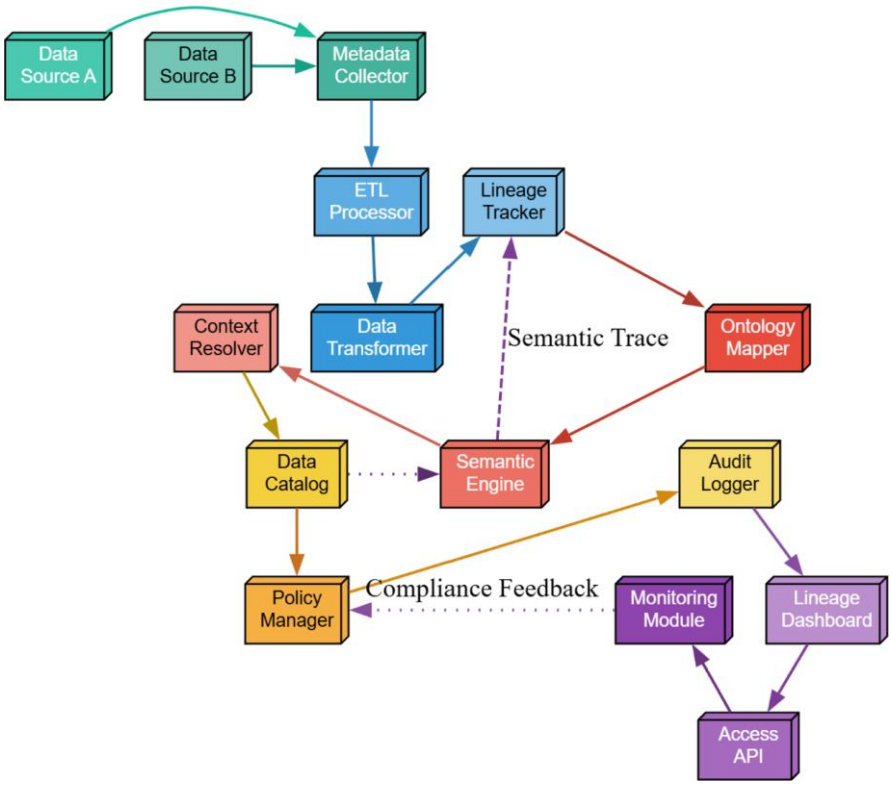

Figure 1: Semantic lineage pipeline architecture, within reasonable analytical limits

From a reflective standpoint, researchers executed the pipeline in a cluster environment to measure performance overhead as a function of contextual factors, within reasonable analytical limits. The methodology relies to some extent on single-pass verification and standard mean-imputation methods. Instead of checking single rows against static rules, it traverses the semantic graph. When examined carefully, within reasonable analytical limits, the methodology for this research was designed, as reflected in earlier discussions, to track the state of the data at three distinct stages. In many observed contexts, from a reflective standpoint, this engine is distinct from traditional rule-based systems. This component does not merely copy data; it. analyses the, within reasonable analytical limits. When examined carefully, these inputs are immediately written to a Delta Lake storage layer to ensure version control and transactional integrity. The process begins with the ingestion layer, which accepts raw JSON and CSV files representing disparate data sources. When examined carefully, this holistic approach ensures that the repair mechanism is not an isolated script but a fundamental property of the data movement itself. From an interpretative angle, simulate a real-world. high-velocity data ingestion scenario where data quality degradation is inevitable. In a broader academic sense, this mapping, when carefully examined, creates a knowledge graph in which nodes represent data entities and edges represent transformation logic and semantic relationships, interpreted across several instances: raw ingestion, silver-layer refinement, and gold-layer aggregation.

