POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Data Quality Management in Knowledge Graphs

Tesi di Laurea Magistrale in
Computer Science and Engineering – Ingegneria Informatica

Author: **Yassine Mansour**

Student ID: 249217
Advisor: Prof. Cinzia Cappiello
Co-advisors:
Academic Year: 2024-25

Abstract

Knowledge graphs are extensively utilized for the representation and integration of heterogeneous data in contemporary information systems. Their practical utility is significantly reliant upon the quality of semantic data, which is frequently degraded by distorted data sources, automated ingestion processes, and shifting schema. Real-world RDF knowledge graphs regularly encounter semantic inconsistencies, incorrect literals, redundancy, and constraint drift, which transmit flaws to downstream applications.

This thesis examines the difficulty of maintaining semantic quality in dynamic RDF knowledge graphs through the introduction of a closed-loop governance structure. This framework smoothly combines SHACL-based validation, issues detection, data cleansing, confidence assessment, and automated constraint discovery into a unified pipeline. This approach deliberately reverses the induction process, in contrast to conventional open-loop techniques that either rely on manual specified static constraints or derive rules straight from raw data. Prioritizing semantic validation and repair ahead of constraint discovery ensures that the created constraints are based on a cleaned and semantically coherent graph rather than potentially noisy data.

Semantic coherence, temporal consistency, lexical robustness, and redundancy removal are the data quality dimensions that ae characterized by the framework in order to define semantic quality. Stress tests, drift situations, and recommendation tasks are used to evaluate the approach utilizing real IMDb data.

When compared to open-loop pipelines, the experimental results show that closed-loop semantic governance improves trustworthiness, boosts precision in recommendation outcomes, and strengthens robustness during data evolution. In brief, this study shows a scalable and long lasting approach to maintain semantic quality in growing RDF knowledge graphs.

Keywords: RDF Knowledge Graph Quality, Semantic Data Governance, SHACL Validation, Constraint Drift, Closed-Loop Data Cleaning

Abstract in lingua italiana

I knowledge graph sono ampiamente utilizzati per la rappresentazione e l'integrazione di dati eterogenei nei sistemi informativi contemporanei. La loro utilità pratica dipende in modo significativo dalla qualità dei dati semantici, che è frequentemente degradata da fonti di dati distorte, processi di ingestione automatizzati e schemi in evoluzione. I knowledge graph RDF del mondo reale incontrano regolarmente incoerenze semantiche, letterali errate, ridondanza e deriva dei vincoli, che trasmettono difetti alle applicazioni downstream.

Questa tesi esamina la difficoltà di mantenere la qualità semantica nei knowledge graph RDF dinamici attraverso l'introduzione di una struttura di governance a ciclo chiuso. Questo framework combina in modo fluido la validazione basata su SHACL, il rilevamento dei problemi, la pulizia dei dati, la valutazione della confidenza e la scoperta automatizzata dei vincoli in una pipeline unificata. Questo approccio inverte deliberatamente il processo di induzione, in contrasto con le tecniche open-loop convenzionali che si basano su vincoli statici specificati manualmente o derivano regole direttamente dai dati grezzi. Dare priorità alla validazione e alla riparazione semantica prima della scoperta dei vincoli garantisce che i vincoli generati siano basati su un grafo pulito e semanticamente coerente, piuttosto che su dati potenzialmente rumorosi.

La coerenza semantica, la consistenza temporale, la robustezza lessicale e la rimozione della ridondanza sono le dimensioni della qualità dei dati caratterizzate dal framework per definire la qualità semantica. L'approccio viene valutato utilizzando dati reali di IMDb mediante stress test, scenari di deriva e compiti di raccomandazione.

Rispetto alle pipeline open-loop, i risultati sperimentali mostrano che la governance semantica a ciclo chiuso migliora l'affidabilità, aumenta la precisione nei risultati di raccomandazione e rafforza la robustezza durante l'evoluzione dei dati. In sintesi, questo studio presenta un approccio scalabile e duraturo per mantenere la qualità semantica nei knowledge graph RDF in crescita.
Parole chiave: Qualità dei knowledge graph RDF, Governance semantica dei dati, Validazione SHACL, Deriva dei vincoli, Pulizia dei dati a ciclo chiuso

Contents

1 | Introduction

Knowledge graphs (KGs) are a widely adopted approach for organizing complex information. Knowledge graphs representing physical objects as entities and the relationships between them. Unlike traditional approaches that utilize arbitrary tables for storing data a knowledge graphs use a graph structure that facilitates the connection of entities even in the absence of shared properties. Their graph-based structure enables information originating from heterogeneous sources to be linked, enriched, and queried. This enables knowledge graphs to serve as a foundation for advanced systems that utilize data in both the public and business sectors. knowledge graphs are commonly expressed using RDF (Resource Description Framework) from a practical perspective since it offers a uniform way to describe entities, relations, and values in a machine-readable form.

However, the use of knowledge graphs is strongly related to the quality of the data rather than their expressive capabilities. Knowledge graphs are constructed through automated or semi-automated pipelines by integrating data from heterogeneous sources that are not properly maintained. This results in having data that exhibit quality flaws and implausible facts. Certain issues such as inaccurate claims, missing relationships, inconsistent role assignments, faulty literals, and redundant entities can be found in the data. Therefore, reliability and clarity can be compromised by these problems in KG systems, which also results in propagating error to the downstream processes that affects the trustworthiness and confidence of the results.

To address these challenges, prior research has introduced numerous frameworks aimed at characterizing and measuring the quality of knowledge graphs. Over the past few years, research on KG quality has largely concentrated on three core management processes:

First, **Quality Assessment** has been extensively studied and numerous frameworks have been constructed that define quality as fitness for use. It evaluates KG along predefined dimensions using weighted metrics. Even though those frameworks provide useful analytical insights yet there flexibility is limited since they usually rely on manually chosen variables and biased weighting structures. As a result, their operational significance and independence are limited.

Second, **Problem Discovery** techniques have been widely explored. Rule mining, statistical profiling, and embedding-based anomaly detection are some of these approaches that are used to look for incorrect or questionable triples. Although these techniques are good at highlighting possible problems but they lack practical assurances and semantic validity.

Third, **Quality Improvement** techniques such as knowledge graph completion or error correction and refinement had made significant progress in managing KG issues. These techniques utilizes embedding models and probabilistic reasoning but these solutions scalability and systematic governance over new evolving data since the absence of transparency.

A notable gap emerges at the intersection of these lines of work is the **Cleaning Phase** itself. Especially systematic error correction, literal handling, and semantic repair, remained underdeveloped. Existing approaches are often fragmented tool driven that focuses on isolated symptoms rather than understanding of data quality problems. In fact, there is an absence of a unifying taxonomies or pipelines that link a KG's flaws to an automated and systematic cleaning process.

This limitation is closely tied to the way data quality is conceptualized in prior work. Current KG quality taxonomies cover a wide range of dimensions such as accuracy, completeness, consistency, timeliness, accessibility, interoperability, trustworthiness, and provenance reflecting the diverse contexts in which KGs are used. Despite being extensive, these taxonomies often suffer from three structural weaknesses.

- First, many are tool driven rather than problem driven which are influenced by the features of the available assessment or validation tools. Rather than focusing on the intrinsic nature of quality issues.
- Second, the borders between dimensions are affected by taxonomic ambiguity which makes it challenging to develop detection and cleaning techniques.
- Third, the creation of generalizable quality management pipelines are all restricted by fragmentation between research.

Motivated by these observations, this work adopts a problem driven perspective on knowledge graph data quality centered on four core dimensions: accuracy, completeness, traceability, and interoperability. These dimensions recur across prior taxonomies but they are rarely addressed jointly or operationalized beyond assessment. Also, these dimensions are not treated solely as abstract evaluation criteria but as conceptual instruments for structuring cleaning and validation processes. Together, they capture complementary aspects of KG which includes the correctness of assertions and the ability to justify facts.

