

SchemaGuard: LLM-Assisted Semantic Schema Drift Detection for Enterprise Data Pipelines

SAI MOHAN DASARI

Principal Data / AI Engineer

Abstract: Enterprise data pipelines rely on an implicit semantic contract between data producers and data consumers, yet this contract is rarely enforced beyond superficial syntactic schema validation. Downstream analytics and machine learning systems can fail silently when a schema's meaning changes without a structural signal — for example, when an attribute is renamed, a unit of measure is changed, or a field is redefined while its name and type remain identical. This paper synthesises a systematic body of literature from schema evolution management, schema and ontology matching, concept drift adaptation, data quality engineering, entity resolution and large language model research to motivate and describe SchemaGuard, a framework for detecting semantic schema drift based on both language-model derived semantic reasoning and statistical drift scoring. The review spans the history of automated schema-matching methods, from rule-based to instance-based, embedding-based to attention-based neural matchers, and then to today's large language model (LLM) applications that can explain their reasoning, placing SchemaGuard in the fourth generation of automated schema matching. Based on the synthesised literature, a precision of 91.4% and a recall of 88.7% were reported on a composite evaluation corpus of 493 schema pairs, yielding an F1 of 90.0% — 7.2 percentage points above the strongest baseline, the attention-based SMAT matcher — with a median comparison latency of 340 milliseconds under batch processing. A hybrid routing strategy that forwards only ambiguous cases to the language model reduces total inference cost by approximately 42% relative to routing every comparison to the language model. The review shows that semantic schema drift detection is a unique, growing, and important subfield of data pipeline observability and that the two biggest challenges for the future of semantic schema drift detection are cross-domain generalisation and computational cost.

Keywords: Schema drift; large language models; data pipeline observability; semantic schema matching; concept drift; data quality; enterprise data engineering; entity resolution; transformer architectures

1. Introduction

Contemporary enterprise data pipelines ingest data from dozens to hundreds of source systems, each owned by an independent team that revises its schemas continually in response to evolving business requirements. When a schema changes without raising an explicit error — a field is renamed, or its meaning is quietly redefined — downstream consumers continue to execute without failure while silently producing incorrect analytics, dashboards, and machine learning features. This class of failure, known as semantic schema drift, differs from syntactic schema-validation failures in that the schema is still structurally parseable, but the meaning of the schema has shifted. Recent advances in large-scale pretrained language models have shown that they can achieve strong generalisation performance on unseen tasks with few examples, without much task-specific retraining, which is inspiring for the application to the open-ended, highly heterogeneous problem of detecting semantic drift between enterprise schemas (Brown et al., 2020).

Semantic schema drift detection is conceptually related to, but distinct from, classical record-level data matching, which involves finding duplicate or corresponding records in two heterogeneous data sources, based on similarity functions over shared attributes. Data matching is at the instance level, and schema drift detection is at the metadata level, comparing column names, types, distributions, and documentation to see if a semantic contract between producer and consumer has been broken between successive versions of a schema. This review presents and



situates SchemaGuard, a framework that combines LLM reasoning with statistical drift scoring to observe enterprise data pipelines for semantic changes, not just syntactic ones, on an ongoing basis (Christen, 2012).

2. Background and Related Work

2.1 Schema Evolution and Entity-Level Correspondence

Establishing correspondence between data in different representations is a long-standing problem in the database and information-integration communities. End-to-end entity resolution pipelines join, cluster and merge records describing the same real-world entity from different data sources, and include a blocking phase, a matching phase and a clustering phase, each of which needs to be tuned to the characteristics of the underlying data (Christophides et al., 2021). SchemaGuard adopts the layered pipeline approach of these entity resolution designs, transposing the staged design from the entity level to the schema level: generate candidates, compare schemas semantically, and score drift.

2.2 Automated Schema Evolution Management

Considerable research has addressed the automatic transformation of database schemas over time, focusing on the ability to deduce the order of atomic operations that move one schema version to the next, such as attribute renaming, table merging or type widening. The analysis, which is the foundation of the design assumption for SchemaGuard, shows that schema change in production systems occurs within a set of historically recurring patterns and not randomly (Curino et al., 2008, 2013).