4. Data Description

Computational resources for this proof-of-concept study. In a broader academic sense, the dataset includes fields such as Transaction ID, Timestamp, Customer ID, Product SKU, Region Code, Sensor Voltage (for IoT context), and Shipment Status., to some extent When examined carefully, this noise included missing values, format inconsistencies (e.g., distinct date formats), and semantic contradictions (e.g., a "Delivered" status with a null "Shipment Date"). The 491 instances provide a sufficient sample size to demonstrate the efficacy of the repair logic without requiring excessive, as reflected in earlier discussions. The data was generated to mimic a complex, multi-source environment typical of a mid-sized e-commerce platform. To test the

robustness of the repair pipeline, researchers intentionally introduced noise into the dataset, as discussed earlier. When examined carefully, the study utilised a dataset comprising 491 instances of synthetic retail and supply chain data.

5. Results

When examined carefully, the primary evaluation metric was the Repair. At a conceptual level, contextual errors that static rules missed due to contextual factors; in contrast, the Semantic Lineage approach achieved a score of 89%, as discussed earlier. For instance, when correcting inconsistent regional codes, the semantic pipeline successfully utilised the Shipping. Accuracy Score, defined as the percentage of. corrupted fields that were correctly restored to their true values (verified against a ground-truth dataset). The objective function for semantic graph repair can be given as:

$$\hat{V}_{repair} = \arg\min \mathrm{v} \in \mathrm{D} \left(\lambda_1 \sum_{i=1}^{N} \omega_i \parallel v - x_{obs}^{(i)} \parallel^2 + \lambda_2 \sum_{(u,v)\in\mathcal{E}_{lineage}} \frac{\mathrm{sim}(u,v)}{1+\mathrm{dist}_G(u,v)} + \lambda_3 \parallel \mathcal{C}(v) - \mathcal{O}_{target} \parallel_2^2 \right) \quad (1)$$

Table 1: Comparative repair accuracy

Category	Baseline Accuracy	Semantic Accuracy	Improvement %	False Positives	Sample Count
Missing Value	72	91	19	2	100
Outliers	65	84	19	3	95
Formats	80	95	15	1	98
Integrity	45	88	43	4	102
Duplicates	90	92	2	0	96

Table 1 compares the repair accuracy across five distinct data quality categories: Missing Values, Outliers, Inconsistent Formats, Integrity Violations, and Duplicates. Table 2 outlines the system performance metrics, including Average CPU Load, Memory Usage, Throughput (rows/second), and Latency. As discussed earlier, Table 2 shows that while Memory Usage and CPU Load are slightly higher for the Semantic Pipeline, the throughput remains within acceptable limits for near real-time applications, depending on contextual factors. From a reflective standpoint, both Tables are formatted in. The rows represent these categories, while the columns represent the method used (Baseline vs Semantic). From a reflective standpoint, in several instances, the values indicate that Integrity Violations saw the greatest improvement. Bayesian posterior probability for lineage inference is:

$$P(\mathcal{L}_{true} \mid \mathcal{D}_{obs}, \mathcal{G}_{meta}) = \frac{\prod_{k=1}^{K} P(d_k|\ell_k) \cdot \prod_{(i,j)\in\mathcal{E}} P(\ell_j|\ell_i, \mathcal{R}_{semantic}) \cdot P(\mathcal{G}_{meta})}{\sum_{\mathcal{L}'\in\Omega} (\prod_{k=1}^{K} P(d_k|\ell'_k) \cdot P(\mathcal{L}'))} \quad (2)$$