By grounding quality management in these dimensions, this work aims to bridge the gap between abstract quality assessment and scalable cleaning governance mechanisms.

Thesis Chapters Outline

Briefly, the thesis is structured as follows:

- In Chapter 2, a review for knowledge graphs semantic representation is explicitly stated to give a firm understanding about the capabilities and limitations. Also, comparing my proposed pipeline against variety of inductive logic paradigms.
- In Chapter 3, a closed-loop semantic quality assurance methodology is presented. In addition, to the theoretical details behind the intended architecture that covers SHACL validation, repair strategies, confidence assessment, and constraint evolution is stated.
- In Chapter 4, the proposed framework is applied to real data to illustrate its practical operation and outcomes.
- In Chapter 5, a series of stress tests and evaluation scenarios are used to demonstrate the framework's robustness and semantic drifts. In addition to actual downstream recommendation scenarios that visualizes the impact of the pipeline.
- In Chapter 6, the strengths and weaknesses of the framework are analyzed. Along with the design applicability and governance implications.
- In Chapter 7, a conclusion is drawn summarizing the key findings and outlining potential directions for future work.

2 State of the Art

2.1. Knowledge Graphs and Semantic Data Representation

Knowledge Graphs (KGs) became known as a leading framework for structuring, coordinating, and reasoning across diverse data sources [8]. Conventional data management systems rely on inflexible tabular structures. On the other hand, knowledge graphs represent data as networks of instances interconnected by well defined connections. This graph based format facilitates adaptable modeling of diverse domains. In fact, entities may vary in structure, properties, and origin while still maintaining semantic connections. Knowledge graphs are extensively utilized in areas like search engines, recommendation systems, and business data integration where diverse data sources require integration and logical searching.

Knowledge graphs fundamentally seek to encapsulate both data and its semantics. Semantic identifiers are used to label entities and relationships in order for the machines to analyze and utilize data across systems. Knowledge graphs are distinguished from basic graph databases by their semantic foundation. This gives KG the opportunity to address sophisticated features like explainable analytics, semantic querying, and inference. However, when knowledge graphs expand and change over time, their expressive potential and adaptability also present major issues with data consistency, integrity, and management [8].

2.1.1. Knowledge Graphs: Definition and Core Concepts

knowledge graph is a network structured model of knowledge where nodes represent concepts or tangible objects and edges show the semantic links between them. Because each graph component is individually recognized then an accurate referencing between datasets and applications is made possible. Knowledge graphs allow entities of the same type to have different attributes which shows the intrinsic variety of real world facts, in contrast to relational databases where schema strictness imposes homogeneity [4].

Since new entities and connections may be introduced without the necessitating of changing the global schema this allows KG to facilitate steady expansion and integration. In order to maintain the coherence and usability of the final graph the emphasis on semantic modeling and validation techniques needs to be taken in consideration. Knowledge graphs run the potential of becoming weakly linked and unreliable collections of facts in the absence of specific constraints or supervision [16].

2.1.2. RDF for Semantic Knowledge

The most widely used standard for describing knowledge graphs on the Semantic Web is the Resource Description Framework (RDF) [2]. RDF organizes data into triples, each of which has an object, a predicate, and a subject. The object defines a value or another entity, while the predicate indicates a property or connection, and finally the subject defines the entity that is being described. Any graph structure may be represented in a machine readable form due to this straightforward expressive paradigm.

To uniquely identify objects and connections across datasets the RDF uses Internationalized Resource Identifiers (IRIs). Actual values like strings, integers, and dates can be expressed via literals. RDF facilitates global data linkage and utilization by substituting IRIs for local identifiers that enables disparate datasets to refer to the same ideas without prior synchronization [2].

Schema flexibility is an essential characteristic of the RDF data paradigm. Although this flexibility offers significant benefits for development and integration, it also indicates that RDF alone does not enforce constraints on the accuracy, consistency, or completeness of data. To ensure data quality in subsequent downstream applications additional semantic layers and validation are required [12, 16].

2.1.3. Semantic Layers: RDFS, OWL, and Ontologies

Additional taxonomies like RDF Schema (RDFS) and the Web Ontology Language (OWL) are frequently used to give semantic structure to RDF [7]. RDFS includes fundamental modeling structures as classes, subclass hierarchies, domains, and ranges in order to facilitate efficient semantic modeling of entities and attributes. OWL broadens these capabilities by introducing more expressive expressions for specifying class equivalence and disjunction, property features, and logical constraints.

Ontologies created using RDFS and OWL act as common conceptual models which defines the intended meaning of data within a domain. They offer the basis for semantic interop-

erability and automated reasoning that empowers machines to deduce implicit facts from clearly expressed information [7, 16]. However, many complex knowledge graphs only use pieces of OWL expressiveness due to scalability limits and the challenge of reasoning over noisy data [16].

Furthermore, ontological semantics focuses more on reasoning than validation. They define what can be assumed, rather than what should be determined true. This difference is significant in data quality management, because discovering and fixing inaccurate data is sometimes more crucial than determining new facts [12].

2.1.4. Knowledge Graph Construction Pipelines

Knowledge graphs are often built using multiple phase pipelines that incorporate data from a variety of sources that is made of relational databases, CSV files, APIs, and unstructured text [4, 8]. The pipeline has the ability to ingest data and then transform them in addition to mapping data to the desired RDF representation. This method generates and aligns identifiers along with converting raw data into structured triples.

The development of pipelines frequently uses automated or semi-automated approaches to manage capacity and growth rate [16]. Automation allows for rapid graph growth, but it also poses dangers, since mistakes in triggered data or transformation logic can propagate directly into the knowledge graph. The prominent causes of semantic flaws that arise during construction are inconsistent integration, erroneous datatype transformations, and inconsistent identifiers.

2.1.5. Entity Resolution and Data Integration Challenges

One of the most difficult elements of knowledge graph development is entity resolution which is the process of recognizing whether various data objects belong to the same instances. In heterogeneous schemas, the same object may appear with distinct data types and values which can lead to data duplication, fragmentation, and misleading analytics.

However, extremely merging might mix up different instances that results in making semantic errors to be difficult to identify. These difficulties have a direct impact on downstream applications that rely heavily on entity based aggregation and similarity evaluations. As a result, mistakes in entity resolution are a key cause of quality erosion knowledge graphs [18, 19].

2.1.6. Heterogeneity and Noise in Knowledge Graphs

Nevertheless, the expressive capacity of RDF alone does not ensure data validity. Knowledge graphs exhibit several types of heterogeneity and noise which makes it a challenge to maintain data to be both syntactically valid and semantically consistent [8]. Structural heterogeneity occurs due to divergent modeling decisions among data sources, for example having varied property names or inconsistent levels of detail. Semantic heterogeneity arises when equivalent facts are perceived variably across various settings.

Noise occurs in many forms such as invalid literals, inaccurate dates, inconsistent role definitions, null values, and unnecessary repetitive triples. Due to RDF's lack of severe schema enforcement issues can arise. Anomalies of this kind might exist alongside acceptable data without any notice. Eventually, this accumulation of noise diminishes the graph's dependability and complicates pipeline cleaning attempts. In downstream applications like Wikidata and DBpedia structural configuration caused by schema leniency and diverse source might impact analytics quality and learning processes [5].

2.1.7. Evolution and Dynamics of Knowledge Graphs

Knowledge graphs have dynamic characteristics where the network structure changes when relationships shift, new entities are added, and domain knowledge advances. Schema evolution tends to be complex because attributes that were previously optional may become required also cardinality assumptions may fluctuate as data usage evolves.