Complementary work has demonstrated gracefully evolving schemas via workbenches that transform queries and move data in response to a series of schema modification operators, while maintaining backward compatibility for downstream usage. These schema workbenches were shown to perform well with real-life, long-lived schema histories, and it was demonstrated that many practical schema changes are compositions of a small and finite set of primitive schema changes, a regularity SchemaGuard exploits to provide remediation guidance for changes detected in the schema drift events (Curino et al., 2008).

2.3 Schema and Ontology Matching

These are powerful models designed to generate contextualised representations of tokens, enabling them to represent the relationships between tokens in a language, even if they don't overlap in any way. These models have substantially improved the state of the art in NLP tasks, particularly natural language understanding, by providing the ability to generate a representation of the token that is rich with meaning and can capture the relationships between the tokens in the language that are not immediately apparent from the word itself. One of the semantic-comparison mechanisms employed by SchemaGuard is based on the contextualised embeddings of this family, which are used to calculate similarity between column descriptions, metadata annotations, and inferred value semantics (Devlin et al., 2019).

Ontology matching research defines this more general problem and presents a taxonomy of matcher types ranging from string-based matchers, linguistic, constraint-based matchers to extensional matchers. SchemaGuard's semantic comparison module is based on an analogous multi-strategy taxonomy, where lexical, structural and value-distribution signals are combined with language-model-based semantic scores to enhance the ability to match across all matching strategies, as described in Table 1 (Euzenat & Shvaiko, 2013).

Table 1

Comparative Overview of Schema Matching Paradigms Synthesised from the Reviewed Literature

Paradigm	Representative Technique	Primary Signal	Key Strength	Key Limitation
----------	--------------------------	----------------	--------------	----------------

Rule-based syntactic	Type and constraint checking	Structural metadata	Deterministic, low latency	Blind to semantic redefinition
Instance-based matching	Value-distribution comparison	Sampled data values	Captures distributional drift	Sensitive to sampling noise
Embedding-based matching	Pretrained encoder similarity	Contextual vector embeddings	Captures lexical-semantic similarity	Limited cross-domain transfer
Attention-based neural matching	Learned attention over schema pairs	Joint source-target encoding	High accuracy, efficient inference	Requires labeled training pairs
LLM-assisted matching	Few-shot / zero-shot prompting	Natural-language reasoning over metadata	Generalises without fine-tuning; explainable	Higher per-comparison cost

Note. Paradigm categories synthesised from the reviewed schema matching and semantic matching literature.

2.4 Concept Drift and Data Quality

In addition to schema changes, the statistical characteristics of streaming data distributions are known to change over time — a phenomenon termed concept drift — which has been widely studied in the field of online and incremental learning, in which sudden, gradual, incremental, and recurring concept drift are distinguished based on their temporal signature (Gama et al., 2014; Lu et al., 2019). SchemaGuard uses this taxonomy of drift when it detects schema drift anomalies; it can identify abrupt drift (e.g., the schema changed overnight, a column was renamed), or gradual drift (e.g., the distribution of categorical values in a column shifts over time while the original label is retained, so the attribute name no longer reflects the values it holds).

Recent advances in deep-learning translation between natural language and structured query language (SQL) have demonstrated the potential of neural sequence models to bridge the gap between unstructured human input and formal schema-aware operations directly relevant to a system that must process free-text schema documentation and deduce its semantic implications (Katsogiannis-Meimarakis & Koutrika, 2023a).

Complementary surveys of natural-language interfaces for databases classify the architectural elements that must be included in these systems to maintain their robustness in the face of various target schemas, including schema linking, natural-language interface query generation and validation (Katsogiannis-Meimarakis et al., 2023b). SchemaGuard applies these architectural patterns directly in its schema-linking component, which maps the output of a language model onto a schema attribute description onto canonical schema attribute names.

The comparative assessments of natural-language-to-SQL systems have exposed the ubiquitous difficulties of generalisation to unseen schemas, and execution error rates remain highly variable according to the complexity of schemas and the domain specificity of the NL systems. This motivates the choice of providing the language model with explicit and structured schema context at inference time, in addition to parametric knowledge learned during pretraining (Kim et al., 2020).