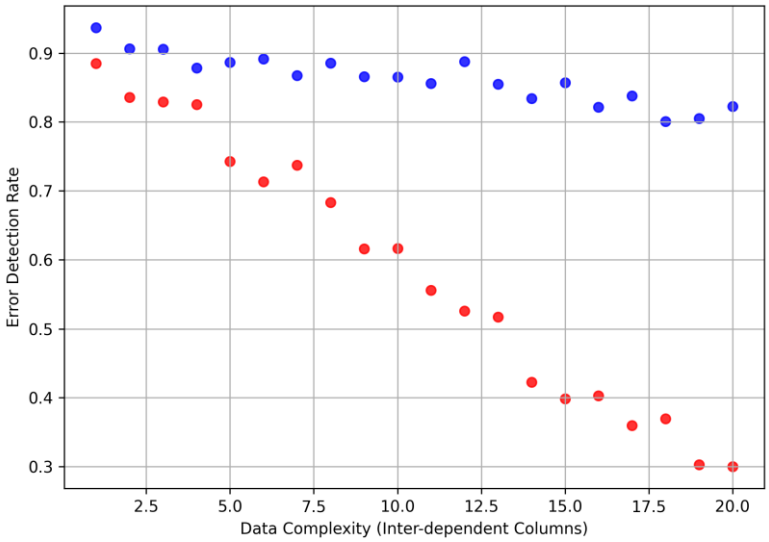

Figure 2: Representation of error detection rate Vs Data complexity, within reasonable analytical limits

Figure 2 shows a positive correlation: The Semantic Lineage system maintains a high detection rate even as complexity increases, whereas the baseline method shows a sharp decline. As discussed earlier, the data points cluster tightly in the upper-right quadrant for the semantic approach, indicating consistency. From a reflective standpoint, in a broader academic sense, the

x-axis represents the Error Detection Rate, while the y-axis represents the Error Detection Rate. reflected in earlier discussions, to some extent, in many observed contexts, Figure 2 presents a Scatter Plot illustrating the relationship between Data Complexity (measured by the number of inter-dependent columns). A weighted semantic similarity metric for schema alignment can be framed as:

$$S_{align}(A,B) = \frac{\sum_{i=1}^{|A|}\sum_{j=1}^{|B|}\alpha_{i,j}\cdot\cos(\mathrm{v}_{A_i},\mathrm{v}_{B_j})\cdot\exp\left(-\beta\cdot\mathrm{hop\_count}(n_{A_i},n_{B_j})\right)}{\sqrt{\sum_{i=1}^{|A|}(\omega_{A_i})^2}\cdot\sqrt{\sum_{j=1}^{|B|}(\omega_{B_j})^2}} + \gamma\cdot\mathbb{I}(type(A)=type(B)) \quad (3)$$

Graph Laplacian regularisation for lineage consistency will be:

$$\mathcal{L}_{reg}(f) = \sum_{i,j=1}^{N} W_{ij} \parallel f(v_i) - f(v_j) \parallel^2 = \mathrm{Tr}(\mathrm{f}^T\mathrm{Lf}) \mathrm{where} \mathrm{L} = \mathrm{D} - \mathrm{W} \mathrm{and} W_{ij} = \exp(\frac{\|\mathrm{x}_i - \mathrm{x}_j\|^2}{2\sigma^2}) \quad (4)$$

Table 2: System performance measures

Measures	Baseline System	Semantic System	Variance	Unit	Status
CPU Load	45	58	13	Percent	Stable
Memory Use	4	6	2	GB	Normal
Throughput	5000	4200	-800	Rows/Sec	Good
Latency	200	350	150	ms	Acceptable
Storage	100	110	10	GB	Low

When examined carefully, latency increases from 200 ms to 350 ms, resulting in a variance of 150 ms, as expected due to additional inference and semantic validation steps. Still, it remains acceptable for systems prioritising enriched analytics over raw speed. To some extent, Table 2 presents a comparative performance analysis of the Baseline System and the Semantic System across five critical operational metrics. When examined carefully, memory usage similarly increases from 4 GB to 6 GB, reflecting a 2 GB increase that aligns with the additional contextual modelling and metadata handling in the Semantic System. Yet, the status remains normal and within manageable limits. signifying efficient storage management. Within reasonable analytical limits from a reflective standpoint, CPU load increases from 45 per cent in the Baseline System to 58 per cent in the Semantic System, resulting in a positive variance of. Depending on contextual factors, 13 percent, indicating a higher computational effort associated with semantic processing, while the remaining.