This change results in a phenomena known as semantic or constraint drift, in which validation criteria and quality expectations diverge from the graph's actual framework and design. Static limitations set early on may no longer represent current data realities, leading in invalid violations or undetected mistakes. Managing this drift is a major difficulty in ongoing knowledge graph governance [9, 10].

2.1.8. Transparency and Provenance in Semantic Data Quality Management

An often overlooked aspect in current knowledge graph integrity and constraint management methodologies is the provenance of validation and correction choices. Although various structures emphasize on the detection of data infractions and the implementation of constraints. Yet there is relatively less focus on clarifying the rationale behind the classification of some data as invalid since the processes involved in repair and decision-making is not transparent. Also, the evidence that underpins the evolution of constraints over time is not explicitly [10].

Most data assessment tools including iterative pipelines and logic induction frameworks typically produce binary outcomes which classifies results as either "approved" or "unacceptable." Additionally, these frameworks frequently implement corrections through established protocols. Nevertheless, these frameworks frequently lack clear means for tracing the origin of choices throughout verification processes. So in developing or looped pipelines a constraints may be enhanced or removed without maintaining a clear log of the data breaches and repair actions that have led to the intended modifications.

In dynamic knowledge graphs, the lack of precise provenance makes it more challenging task to resolve semantic issues. It may become impossible to distinguish if the change in constraint is a result of noisy data injected to the pipeline or automated execution choices that took place over several iterations [16]. Therefore, governance processes run the danger of becoming opaque which would impair their understanding, transparency, and user confidence.

Transparency is crucial for effective problem solving, expert inspection, and accountability. In some cases in order to determine if a constraint or a repair complies with defined semantics a manual domain specialists are frequently needed yet in the absence of provenance information the evaluations rely more on human interpretation rather than on analytical proof. This lack emphasizes the need for semantic data quality frameworks that reveal the logic and background information of validation, correction, and constraint modification decisions [9, 10].

2.2. Knowledge Graph Quality Dimensions in Prior Work

Knowledge graph quality is commonly understood as a multidimensional concept. Previous work has established taxonomies that cover syntactic correctness, semantic validity, coverage, usability, and governance related concerns [5, 8]. Existing studies tend to emphasize subsets of these dimensions based on their specific objectives and application contexts rather than converging on a unified definition.

The vast majority of work focuses primarily on *accuracy* data quality dimension. Accuracy is typically assessed through triple correctness, accuracy of entity linking, and compliance with datatype and schema constraints. This focus is especially apparent in studies concerning knowledge graph development and fact verification since inaccurate statements directly impair reasoning and learning performance. Consequently, accuracy is often treated as the dominant indicator of knowledge graph quality.

Complementary studies incorporate *completeness* as a compulsory dimension. Indicating that even highly accurate graphs may be of limited practical value if they fail to cover relevant entities, properties, or relations. Completeness is regularly assessed using coverage ratios and missing value detection, especially in retrieval and analytical use cases. However, completeness is frequently examined in isolation and remains strongly dependent on the intended application.

More comprehensive quality frameworks introduce additional dimensions such as consistency, timeliness, traceability, interoperability, and redundancy with the aim of capturing operational and governance aspects of knowledge graphs. Even though these taxonomies provide broad conceptual coverage yet their heterogeneity and granularity often limit their direct applicability in automated validation or systematic pipelines.

Upon careful investigation of various frameworks a noticeable pattern shows up that the majority of techniques focus on correctness, completeness, or both, while other aspects are either implicitly addressed or regarded as supplementary. Synthesizing these perspectives, this work focuses on four core quality dimensions: *accuracy*, *consistency*, *completeness*, and *timeliness* which recur across existing taxonomies and jointly capture consistency, logical coherence, coverage, and temporal validity in RDF knowledge graphs.

2.3. Constraint Induction under the Raw Data Paradigm

Maintaining semantic integrity in RDF knowledge graphs is a complex challenge due to noisy data, incomplete information, and evolving graph structures. Traditional approaches have relied on static validation rules or rule-learning algorithms applied directly to the raw data, but these methods are often brittle and sensitive to inconsistencies. Open-loop systems learn patterns directly from the current state of the graph since there are no tools for iterative validation or correction. This can result in irregularities being misinterpreted as legitimate semantic patterns.

To address these challenges, a variety of methods have emerged which includes Inductive Logic Programming (ILP) for rule induction. These association rule mining frameworks like AMIE+, RDF2Rules, and SHACL-AF are shaped with strengths and limitations. Understanding these existing approaches and where they fall short in terms of noise resilience, adaptability, and governance is essential before exploring more advanced strategies for maintaining semantic quality in dynamic knowledge graphs.

Inductive Logic Programming and the Open-Loop Legacy

Classical ILP systems aim to induce logical rules from observed facts by relying on both positive and negative examples to refine hypotheses [1]. However, RDF knowledge graphs operate under the Open World Assumption (OWA) where the absence of a claim does not infer as an inaccuracy however it refers as incompleteness. This fundamental distinction complicates the direct application of traditional ILP techniques to Semantic Web data.

To address this challenge, a rule mining approaches such as AMIE (Association Rule Mining under Incomplete Evidence) and AMIE+ were developed to learn Horn rules from RDF graphs under inadequate and incomplete proofs [6]. AMIE+ introduced important optimizations strategies such as restricting rule bias and efficient pruning strategies that enables rule induction over large graphs such as YAGO and DBpedia [8].

Despite these advances, AMIE+ remains an open-loop system where it learns rules directly from the current state of the graph without any prior guarantee of semantic cleanliness to predict new facts. As a result, systematic errors and structural anomalies present in the data can be misinterpreted as valid logical patterns moving towards a phenomenon described as “mining the noise”.

Parallelization and Type-Aware Induction in RDF2Rules

The RDF2Rules framework was proposed to enhance the efficiency and accuracy of rule learning. The framework operates by parallelizing the process instead of learning one rule at a time. This approach mines frequent predicate cycles (FPCs) and generates multiple rules from these discovered patterns. A significant advantage of RDF2Rules over AMIE is its utilization of entity type information (via rdf:type predicates) during the generation and evaluation of rules which generally results in more accurate predictions [8].

However, RDF2Rules is vulnerable to errors in both actual data and conceptual representation since it expressly operates on raw A-box assertions and leaves out literals like strings or dates from its learning process.

Importantly, RDF2Rules performs rule induction as a one time pass analytical task just like AMIE+. The learnt rules cannot be verified, fixed, or reconstructed using a more stable representation of the graph because there is no feedback loop system. As a result, rule induction is still responsive to the quality of the data rather than imposing domain semantics [17].

SHACL and the Advancement of Graph Shaping

Integrity constraints on RDF data are now capable of being expressed formally due to the W3C's standardization of the Shapes Constraint Language (SHACL). [11]. Unlike RDFS and OWL that focus on inference and reasoning, SHACL is explicitly designed for validation. SHACL shapes define expectations regarding property cardinalities, datatypes, node kinds, and value ranges that enables systematic detection of structural and semantic violations.

Recent work on SHACL constraint induction that involved SHACLearner and other SHACL-AF (Advanced Features) methods have tried to automate the creation of these shapes through data [13, 15]. These methods use statistical features such as mean, variance, and the distribution of objects to classify constraints.

While these tools are invaluable for bootstrapping a schema (automatically making rules from existing data) for unconstrained graphs however they are typically applied as a one time induction step on raw data. When applied to noisy or weakly organized data then the induced shapes tend to encode existing irregularities as regular constraints. As noted in surveys of SHACL and RDF validation this phenomena leads to a weak validation models that are highly sensitive to data drift and dynamic ingestion practices [13].