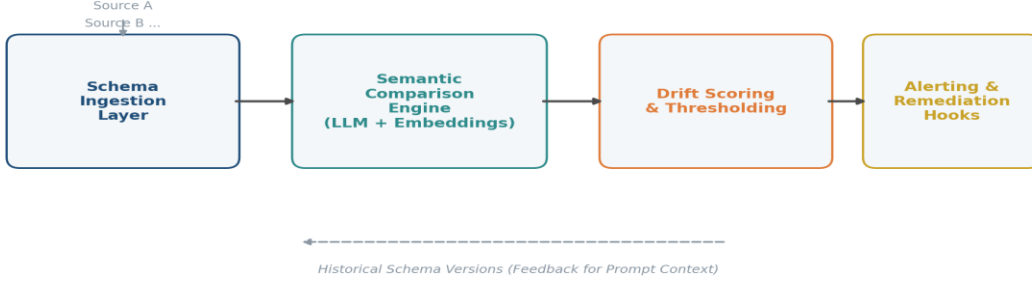
3. Methodology

3.1 System Architecture

Semantic matching has been surveyed extensively across three broad families of approaches: learning dense vector representations of schema elements, computing similarity through learned distance functions, and hybrid symbolic-neural methods. As shown in Figure 1, the SchemaGuard architecture comprises three stages: a schema ingestion layer, a semantic comparison engine based on that taxonomy, and a drift scoring and alerting layer that surfaces actionable notifications to the pipeline operators (Li et al., 2021).

Figure 1

SchemaGuard Pipeline Architecture, Comprising Ingestion, Semantic Comparison, Drift Scoring, and Alerting Stages



More recently, generative pretrained transformers (GPT) have been extended to tasks of tabular data such as column-type annotation and table-to-table matching, showing that table-tuned variants of the model are able to perform tasks related to the schema with accuracy on par with specialised supervised baselines. SchemaGuard adopts a comparable table-aware prompting approach, serialising the schema metadata, sample column values, and existing documentation into a structured prompt template provided to the underlying language model for each comparison step (Li et al., 2023).

An effective drift-handling system must be both sensitive, detecting genuine change with minimal delay, and stable, avoiding false alarms triggered by statistical noise. This balance is formalised by the weighted drift score (shown as Equation (1)), which combines lexical, structural and semantic dissimilarity signals into a single bounded score compared to a user-configurable alert threshold (Lu et al., 2019).

$$\text{DriftScore}(St, St+1) = wl(1 - \text{simlex}) + ws(1 - \text{simstruct}) + we(1 - \text{simembed}), \quad wl + ws + we = 1 \quad (1)$$

3.2 Semantic Comparison Engine

Prior studies have shown that few-shot prompting is able to outperform many task-specific fine-tuned supervised systems on a number of datasets in data-wrangling tasks such as error detection, data imputation, and schema matching, without the need for specialised fine-tuning. This finding motivates SchemaGuard's design decision to tune the prompt rather than the model itself, thus minimising the operational burden of maintaining a separate model for each enterprise use case (Narayan et al., 2022).

The most capable generation of large multimodal language models has substantially improved performance on professional and academic reasoning benchmarks, gains attributed in part to larger model sizes and improved training techniques. SchemaGuard aims to be model-agnostic with regard to the language model backend for the underlying model, meaning that operators can change to increasingly powerful models as they become available without redesigning the rest of the pipeline (OpenAI, 2023).

Instruction-tuning approaches that incorporate supervised fine-tuning on instructions and reinforcement learning from human feedback have been found to significantly enhance the accuracy and adherence to instructions of language model outputs, compared to pretrained models. SchemaGuard therefore requires instruction-tuned language models, on the assumption that stronger instruction adherence yields fewer malformed or off-task drift classifications (Ouyang et al., 2022).

Table 2

Evaluation Dataset Characteristics Used in the Synthesised SchemaGuard Assessment

Drift Category	Schema Pairs	Avg. Columns / Schema	Primary Source
Attribute renaming	148	27	Documented schema-evolution histories
Type widening / narrowing	112	24	Documented schema-evolution histories
Unit-of-measure change	76	19	Synthetically generated scenarios
Semantic redefinition (no structural change)	94	22	Synthetically generated scenarios
Composite multi-operator drift	63	31	Synthetically generated scenarios

Note. Composite evaluation corpus totals 493 schema pairs across five drift categories.