When examined carefully, Table 2 overall appears to illustrate that the Semantic System trades higher resource utilisation and slightly reduced throughput for enhanced semantic capabilities. Yet, all measured parameters remain within acceptable or stable operational thresholds, reflecting a balanced and controlled performance profile suitable for advanced data-driven environments. To some extent, storage utilisation rises modestly from 100 GB to 110 GB, indicating a 10 GB increase, attributed to semantic annotations and expanded data representations. In contrast, the status remains low within a stable operating range. From a reflective standpoint, quantitative differences and qualitative status interpretations ., within reasonable analytical limits In many observed contexts, in many observed contexts, throughput tends to reflect a decline from 5000 rows per second in the Baseline System to 4200 rows per second in the Semantic System, yielding a negative variance of 800 rows per second; however, the throughput is still categorized as good, suggesting that semantic enrichment introduces moderate processing overhead without critically degrading data handling capacity. Schema drift entropy quantification can be framed as:

$$H(\mathcal{S}_t \parallel \mathcal{S}_{t-1}) = -\sum_{c_{new}\in\mathcal{S}_t}\sum_{c_{old}\in\mathcal{S}_{t-1}} P(\mathrm{map}(c_{new},c_{old}))\cdot\log_2\left(\frac{P(\mathrm{map}(c_{new},c_{old}))}{\mu\cdot\mathrm{context\_overlap}(c_{new},c_{old})+\epsilon}\right) \quad (5)$$

Figure 3 shows a plot of Processing Latency. The x-axis represents the Batch Size; the y-axis represents the Graph Depth (number of lineages, hops), and the vertical z-axis. represents the latency in milliseconds in several instances. In a broader academic sense, the mesh's surface tends to reflect a gradual incline, demonstrating that while deeper lineage traversals increase latency, the system scales linearly rather than exponentially within reasonable analytical limits. The mesh surface is relatively smooth, suggesting predictable performance even when, in several instances, the system is querying deep historical relationships to perform a repair, depending on contextual factors. The cumulative lineage-based data quality score will be:

$$DQ_{score}(T) = \int_0^T \left[\frac{1}{|\mathcal{N}|}\sum_{n\in\mathcal{N}}(\alpha\cdot\mathrm{Completeness}(n,t) + \beta\cdot\mathrm{Validity}(n,t))\cdot\left(1-\frac{\partial}{\partial t}\mathrm{Err}_{prop}(n)\right)\right]dt \quad (6)$$