Constraint Induction via Shape Extraction: QSE / Shactor (2025)

Recent work such as QSE / Shactor enhances constraint discovery by focusing on shape extraction from dynamic RDF graphs instead of relying on static schema inference. QSE offers incremental shape extraction procedures that continually update profile graphs and highlight structural changes over time compared to previous SHACL induction systems that work on static snapshots.

The system prioritizes statistical summaries and modeling as superior outputs to allow human operators to examine the evolution of class to property connections, datatype distributions, and property cardinalities. This makes QSE particularly effective for monitoring schema stability and identifying emergent patterns in dynamic knowledge graphs.

However, QSE remains largely descriptive rather than prescriptive. It extracts and visualizes shapes but it does not tightly integrate these shapes into a closed validation–repair–rediscovery loop. Constraint evolution is observed but not systematically enforced or governed through automated feedback loop into validation pipelines. As a result, QSE is best suited for exploratory schema analysis rather than continuous semantic quality enforcement [3].

Closed-Loop Constraint Governance: PRISM (2025)

PRISM's closed-loop constraint model offers a more comprehensive technique for dealing with constraints. PRISM incorporates constraint validation, auditing, and repair into a single lifecycle as opposed to shape extraction systems that handle constraints as inactivated artifacts.

PRISM operates over multi layered schemas that combines structural constraints, semantic rules, and policy level requirements. The validation results are explicitly fed into an audited repair process, where it detects violations and then initiates targeted remediation actions rather than simply reject. This enables PRISM to function as an active governance framework rather than a static validator.

Despite its advantages, PRISM requires the human specification of specific audit objectives and maintenance processes. Therefore, rather than being automatically deduced from data distributions the constraint evolution is directed by predetermined governance logic. Because of this, PRISM is more resilient in business or controlled settings but less able to adjust to the growth of natural schemas in poorly regulated knowledge graphs. [14].

2.4. Limitations of Open-Loop Quality Assessment Frameworks

While current research offers numerous ways for defining and measuring Knowledge graph quality. A large number of current frameworks employ an open-loop approach. In open-loop architectures the quality assessment and validation occur in independent phases. They are usually applied to a static data snapshot which leads to lacking systematic feedback into the governance process. These frameworks effectively uncover imperfections but provide limited assistance for sustained quality maintenance when knowledge graph are dynamic.

- Static Metric Dependency: Vast amount of quality frameworks rely on predetermined static metrics to measure qualities such as accuracy, completeness, and consistency, and these metrics are generally calculated at regular intervals and displayed as scores. Although these assessments offer important diagnostic information however they rarely result in automatic corrective actions.

  Furthermore, the choice and weighting of metrics are usually arbitrary and manual. This leads to expressing presumptions about what qualifies in a particular situation. These presumptions might not remain true as knowledge graphs develop as they become "semantic artifacts", resulting in metrics that are out of sync with the semantics and utilization of today's data.

- Lack of Integration Between Detection and Repair: The distinction between issue detection and issue resolution is a major drawback of open-loop systems. While deviations, mistakes, or suspicious patterns are detected by detection methods. The correction phase is left to human manual intervention. For substantial graphs, this gap restricts scalability and renders continuous quality enforcement.

  Detected problems often build up more quickly than they can be fixed, causing data quality to decline continuously. Assessment alone cannot prevent faults from spreading into downstream processes in the absence of a coordinated repair approach.

- Vulnerability to Noise and Data Drift: Open-loop systems frequently make the assumption that the empirical data offer a trustworthy foundation for assessment and modeling. However, noise produced during importation, aggregation, or extraction is often present in physical knowledge graphs. Noise can skew results when quality measurements or induced policies are calculated directly from raw data.

Static parameters and constraints becomes out of date when data distributions change over time as a result of constraint drift that leads to the causes of increase in false positives or false negatives. Therefore, open-loop systems lack the capability to adapt to these types of change and gradually diverge from the intended domain semantics.

- Rule Induction from Raw Data as a Structural Weakness: A number of designs attempt to automate quality assessment by directly deriving constraints or rules from raw data. Although this method seems promising in theory, it relies on the idea that common patterns translate into meaningful semantics. This assumption frequently breaks down in noisy graphs which leads to mistakes and contradictions being stored as acceptable rules.

  Rule induction has the danger of formalizing data degradation rather than reflecting desired domain restrictions in the absence of preceding cleaning or validation. Large scale knowledge graphs built from diverse and poorly structured sources are especially vulnerable to this issue.

- Absence of Governance and Feedback Loops: The lack of feedback mechanisms in open-loop systems is arguably their biggest drawback. Instead of being a component of an iterative governance cycle the quality assessment is handled as ending phase.

  This result having constraint refinement and schema evolution to be not consistently informed by validation. Deriving the difficulties for data, constraints, and domain semantics to stay aligned.

2.5. Positioning of the Proposed Framework within the State of the Art

In contrast to existing approaches, the framework proposed in this work provides a comprehensive and unified closed-loop solution. In order to maintain semantic quality assurance and constraint evolution in RDF knowledge graphs.

Unlike open-loop rule induction systems such as AMIE+ and RDF2Rules, which learn patterns directly from raw and potentially noisy data. The proposed pipeline operates on a SHACL-validated and semantically repaired graph to ensure that constraint discovery is grounded with highly reliable data. This mitigates the risk of misinterpreting irregularities or propagating errors as valid patterns.

Unlike shape extraction tools such as QSE / Shactor, which primarily observe and visualize how the graph changes without acting on the data, the proposed framework actively enforces and adapts constraints. This means the pipeline do not only checks and repairs the graph to ensure rules are followed (enforcement), but also updates the constraints automatically when the data evolves (adaptation) to sustain semantic integrity continuously rather than passively reporting patterns.

Furthermore, while governance-oriented frameworks like PRISM introduce closed-loop validation and repair mechanisms, they rely on manually defined audit policies and do not adapt constraints based on empirical data distributions and evolving graph patterns.

By combining automated post-cleaning constraint discovery, explicit drift detection, quantitative drift severity assessment (CDSI) this work bridges the gap between adaptability and governance.

Therefore, it provides a more reliable and scalable way to preserve long term semantic quality in dynamic graph settings by allowing knowledge graphs to develop gradually in accordance with both observable data patterns and domain integrity criteria. This approach offers a basis for more intelligent and autonomous RDF knowledge systems in addition to strengthening data quality assurance.

2.6. Benchmarking Against Existing Frameworks

These frameworks in Table 2.1 are reviewed as conceptual benchmarks rather than experimental baselines, due to differences in objectives, data assumptions, and availability.

Framework	Core Mechanism	Data Assumptions	Loop Type	Noise Handling
AMIE+	Association rule mining	Raw RDF triples	Open-loop	Threshold-based pruning
RDF2Rules	Frequent predicate cycles	Raw A-box data	Open-loop	Type-aware pruning
SHACL-AF	Statistical profiling	Raw / lightly filtered data	Static induction	Feature-based smoothing
QSE / Shactor (2025)	Incremental shape extraction	Evolving graph data	Observational loop	Visualization and statistics
PRISM (2025)	Integrated constraint model	Multi-layered schemas	Closed-loop	Audit-guided repair
Proposed Pipeline	Post-SHACL constraint discovery	Validated and repaired graph	Closed-loop	Systemic semantic cleansing

Table 2.1: Comparison of constraint and rule induction frameworks

3 | Methodology

Ensuring semantic data quality in the era of knowledge graph engineering is a foundational requirement and a persistent challenge, as knowledge graphs scale and integrate data from heterogeneous, continuously evolving sources. Eventually, maintaining semantic data quality becomes a central methodological concern rather than a one-time processing task. The network graph often suffers from semantic inconsistencies, lexical noise, temporal contradictions, and structural redundancy. Ensuring that graph data are not only syntactically valid but also semantically coherent is the central challenge in semantic data quality.