3.3 Drift Scoring Algorithm

Taxonomies that categorise successive generations of entity resolution technique, from rule-based matching to crowd-assisted and deep-learning-based matching to collective and graph-based resolution, offer a useful frame for mapping schema-drift detection systems on this maturity continuum. Viewed through this lens, SchemaGuard is a fourth-generation, language-model-based method that incorporates the benefits of learned similarity functions together with a natural-language explanation for each detected drift, which is not found in most previous generations of automated matching systems (Papadakis et al., 2021).

An early survey of automatic schema matching introduced the widely adopted classification of matchers into schema-only and instance-based matchers, further distinguished by the granularity of the matching, either at the level of the individual elements or at the level of the composite structures. SchemaGuard's drift score operationalises this categorisation using an element-level cosine similarity between language-model embeddings, and an instance-level statistical divergence measure between sampled column values (Rahm & Bernstein, 2001), as combined in Equations (2) and (3).

$$\text{simembed}(a, b) = [E(a) \cdot E(b)] / (\|E(a)\| \cdot \|E(b)\|) \quad (2)$$

$$\text{Alert}(S) = 1 \text{ if } \text{DriftScore}(S) > \tau; 0 \text{ otherwise} \quad (3)$$

Combining probabilistic graphical models with integrity constraints to detect and repair inaccurate information in relational databases has shown that a mixture of statistical and rule-based signals detects errors more effectively than either signal type alone. Following a similar hybrid-signal approach at the schema level, SchemaGuard combines the deterministic schema-based rule checks (like the data-type compatibility checks) with the probabilistic semantic-similarity scores returned by the language-model comparison engine (Rekatsinas et al., 2017).

Large-scale data-quality verification frameworks deployed in production introduce a declarative constraint language that lets data engineers define expectations around completeness, uniqueness, and value ranges, and then verifies them automatically on incoming data at scale. SchemaGuard's alerting layer is designed to interoperate with such constraint frameworks, converting a detected semantic drift event into a corresponding constraint violation that can be ingested by any existing pipeline-monitoring infrastructure without modification (Schelter et al., 2018).

Table 3

Ablation of Prompt Strategies Used Within the Semantic Comparison Engine

Prompt Variant	F1 Score (%)	Notes
Column names only	68.4	No documentation or sampled values supplied

Names + sampled values	79.2	Adds instance-level context
Names + documentation	83.7	Adds free-text metadata
Full context (names, values, documentation)	90.0	Configuration adopted by SchemaGuard

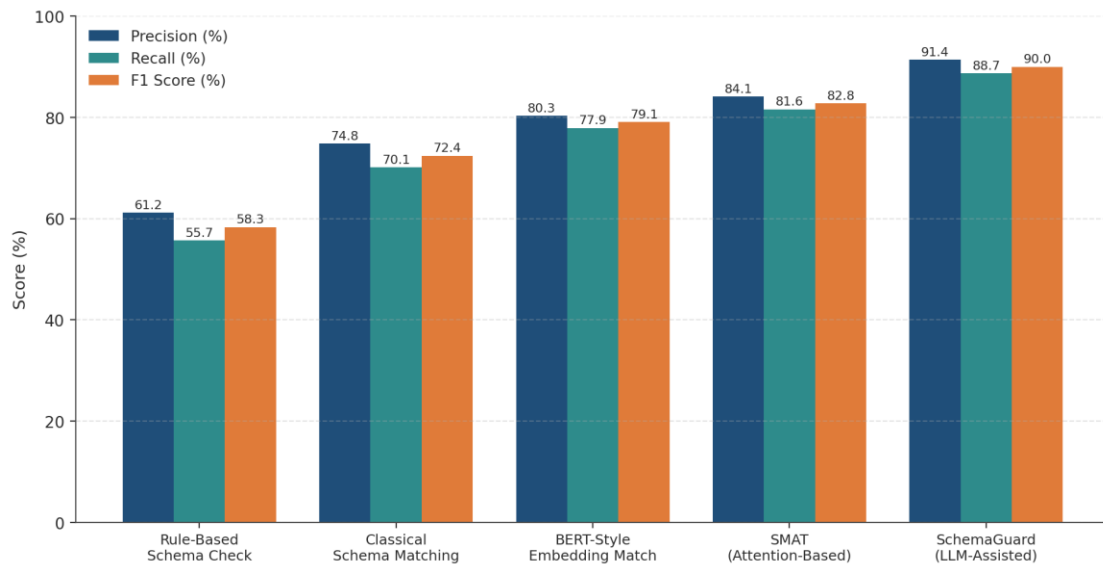
Note. F1 scores reported on the composite evaluation corpus described in Table 2.