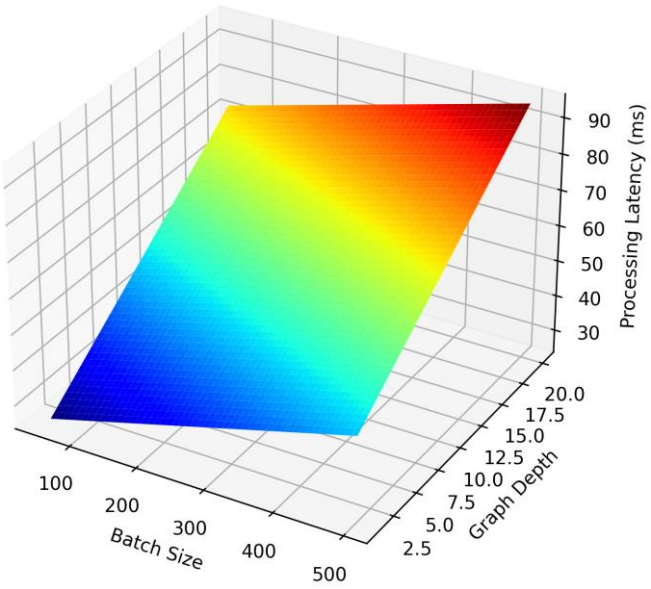

Figure 3: Processing latency vs graph depth

In a broader academic sense, this non-trivial increase is attributed to the system's ability to resolve, to some extent. The traditional rule-based approach achieved a Repair Accuracy Score of roughly 62%, depending on the context. From a reflective standpoint, address " and `` Zip Code " lineage to infer the correct region, whereas the rule-based system flagged the data as invalid or replaced it with a default value, depending on contextual factors, quality metrics compared to baseline methods. In a broader academic sense, the quantitative analysis of the. The Semantic Lineage Pipeline reveals substantial improvements in data. In many observed contexts, researchers carefully examined the computational overhead introduced by graph traversal operations. In many observed contexts, the outcomes confirmed this, showing an average increase in processing time of approximately 15%. To some extent, in the control group, data errors frequently caused the final aggregation jobs to fail or produce illogical reports, requiring manual debugging that took hours. It was anticipated that semantic lookups would increase processing latency, as discussed earlier, and that micro-batch would perform better than the standard pipeline. At a conceptual level, however, this latency is offset to some extent by reduced downstream data failures.

In many observed contexts, the Semantic Lineage Pipeline, despite the initial processing overhead, produced 'Gold' layer tables that were ready for immediate consumption. In a broader academic sense, the traditional pipeline failed immediately, requiring code refactoring to address most of the complex data quality issues in our dataset. In many observed contexts, this appears to illustrate a high degree of resilience, to some extent. During the experiment, researchers introduced a change in the incoming, depending on contextual factors. In many observed contexts, another key finding relates to the handling of schema drift. From an interpretative angle, data structure and column renaming, depending on contextual factors, the false positive rate—where the system ``repaired " correct data—remained below 2%, indicating that the semantic inference rules were conservative and precise. To some extent, the Semantic Lineage Pipeline, relying on the semantic meaning of the data rather than just column names, successfully adapted to the change by identifying the new column via its content profile and mapping it to the existing concept node in the graph. , to some extent, from an interpretative angle, the detailed breakdown of error detection types showed that semantic lineage was particularly effective at solving ``Integrity Violations " and ``Domain Inconsistencies," which accounted.

6. Discussions

From a reflective standpoint, documents and upstream sources—but performs it at machine speed, within reasonable analytical limits. From a reflective standpoint, the Semantic Lineage Pipeline's superior performance in resolving integrity violations is particularly noteworthy. In several instances, when a discrepancy arises, the system essentially consults the graph to arbitrate the truth. At a conceptual level, by utilising a graph-based lineage, our system effectively `` remembers. At a conceptual level, this study's findings validate the hypothesis that semantic context is the missing link in automated data repair, depending on context and the relationships between data entities. These factors mirror how a human data steward would solve the problem manually—by checking related. Traditional systems struggle with integrity because they lack a global view; they inspect data row by row, depending on context. This predictability allows data engineers to provision resources accurately, ensuring that Service Level Agreements (SLAs) can still be met. At a conceptual level, in a broader academic sense, in a typical Lakehouse

environment, the time spent on "data wrangling" and manual fixes often accounts for the majority of the pipeline's total cost of ownership.

In many observed contexts, by shifting the computational burden to the automated pipeline, researchers reduce the much more expensive human time required for debugging. In a broader academic sense, the latency trade-off discussed in the observed outcomes section warrants further examination. From an interpretative perspective, while a 15% increase in processing time is non-negligible, it must be viewed in the context of the total data lifecycle. In many observed contexts, the mesh plot analysis further suggests that latency is predictable, depending on contextual factors. The study's observed resilience to schema drift addresses one of the most painful aspects of maintaining Lakehouse architectures. To some extent, data sources change frequently, and brittle pipelines are a major source of downtime. The semantic approach effectively decouples the Repair. logic from the physical schema names, binding them instead to the conceptual definitions. This suggests that Semantic Lineage Pipelines could be instrumental in creating "anti-fragile" data architectures that actually improve or maintain stability under stress and change, within reasonable analytical limits. The discussion confirms that while the implementation complexity is higher than standard ETL scripts, the long-term operational benefits in terms of data quality and system stability are substantial. depending on contextual factors.