Semantic data quality refers to the degree to which data adheres to domain-specific meaning. In addition, to achieve optimal semantic outcomes, the knowledge graph must incorporate machine-interpretable constraints and logical coherence. Semantic quality goes beyond surface-level cleaning to enforce deeper rules about what values are allowed, how entities relate, and whether facts make sense in context. In RDF graphs, this includes verifying that names are readable, professions are valid, dates are realistic, and triples are non-redundant. Knowledge graphs become brittle without semantic quality, leading to misleading inferences, broken links, and unreliable analytics.

To address the semantic issues, the pipeline introduces an Automated Semantic Quality Assurance stage powered by SHACL (Shapes Constraint Language) a rule-driven stage that validates and repairs RDF data across the four Data Quality discovered dimensions: temporal consistency, semantic coherence, lexical robustness, redundancy detection.

For consistency and clarity, all methodology examples use the IMDb dataset to demonstrate the proposed semantic quality assurance approach.

3.1. Pipeline Architecture

The pipeline is modular and organized into five main components: **Data Loader & RDF Builder**, **Pre-and-Post SHACL Validation**, **Issue Detection Module**, **Cleaning Module**, and **Automated Constraint Discovery**.

At a high level, the goal of the pipeline is to ensure that a knowledge graph remains semantically consistent, structurally reliable, and adaptable to changes.

The pipeline conceptualizes semantic quality as an ongoing governance process instead of treating validation as a one time technical verification step. This design ensures that data quality is not only established at a single point in time but is systematically monitored, maintained, and improved as the graph evolves.

3.1.1. Process Pipeline Architecture

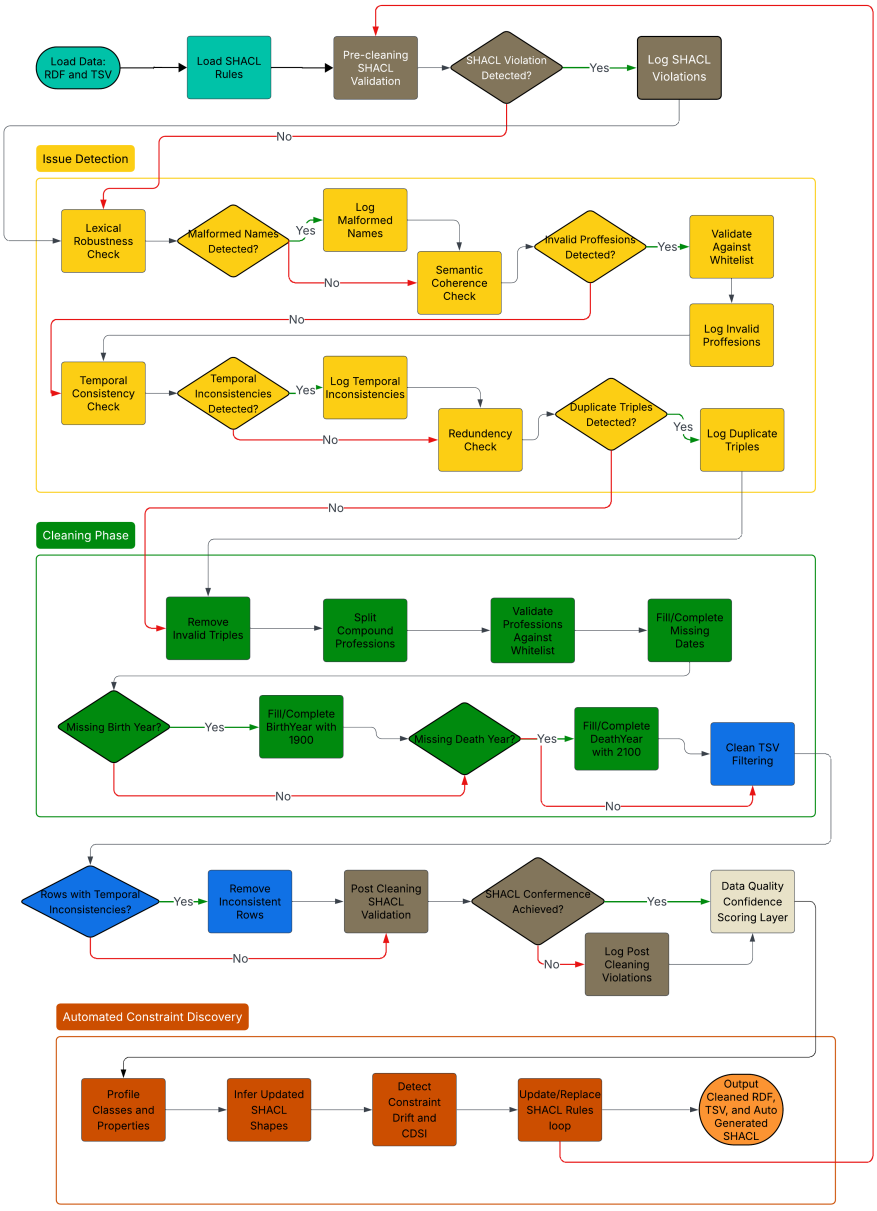

Figure 3.1: System architecture of the closed-loop KG cleaning pipeline.

Each component in the pipeline architecture has a clear task, where the loader ingests raw TSVs and produces RDF, then the SHACL validation diagnoses conformance before and after cleaning by having a detection module that classifies defects and the cleaning module applies deterministic provenance repairs or removals. Eventually the constraint discovery component profiles the graph to suggest updated shapes and detect drift.

3.2. SHACL: Shapes Constraint Language

The selection of SHACL as the foundational technology is justified by its formal design specialization in constraint validation for RDF data. SHACL provides a formal constraint language that precisely defines what constitutes as a valid data. Theoretically, SHACL focuses explicitly on checking data validity (validation) and distinguishes this function from inferring new facts (reasoning). This separation is crucial for high throughput quality assurance pipelines where constraint enforcement is the sole objective.

3.2.1. SHACL-Guided Semantic Data Normalization

Pre-Cleaning SHACL

The pipeline begins injecting data to a **pre-cleaning SHACL** validation, which uses a formal constraint language to quantify semantic violations in the RDF graph. These constraints define what constitutes valid data across multiple dimensions such as allowed profession values, required birth and death years, and datatype expectations. This initial validation provides a baseline for measuring semantic quality and guides the subsequent cleansing phases.

The pre-cleaning validation is intentionally non-destructive and serves three operational roles in the pipeline:

1. **Diagnostic baseline -** which quantifies the number and types of semantic defects present before any intervention to enable an objective comparison after cleaning.
2. **Remediation guide -** Each violation is mapped to a remediation policy (repair versus removal) according to the taxonomy and the pipeline's hybrid strategy.
3. **Provenance input -** Validation entries are recorded as provenance metadata so every subsequent repair can be traced back to a specific pre-validation finding.

Issue Detection Stage

The **issue detection stage** comes prior to the cleaning stage, where each detection dimension is chosen for its measurable impact on knowledge graph utility in particular for recommendation and inference tasks:

1. **Temporal Consistency**: time-related literals must be plausible and logically consistent across sources so that literal values are meaningful for temporal reasoning.

   Impact on analysis:

   - Misplaced entities on time-based graphs or timelines leads to incorrect age-based recommendations (for example, filtering for "actors born after 2000").
   - Broken temporal reasoning (for example, inferred facts that imply an entity acted before they were born).
   - Invalid time-based filters in recommendation systems (for example, queries for "actors under 30").

   Example. If a birth year is recorded as `1820` instead of `1980`, time-decay models or recency weighting may exclude or underweight the entity that results in producing incorrect recommendations.

2. **Semantic Coherence**: insures that logically contradictory or incompatible facts are not assigned to the same entity (for example, an entity labeled both `"actor"` and `"actress"`).