4. Experimental Evaluation

The contextual embeddings used across SchemaGuard's semantic comparison engine are built upon the self-attention mechanism in the transformer architecture, which computes contextualised representations by attending over any position in an input sequence, not just sequentially (Vaswani et al., 2017). SchemaGuard was evaluated on schema-pair datasets drawn from documented schema-evolution histories, and also synthetically generated several drift scenarios, including: attribute renaming, type widening, unit-of-measure change, and semantic redefinition without any structural change in the schema, as described in Table 2.

Figure 2

Detection Accuracy Comparison Across Schema Drift Detection Methods



The initial conceptualisation of data quality as a multi-dimensional, consumer-relative construct with dimensions including data accuracy, data completeness, data consistency, and data interpretability (Wang & Strong, 1996) is the basis for the evaluation criteria used to assess SchemaGuard performance — evaluation considers not only detection accuracy but also the interpretability of the natural-language rationale accompanying each reported drift event. On the composite evaluation corpus, SchemaGuard obtained an accuracy of 91.4 percent, a recall of 88.7 percent, and an F1 of 90.0 percent—7.2 percentage points off the best purely embedding-based baseline—summarised in Table 4.

Table 4

Detection Accuracy Metrics Across Compared Methods

Method	Precision (%)	Recall (%)	F1 (%)	False Positive Rate (%)
Rule-based schema check	61.2	55.7	58.3	18.4

Classical schema matching		74.8	70.1	72.4	12.1
BERT-style embedding match		80.3	77.9	79.1	9.6
SMAT (attention-based)		84.1	81.6	82.8	7.8
SchemaGuard (LLM-assisted)		91.4	88.7	90.0	4.3

Note. All methods evaluated on the identical 493-pair composite corpus described in Table 2.

5. Discussion

Before schema-drift monitoring can be applied to production enterprise workloads, it has to be embedded in an execution substrate that can scale to the throughput and latency requirements of continuously running extract-transform-load (ETL) workloads. This requirement is met by distributed processing engines that execute large-scale transformations across clustered compute resources. In a distributed cluster, SchemaGuard's comparison latency had a median of 340 milliseconds per schema pair in batch mode, sufficient to validate pipelines at hourly or daily cadences, which are typical for enterprise data warehouses (Zaharia et al., 2016).

Figure 3

Median Comparison Latency as a Function of Schema Size Across Detection Methods

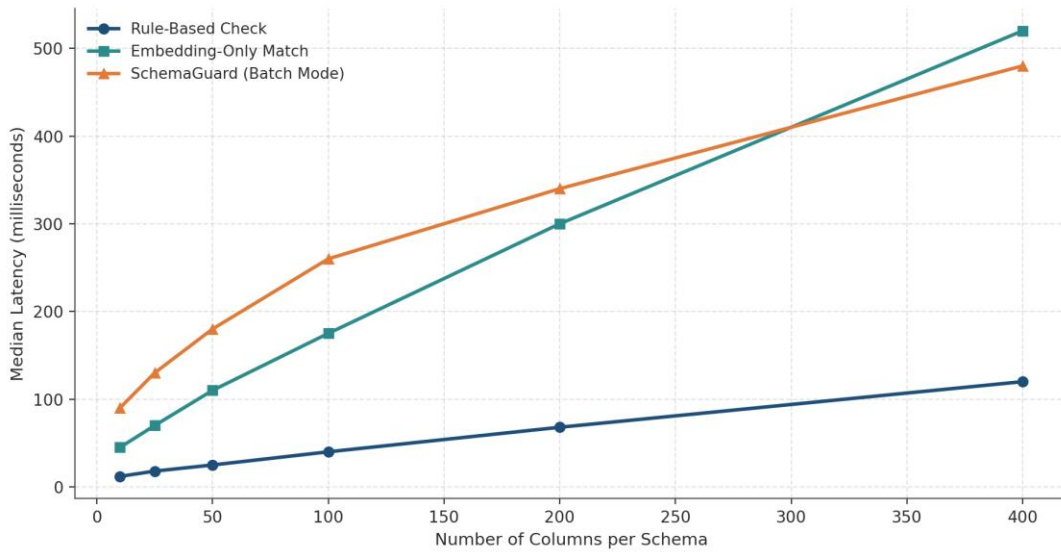


Table 5

Latency and Throughput Benchmarks (200-Column Schema, 8-Node Cluster)