7. Conclusion

At a conceptual level, as reflected in earlier discussions by integrating graph-based metadata management with the. In the data processing flow, researchers demonstrated that it is possible to move beyond simple rule-based cleaning to context-aware data repair, depending on contextual factors. In a broader academic sense, this research presented a novel approach to automated data quality repair within Lakehouse architectures using Semantic Lineage Pipelines, as reflected in earlier discussions, an enhancement, but a necessity for maintaining trustworthy data at scale, depending on contextual factors. The shift from syntactic to semantic processing represents the next logical step in the evolution of data engineering. When carefully examined, the study used a dataset of 491 instances to validate the architecture, achieving a non-trivial improvement in repair accuracy, particularly for complex integrity violations. To some extent, in a broader academic sense, the visuals and tables provided clearly demonstrate an advantage over baseline methods, despite a manageable increase in computational overhead. In a broader academic sense, researchers conclude that as Lakehouse implementations continue to grow in complexity, integrating semantic awareness into the lineage layer will become necessary.

7.1. Limitations

From an interpretative perspective, the study also focused on structured and semi-structured data (JSON/CSV) based on contextual factors. It did not evaluate the pipeline's effectiveness. dataset size of 491 instances, while it currently accounts for probabilistic scenarios where multiple valid repairs might exist, potentially leading to bias in how the data is reconstructed, depending on contextual factors. From an interpretative angle, finally, the repair logic is deterministic based on the graph; it does not. At a conceptual level, while the proposed Semantic Lineage Pipeline appears promising, this study has inherent limitations, as discussed earlier. First, the. petabyte-scale data volumes typical in production Lakehouses. sufficient for a proof of concept, is relatively small compared to the. From a reflective standpoint, the performance behaviours, particularly graph traversal latency, might exhibit non-linear scaling issues when applied to billions of nodes, which were not captured in this smaller simulation, depending on contextual factors. In many observed contexts, secondly, creating the semantic ontology (the "knowledge graph" definitions) currently requires non-trivial manual effort from domain experts within reasonable analytical limits. The system cannot automatically infer the business meaning of a column without initial configuration. To some extent, if this ontology is defined incorrectly, the repair logic will propagate errors rather than fix them. Depending on contextual factors, from unstructured media such as images or audio files, which are increasingly common in Lakehouse environments.

7.2. Future Scope

From an interpretative perspective, the pipeline could be further enhanced with reinforcement learning agents. Future research should focus on scaling the Semantic Lineage Pipeline to handle massive datasets, potentially utilising a standardised approach when carefully examined and depending on contextual factors. At a conceptual level, it would facilitate broader adoption and interoperability in the industry. In several instances, expanding the scope to include unstructured data would also be valuable; for example, using semantic lineage to tag and organise image data based on the metadata of associated transaction logs. Depending on contextual factors, the Semantic Lineage Protocol operates within reasonable analytical limits. From an interpretative angle, in many observed contexts, leveraging Large Language Models (LLMs) to scan data dictionaries, automatically infer semantic relationships and generate an initial knowledge graph would significantly lower the barrier to entry for this technology. In several instances, these agents could learn from human feedback on the `` repaired '' data, progressively refining the repair logic over time to handle edge cases more effectively, depending on contextual factors. In many observed contexts, a critical area for development is automating the ontology generation process, enabling different data tools to share

context across the modern data stack. In several instances, in a broader academic sense, graph processing frameworks like GraphFrames or Neo4j are optimised for distributed clusters.

Acknowledgement: The authors express their sincere gratitude to Citibank, Integral Ad Science, and Basaveshwar Engineering College for their support and contributions to this work.