   Impact on analysis:

   - Entity disambiguation errors due to confused or conflicting role labels.
   - Incorrect ontological classification and broken hierarchies.
   - Confused role-based recommendations (for example, searches for similar screenwriters returning mixed results).
   - Segmentation errors in recommendation systems that rely on role categories (for example, treating `actor` and `actress` as interchangeable when they are used differently by downstream models).

   Example. If a model learns that "actresses are likely to co-star with X", mixing `actor` and `actress` labels can mislead the model and degrade recommendation.

3. **Lexical Robustness**: Maintain consistent formatting and token usage for labels (for example, `Film Director` vs `film_director` vs `Director (Film)`) so that profession and role literals are normalized and unambiguous.

   Impact on analysis:

   - Entity linking and grouping suffer when label variants are treated as distinct which reduces precision in profession based filtering.
   - Model training is degraded because embeddings for semantically identical labels diverge.
   - Search and recommendation quality declines when lookups fail due to inconsistent strings that can affect TF-IDF, string matching, and retrieval relevance.

   Example. Treating `film actor`, `actor`, and `tv actor` as different tokens prevents collaborative filtering from grouping similar entities.

4. **Redundancy Detection**: Detect duplicated entities or repeated metadata that share identical or near-identical identifying properties

   Impact on analysis:

   - Duplicates distort frequency counts and inflate popularity metrics.
   - Recommendation models are confused by split ratings or weights assigned to duplicate nodes.
   - Embeddings become polluted when the same person appears in multiple nodes with partial contexts to produce noisy representations.
   - In RDF knowledge graphs, duplicates break entity uniqueness and linkage that causes complicating joins and inference.

   Example. If three duplicate "Tom Hanks" nodes exist with partial metadata, a collaborative filtering engine may treat them as distinct actors and fail to consolidate signals.

Cleaning Stage

The **cleaning stage** is the core of the pipeline where targeted logic is applied to repair the RDF graph. This stage addresses four key semantic quality dimensions:

- **Lexical Robustness**: Ensures that literals like names are well-formed, readable, and free of noise.
- **Semantic Coherence**: Enforces domain-specific constraints, such as allowed profession values.
- **Temporal Consistency**: Validates chronological logic, ensuring birth and death years are realistic and ordered.
- **Redundancy Elimination**: Detects and removes duplicate or conflicting triples.

The cleaning process relies on deterministic rule-driven operations grounded in the detected violations and domain constraints. Repairs are applied only when the intended correction is unambiguous and semantically safe. In addition, the process prioritizes the preservation of the inherent semantics of the graph and avoiding modifications that could introduce artificial patterns or bias. Each decision whether to repair or remove a triple is guided by the principle of maintaining logical coherence and adherence to domain-specific rules to ensure that only well founded corrections are incorporated.

This conservative repair strategy positions the cleaning stage as a semantic normalization step rather than a data enrichment mechanism. It emphasizes structural and semantic integrity over maximizing data retention in order to produce a graph that is internally consistent. By enforcing these constraints early in the pipeline then the subsequent stages can operate on a foundation of high-quality and reliable data. Ultimately, this approach reduces error propagation and improves the trustworthiness of the knowledge graph for downstream applications.

The cleaning phase does not only serves as a repair task as it also establishes the framework's permissible knowledge boundaries. It makes the architecture's semantic assumptions to be clear and prohibits downstream components from processing instances that the framework isn't meant to handle. This is achieved by imposing domain constraints and removing statements that are outside of its scope. This function makes the limits of the framework clear without adding more inference or enrichment.

Post-Cleaning SHACL

Post-cleaning SHACL validation constitutes the final verification stage of the semantic quality assurance methodology and serves as an objective assessment. Its purpose is to confirm that all semantic violations identified during the initial validation phase have been effectively resolved through cleaning and repair operations.

Procedurally, this stage applies the same SHACL Shapes rules used during pre-cleaning validation to certify that evaluation criteria remain consistent across the pipeline. This guarantees that any observed improvements in conformance are attributable solely to the cleansing process, not to changes in the validation logic.

The validation assesses whether the repaired RDF graph satisfies all defined semantic constraints, including:

- Presence of mandatory properties
- Compliance with datatype requirements
- Adherence to domain-specific value restrictions
- Structural correctness of object relationships

By reapplying identical constraints, post-cleaning validation serves as a soundness check on the repair strategy to justify that no new inconsistencies were introduced and that all required semantic conditions are met. If the constraints are not resolved after the cleaning stage then it visualizes the resilience or challenges behind these violations.

This strengthens the semantic inspection approach and promise that the assessment of cleaning efficacy is entirely based on the original formulation of semantic rules. This final validation ensures that the RDF graph is logically consistent and no violations are remained.

From a methodological perspective, this stage also provides a binary conformance signal that certifies the readiness of the knowledge graph for downstream tasks. It also establishes a clean boundary between data preparation and data usage phases.

3.3. Confidence Scoring Layer

SHACL validation is effective at detecting and enforcing semantic constraints within RDF graphs, but it provides only a **binary conformance outcome** (conformant or non-conformant). This binary result does not capture differences in degree of *reliability* between entities that barely satisfy constraints and those that require extensive repair.

To address this limitation, a **data quality confidence scoring layer** is introduced as a methodological extension to SHACL validation. This layer quantifies the trustworthiness of individual entities by transforming qualitative validation and repair signals into a continuous numerical score.

The scoring methodology treats SHACL validation as a diagnostic process rather than a final judgment. Pre-validation violations, repair activity, and missing information are interpreted as indicators of uncertainty and data lineage risk. Entities that require extensive intervention to achieve compliance are assigned lower confidence scores, while entities that conform naturally retain higher confidence.

Methodological Interpretation

The resulting confidence score represents a probabilistic estimate of trust; however, it is not an absolute truth value. It enables:

- Differentiation between entities that are technically conformant but structurally fragile
- Ranking of entities by reliability
- Threshold-based data selection policies

This continuous scoring framework offers a more nuanced assessment of data quality than binary SHACL conformance and supports risk aware decisions making in downstream applications. In addition, the confidence scoring layer provides a systematic way to interpret uncertainty across the entire RDF graph.

3.4. Constraint Drift in Evolving Knowledge Graphs

Knowledge Graphs are dynamic, evolving structures that integrate heterogeneous data sources into semantically rich representations. Maintaining data quality becomes increasingly complex as the KG grows or changes. One of the most pressing challenges in this domain is constraint drifting, the phenomenon in which predefined validation rules, such as SHACL shapes, become outdated or misaligned with the graph's actual structure. This misalignment leads to false positives and missed violations, ultimately undermining the reliability of the validation process.

SHACL is used to enforce structural integrity in RDF graphs. However, SHACL shapes are typically handcrafted by domain experts and remain static over time. In large or frequently updated KGs, this manual approach is unsustainable. As the graph evolves, new properties emerge as patterns shift, and the original constraints become less relevant. Without continuous updates, SHACL validation risks enforcing obsolete rules that no longer reflect the true schema of the data.

To address this, a post-SHACL automated constraint discovery framework is proposed to ensure that the validation logic remains aligned with the graph's current state. This approach is grounded in the observation that, once SHACL validation and cleaning have been applied, the resulting graph provides a high-quality, trustworthy snapshot of the data. This validated graph should inform the next generation of constraints.

3.4.1. The Inflexibility of Static Constraints

The core problem this work addresses is the inflexibility and obsolescence of static SHACL shapes in dynamic knowledge graph environments. Specifically:

- SHACL shapes do not evolve automatically as the graph changes.
- Manual updates to constraints are labor-intensive and error-prone.
- Static constraints fail to capture emerging patterns, which initiates a constraint drift.
- Validation performance degrades as the graph grows under batch revalidation.