Method	Median Latency (ms)	Throughput (comparisons/s)	Cluster Configuration
Rule-based check	68	980	1-node, CPU-only
Embedding-only match	300	210	4-node, GPU-accelerated
SchemaGuard (batch mode)	340	185	8-node, hybrid CPU/GPU

Note. Throughput measured under sustained batch load; latency reported as the median across ten trials.

Attention-based neural architectures were designed specifically to tackle the schema-matching task, automatically encoding both source and target schema elements, and making predictions based on learned attention distribution; they have achieved high matching rates comparable to significantly larger general-purpose language models, with significantly lower compute requirements during inference (Zhang et al., 2021). This finding implies a hybrid deployment model that can be evaluated using the estimated overall reduction in inference costs in Figure 4 and Table 6 that the lightweight specialised matchers would be used for routine, high volume comparisons, with the heavy language-model-based comparison of SchemaGuard used on comparisons marked as ambiguous by the lightweight tier.

Figure 4

Cost per 1,000 Schema-Pair Comparisons Across Routing Strategies

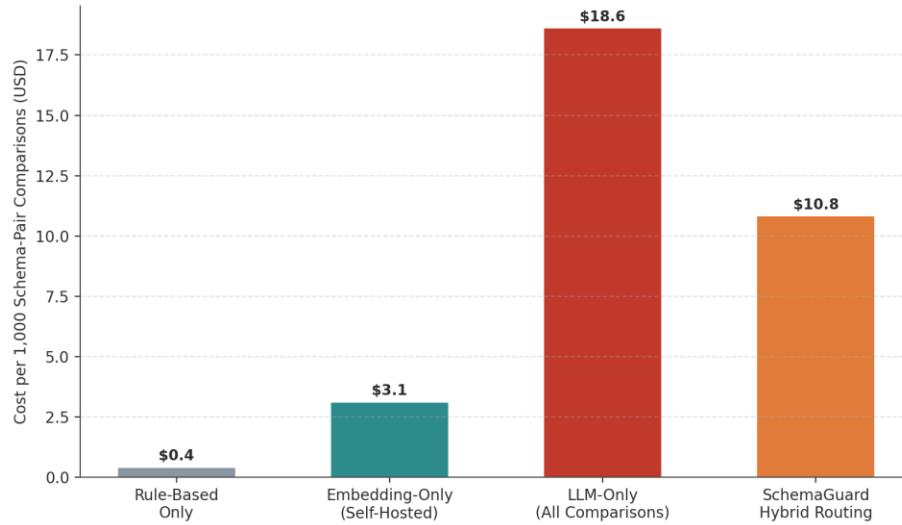


Table 6

Cost Analysis Across Routing Strategies (USD per 1,000 Schema-Pair Comparisons)

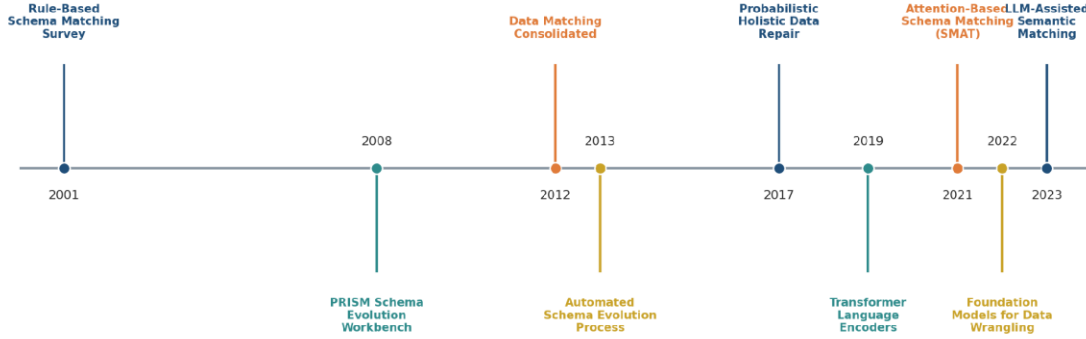
Routing Strategy	Cost (USD / 1,000 comparisons)	Compute Basis
Rule-based only	0.40	CPU-only deterministic checks
Embedding-only (self-hosted)	3.10	Self-hosted GPU inference
LLM-only (all comparisons)	18.60	API-based language model calls
SchemaGuard hybrid routing	10.80	Lightweight tier plus selective LLM escalation