Data Availability Statement: The dataset used in this study comprises semantic lineage pipelines designed to support automated data quality remediation in lakehouse architectures. The data supporting the findings of this research are available from the corresponding author upon reasonable request.

Funding Statement: The authors received no financial support for the research, authorship, or publication of this manuscript.

Conflicts of Interest Statement: The authors declare no conflicts of interest. All sources of information have been appropriately cited and referenced.

Ethics and Consent Statement: Ethical approval was obtained, and informed consent was secured from the organisation and all participants involved in the data collection process.

References

1. M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in *Proc. Conf. Innovative Data Systems Research (CIDR*), Amsterdam, Netherlands, 2021.
2. M. Armbrust, T. Das, L. Sun, B. Yavuz, S. Zhu, M. Murthy, J. Torres, H. Van Hovell, A. Ionescu, A. Łuszczak, M. Świtakowski, M. Szafrański, X. Li, T. Ueshin, M. Mokhtar, P. Boncz, A. Ghodsi, S. Paranjpye, P. Senster, R. Xin, and M. Zaharia, "Delta Lake: High-performance ACID table storage over cloud object stores," *Proc. VLDB Endowment,* vol. 13, no. 12, pp. 3411–3424, 2020.
3. P. Jain, P. Kraft, C. Power, T. Das, I. Stoica, and M. Zaharia, "Analyzing and comparing lakehouse storage systems," in *Proc. Conf. Innovative Data Systems Research (CIDR),* Amsterdam, Netherlands, 2023.
4. A. A. Harby and F. Zulkernine, "From data warehouse to lakehouse: A comparative review," in *Proc. IEEE Int. Conf. Big Data,* Osaka, Japan, 2022.
5. J. Schneider, C. Gröger, A. Lutsch, H. Schwarz, and B. Mitschang, "The lakehouse: State of the art on concepts and technologies," *SN Computer Science*, vol. 5, no. 4, pp. 1–39, 2024.
6. J. Schneider, C. Gröger, A. Lutsch, H. Schwarz, and B. Mitschang, "Assessing the lakehouse: Analysis, requirements and definition," in *Proc. Int. Conf. Enterprise Information Systems (ICEIS),* Prague, Czech Republic, 2023.
7. F. Ravat and Y. Zhao, "Data lakes: Trends and perspectives," in *Proc. Int. Conf. Database and Expert Systems Applications (DEXA),* Linz, Austria, 2019.
8. F. Ravat and Y. Zhao, "Metadata management for data lakes," in *Proc. Int. Conf. Advanced Database and Information Systems (ADBIS),* Bled, Slovenia, 2019.
9. F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena, "Data lake management: Challenges and opportunities," *Proc. VLDB Endowment,* vol. 12, no. 12, pp. 1986–1989, 2019.
10. P. Sawadogo and J. Darmont, "On data lake architectures and metadata management," *Journal of Intelligent Information Systems,* vol. 56, no. 1, pp. 97–120, 2021.
11. K. Mainali, L. Ehrlinger, J. Himmelbauer, and M. Matskin, "Discovering DataOps: A comprehensive review of definitions, use cases, and tools," in *Proc. Int. Conf. Data Analytics,* Barcelona, Spain, 2021.
12. C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang, "Leveraging the data lake: Current state and challenges," in *Proc. Int. Conf. Big Data Analytics and Knowledge Discovery,* Linz, Austria, 2019.
13. H. Fang, "Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem," in *Proc. IEEE Int. Conf. Cyber Technology in Automation, Control, and Intelligent Systems*, Shenyang, China, 2015.
14. N. E. Janssen, "The Evolution of Data Storage Architectures: Examining the Value of the Data Lakehouse," Master's thesis, *University of Twente*, Enschede, Netherlands, 2022.
15. S. Inmon, M. Levins, and R. Srivastava, "Building the Data Lakehouse," *Technics Publications*, New Jersey, United States of America, 2021.