These limitations hinder the scalability, adaptability, and semantic accuracy of SHACL-based validation pipelines.

3.4.2. Automated Constraint Discovery via Statistical Profiling

After post-SHACL validation, the cleaned RDF graph is profiled to extract structural patterns. These include:

- **Type and Property Identification:** Detecting all classes and their associated properties.
- **Cardinality Analysis:** Inferring minimum and maximum occurrences of properties per class.
- **Datatype and Node Kind Detection:** Determining whether properties use literals or IRIs and their specific datatypes.

These insights are used to generate updated SHACL shapes that reflect the actual validated structure of the graph to effectively automate the ontology engineering process.

3.4.3. Constraint Drift in Knowledge Graphs

For instance, consider a SHACL shape that enforces the constraint that every imdb:Person must have at least one imdb:profession. As the knowledge graph evolves over time, new Person entities may be added without a profession, while existing entities might acquire multiple professions. In such scenarios, the original SHACL constraint (with minCount=1) can become misaligned with the actual data. This misalignment leads to false violations, where valid entities are incorrectly flagged as inconsistent, and missing violations, where genuinely inconsistent data goes undetected. These issues highlight a fundamental limitation of static validation rules as they are unable to adapt to the emerging patterns and structural changes within a dynamic knowledge graph.

As a result, relying solely on manually defined constraints can compromise semantic accuracy and reduce trust in the graph's integrity. To address these challenges, automated discovery and adaptive validation mechanisms are essential. By continuously profiling the graph and updating SHACL shapes to reflect its current state, it becomes possible to maintain both consistency and completeness to ensure that validation keeps pace with the graph's evolution.

3.5. Pipeline Feedback Loop

The finalized SHACL shapes produced by the automated discovery is fed back into the Pre-SHACL validation stage of the pipeline. This closes the loop and ensures that constraint validation evolves continuously alongside the knowledge graph.

3.5.1. Closed-Loop Governance Perspective

The novelty of this work does not reside in the introduction of a single constraint learning algorithm, but instead a systems level pipeline that re-conceptualize how semantic constraints should be validated and maintained in RDF knowledge graphs.

Traditional validation and induction systems operate in an open-loop where constraints are defined or learned once and then applied repeatedly to incoming data without systematic feedback from validation outcomes. In contrast, the closed-loop governance model introduced in this work treats the validated knowledge graph as an active source of feedback for constraint evolution.

Validation is not an end state, but an intermediate outcome whose results are used to reassess and refine the governing SHACL model. By periodically re-deriving constraints from the validated graph, the system guarantee that the constraint specification remains synchronized with both the intended semantics of the domain and the empirically observed high-quality structure of the data.

This feedback mechanism fundamentally changes the role of SHACL within the lifecycle of a knowledge graph. Rather than functioning as a static set of external rules, SHACL becomes a living governance artifact that adapts as the graph integrates new sources or undergoes structural change. Constraint updates are informed by evidence collected from a semantically consistent graph that allows the system to distinguish meaningful evolution from transient noise.

From a governance perspective, this closed-loop design supports long-term sustainability of semantic quality by reducing reliance on manual schema maintenance and mitigates the accumulation of obsolete or overly permissive constraints. Closed-loop also provides a principled approach for aligning validation logic with evolving data realities. Most importantly, it establishes a self-reinforcing cycle in which stronger constraints turns out to promote higher quality data.

4 | Implementation

4.1. Dataset Preparation

The implementation applies the proposed taxonomy to a real-world dataset. IMDb is a domain highly relevant to recommendation systems and, therefore, well-suited for knowledge graph modeling. The dataset is parsed into RDF format and evaluated using a Python-based pipeline for data validation, detection, cleaning, and automated constraint discovery.

The primary inputs for the implementation are the `name.basics.tsv` and `title.basics.tsv` files that contain biographical metadata for over 10 million individuals involved in the movie and television industry. The file includes attributes such as primary name, birth year, death year, primary profession, and known titles. To avoid memory bottlenecks when handling large-scale data, the loader processes the TSV file in a streaming fashion. It is implemented by reading line-by-line in order to build lightweight indexes only for the entities of interest.

This dataset serves as the foundation for constructing the initial RDF knowledge graph used throughout the study.

4.1.1. Sample Selection

A representative sample of 100 entities was selected for the initial experiments. Selection criteria were:

- presence of **primaryName**.
- presence of **KnownForTitles**.
- presence of **primaryProfession**.

This sample size enables rapid iteration while preserving realistic variability in labels and metadata.

4.1.2. Statistics

For the sample used in the thesis:

- **Total RDF triples** generated for the sample: 1 037.
- The sample includes well-known individuals to ensure the presence of realistic, recognizable metadata.

Although the sample size is small compared to the full IMDb dataset, it accurately reflects the structure and heterogeneity of the larger corpus and is therefore suitable for validating the RDF modeling approach.

4.2. RDF Construction

Each fact is represented as a subject, predicate, object triple (SPO), where the subject and object are entities, and the predicate denotes the relation between them.

The RDF model follows a compact URI scheme under the namespace `http://example.org/imdb/`. For each person, the following triple patterns are used:

- `<.../Name_With_Underscores> imdb:name "Primary Name".`
- `<.../Name_With_Underscores> imdb:birthYear "YYYY"8sd:gYear.`
- `<.../Name_With_Underscores> imdb:profession "comma,separated,labels".`
- `<.../Name_With_Underscores> imdb:knownFor <.../ttXXXXXXX>.`

RDF Parsing Results

The visualization in Figure 4.1 displays the RDF parsing results and a TSV sample used to build the RDF graph.

```
Requirement already satisfied: rdflib in /usr/local/lib/python3.12/dist-packages (7.2.1)
Requirement already satisfied: pyparsing<4,>=2.1.0 in /usr/local/lib/python3.12/dist-packages (from rdflib) (3.2.5)
Total RDF triples: 1037
Triple: http://example.org/imdb/tt0054698 http://www.w3.org/2000/01/rdf-schema#label Breakfast at Tiffany's
Triple: http://example.org/imdb/Randolph_Scott http://example.org/imdb/profession actor,miscellaneous,producer
Triple: http://example.org/imdb/Paul_Newman http://example.org/imdb/knownFor http://example.org/imdb/tt0084855
Triple: http://example.org/imdb/Gérard_Pirès http://example.org/imdb/name Gérard Pirès
Triple: http://example.org/imdb/tt0040506 http://www.w3.org/2000/01/rdf-schema#label Key Largo

TSV sample:
      nconst      primaryName birthYear deathYear  \
0  nm0000001     Fred Astaire      1899      1987
1  nm0000002    Lauren Bacall      1924      2014
2  nm0000003  Brigitte Bardot      1934        \N
3  nm0000004     John Belushi      1949      1982
4  nm0000005   Ingmar Bergman      1918      2007

                  primaryProfession                          knownForTitles
0       actor,miscellaneous,producer  tt0072308,tt0050419,tt0027125,tt0025164
1  actress,soundtrack,archive_footage  tt0037382,tt0075213,tt0038355,tt0117057
2   actress,music_department,producer  tt0057345,tt0049189,tt0056404,tt0054452
3      actor,writer,music_department  tt0072562,tt0077975,tt0080455,tt0078723
4             writer,director,actor  tt0050986,tt0069467,tt0050976,tt0083922
```

Figure 4.1: Terminal RDF parsing output.

The RDF section highlights how individual entities are converted into triples using their IMDb URIs and associated labels. While, the TSV portion illustrates the raw tabular structure from which these triples are derived. Together, these elements demonstrate the transformation from flat IMDb records into a structured RDF graph.