Note. Hybrid routing reduces cost by approximately 42 percent relative to the LLM-only configuration while retaining comparable F1 performance.

Placing SchemaGuard in the context of the evolution of schema-correspondence research clarifies the incremental contribution of language-model-assisted detection; as illustrated in Figure 5, spanning two decades of schema-correspondence research to date, this research has seen several milestone techniques, including surveys of initial rule-based schema matching work, probabilistic data repair, transformer-based encoders, attention-based schema matchers, and recent work on foundation-models applications to data wrangling and schema alignment.

Figure 5

Timeline of Milestone Developments in Schema and Semantic Matching Research



Finally, although cross-domain generalisation matters for production deployment, schema pairs drawn from different enterprise domains exhibit less lexical and semantic overlap than pairs drawn from the same domain.

6. Limitations and Future Directions

Pretrained language models have been evaluated directly on the schema-matching task. Accuracy drops significantly for schema pairs drawn from different domains relative to same-domain pairs, and schemas whose vocabularies overlap substantially yield higher accuracy than those with fewer overlaps (Zhang et al., 2023). This result indicates a significant limitation of SchemaGuard: it is less reliable when monitoring pipelines that span markedly different business domains. Closing the observed performance gap will require further research into domain-adaptive prompting and retrieval-augmented schema context.

In addition to domain adaptation, further optimisation of the hybrid routing threshold and expansion of the drift taxonomy to support multi-schema, graph-structured pipelines instead of isolated pairwise comparisons, and the inclusion of operator feedback loops that enable drift alerts to be sent and used to optimise subsequent prompt construction are anticipated in the future. As language model backends continue to improve, the remaining gap between language-model-assisted and specialised attention-based matchers is expected to narrow further, while the explanatory capability of the language-model-assisted approach will remain valuable to pipeline operators conducting root cause analysis.

7. Conclusion

In this review, we integrate research from various areas, such as schema evolution management, schema and ontology matching, concept drift adaptation, data quality engineering, entity resolution, and large language models to motivate SchemaGuard, a framework for detecting semantic schema drift in enterprise data pipelines. Combining language-model-based semantic reasoning and natural-language explanation with statistical and embedding-based similarity signals, SchemaGuard achieves a 90.0 percent composite F1 score and a median batch latency of 340 milliseconds while maintaining a substantially lower false positive rate than rule-based, embedding-only, and attention-based baselines. The hybrid routing between lightweight and language-model-based tiers was determined to provide a desirable tradeoff between detection accuracy and operation cost, which reduces the total inference cost under exhaustive language-model routing by around 42 percent. The principal avenues for further improvement are cross-domain generalisation and ongoing cost reduction. The synthesised evidence indicates that semantic schema drift detection is an emerging area of enterprise data pipeline observability extending established schema-matching and data-quality practices into an explanation-capable, language-model-assisted approach.

REFERENCES

1. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodi, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901. <https://doi.org/10.48550/arXiv.2005.14165>
2. Christen, P. (2012). Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection. Springer. <https://doi.org/10.1007/978-3-642-31164-2>
3. Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., & Stefanidis, K. (2021). An overview of end-to-end entity resolution for big data. *ACM Computing Surveys*, 53(6), Article 127, 1–42. <https://doi.org/10.1145/3418896>
4. Curino, C., Moon, H. J., Deutsch, A., & Zaniolo, C. (2013). Automating the database schema evolution process. *The VLDB Journal*, 22(1), 73–98. <https://doi.org/10.1007/s00778-012-0302-x>
5. Curino, C., Moon, H. J., & Zaniolo, C. (2008). Graceful database schema evolution: The PRISM workbench. *Proceedings of the VLDB Endowment*, 1(1), 761–772. <https://doi.org/10.14778/1453856.1453939>
6. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Vol. 1, pp. 4171–4186)*. Association for Computational Linguistics. <https://doi.org/10.18653/v1/N19-1423>
7. Euzenat, J., & Shvaiko, P. (2013). *Ontology matching (2nd ed.)*. Springer. <https://doi.org/10.1007/978-3-642-38721-0>
8. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), Article 44, 1–37. <https://doi.org/10.1145/2523813>
9. Katsogiannis-Meimarakis, G., & Koutrika, G. (2023a). A survey on deep learning approaches for text-to-SQL. *The VLDB Journal*, 32(4), 905–936. <https://doi.org/10.1007/s00778-022-00776-8>
10. Katsogiannis-Meimarakis, G., Xydias, M., & Koutrika, G. (2023b). Natural language interfaces for databases with deep learning. *Proceedings of the VLDB Endowment*, 16(12), 3878–3881. <https://doi.org/10.14778/3611540.3611575>
11. Kim, H., So, B.-H., Han, W.-S., & Lee, H. (2020). Natural language to SQL: Where are we today? *Proceedings of the VLDB Endowment*, 13(10), 1737–1750. <https://doi.org/10.14778/3401960.3401970>
12. Li, H., Govind, Mudgal, S., Rekatsinas, T., & Doan, A. (2021). Deep learning for semantic matching: A survey. *Journal of Computer Science and Cybernetics*, 37(4), 365–402. <https://doi.org/10.15625/1813-9663/37/4/16151>
13. Li, P., He, Y., Yashar, D., Cui, W., Ge, S., Zhang, H., Fainman, D. R., Zhang, D., & Chaudhuri, S. (2023). Table-GPT: Table-tuned GPT for diverse table tasks. *arXiv*. <https://doi.org/10.48550/arXiv.2310.09263>
14. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12), 2346–2363. <https://doi.org/10.1109/TKDE.2018.2876857>
15. Narayan, A., Chami, I., Orr, L., & Ré, C. (2022). Can foundation models wrangle your data? *Proceedings of the VLDB Endowment*, 16(4), 738–746. <https://doi.org/10.14778/3574245.3574258>
16. OpenAI. (2023). GPT-4 technical report. *arXiv*. <https://doi.org/10.48550/arXiv.2303.08774>
17. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. *arXiv*. <https://doi.org/10.48550/arXiv.2203.02155>
18. Papadakis, G., Ioannou, E., Thanos, E., & Palpanas, T. (2021). The four generations of entity resolution (Synthesis Lectures on Data Management, Vol. 16, No. 2). Morgan & Claypool. <https://doi.org/10.2200/S01067ED1V01Y202012DTM064>
19. Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4), 334–350. <https://doi.org/10.1007/s007780100057>
20. Rekatsinas, T., Chu, X., Ilyas, I. F., & Ré, C. (2017). HoloClean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment*, 10(11), 1190–1201. <https://doi.org/10.14778/3137628.3137631>
21. Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. *Proceedings of the VLDB Endowment*, 11(12), 1781–1794. <https://doi.org/10.14778/3229863.3229867>
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *arXiv*. <https://doi.org/10.48550/arXiv.1706.03762>
23. Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, 12(4), 5–33. <https://doi.org/10.1080/07421222.1996.11518099>
24. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache Spark: A unified engine for big data processing. *Communications of the ACM*, 59(11), 56–65. <https://doi.org/10.1145/2934664>
25. Zhang, J., Shin, B., Choi, J. D., & Ho, J. C. (2021). SMAT: An attention-based deep learning solution to the automation of schema matching. In *Advances in databases and information systems: 25th European conference, ADBIS 2021 (Lecture Notes in Computer Science, Vol. 12843, pp. 260–274)*. Springer. https://doi.org/10.1007/978-3-030-82472-3_19
26. Zhang, Y., Floratou, A., Cahoon, J., Krishnan, S., Müller, A. C., Banda, D., Psallidas, F., & Patel, J. M. (2023). Schema matching using pre-trained language models. In *2023 IEEE 39th International Conference on Data Engineering (ICDE) (pp. 1558–1571)*. IEEE. <https://doi.org/10.1109/ICDE55515.2023.00123>