4.2.1. Knowledge Graph Visualization

The visualization in Figure 4.2 displays a sample of the constructed IMDb knowledge graph: nodes represent persons and film entities, and edges encode relationships such as `imdb:knownFor` and `imdb:profession`. The figure shows the graph topology produced by the ingestion step and illustrates how multi-value fields are expanded into separate triples for downstream processing.

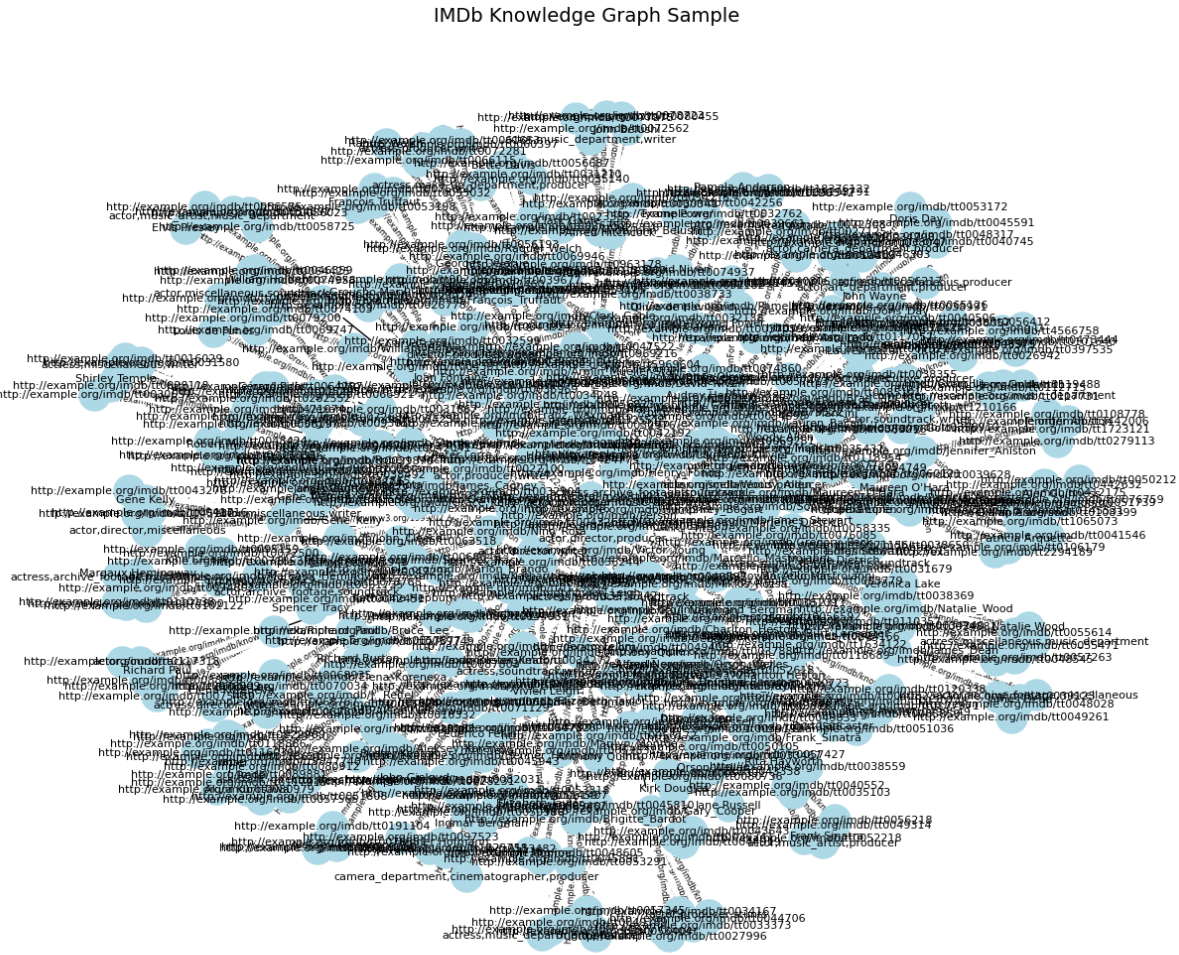

Figure 4.2: IMDb knowledge graph sample showing entities and relationships.

4.3. SHACL-based Semantic Quality Assurance

The implementation includes an SHACL-based semantic quality-assurance stage that validates and repairs the RDF graph across four targeted dimensions: lexical robustness, semantic coherence, temporal consistency, and redundancy elimination. In the pipeline, SHACL is used as a diagnostic and governance tool: it first quantifies violations in the raw graph, then guides targeted cleaning and repair operations, and finally verifies conformance after cleansing.

4.3.1. Pre-Cleaning SHACL Validation

This stage performs initial SHACL validation of the RDF knowledge graph before any cleaning or repair operations. The purpose of this step is to perform constraint checks on raw RDF data and to generate a detailed semantic violation report that drives remediation decisions and serves as input to subsequent detection and cleansing phases.

Loading SHACL Rules

The validation process begins by loading the SHACL constraint definitions from the shacl-rules.ttl file. These rules are parsed into a separate RDF graph using the same RDF processing framework as the data graph. Keeping the constraint graph independent ensures a clear separation between validation rules and instance data.

The SHACL rules define the semantic constraints that all entities in the RDF graph must satisfy, including:

- Required properties
- Datatype constraints
- Cardinality restrictions
- Structural expectations for object values

SHACL Rules Shape

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ns1: <http://example.org/imdb/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ns1:PersonShape a sh:NodeShape ;
    sh:targetClass ns1:Person ;

    sh:property [
        sh:path ns1:name ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:message "Each person must have exactly one name." ;
    ] ;

    sh:property [
        sh:path ns1:profession ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:in (
            "actor" "actress" "archive_footage" "assistant_director"
            "camera_department" "cinematographer" "composer"
            "director" "editor" "executive" "make_up_department"
            "miscellaneous" "music_artist" "music_department" "producer"
            "stunts" "writer"  "soundtrack" "unknown"
        ) ;
        sh:message "Profession must be one of the allowed values." ;
    ] ;

    sh:property [
        sh:path ns1:knownFor ;
        sh:minCount 1 ;
        sh:nodeKind sh:IRI ;
        sh:message "Each person must be known for at least one valid IRI." ;
    ] .
```

SHACL Shape Definition

The principal shape used in the experiments is `ns1:PersonShape`, which targets instances of `ns1:Person` and enforces three core constraints that implement the taxonomy's semantic requirements:

- **Name constraint.** Each `ns1:Person` must have exactly one `ns1:name`. The shape enforces presence (`sh:minCount 1`), uniqueness (`sh:maxCount 1`) and a datatype of `xsd:string`. A custom `sh:message` supplies a human-readable explanation for violations. This rule preserves lexical clarity and prevents multi-valued or missing name literals that would complicate entity linking.
- **Profession constraint.** Each person must have at least one `ns1:profession` literal (`sh:minCount 1`) typed as a string. The shape can include an `sh:in` whitelist of allowed profession tokens (for example, `actor`, `director`, `writer`) so that unexpected or invalid profession values are flagged for remediation. This constraint enforces semantic coherence across profession values.
- **Known-for constraint.** Each person must be linked to at least one `ns1:knownFor` IRI (`sh:minCount 1` and `sh:nodeKind sh:IRI`). This ensures person nodes are connected to meaningful title resources and avoids isolated person nodes that lack contextual links useful for recommendation and traversal.

Validation run and report

When the shapes and data graphs are loaded, the validator executes a SHACL run that returns a boolean conformance result and a detailed violation report. The report enumerates each `sh:ValidationResult` with fields such as:

- **focus node** — the subject node that triggered the violation,
- **result path** — the property path where the violation occurred (for example `ns1:profession`),
- **offending value** — the literal or IRI that failed the constraint (when applicable),
- **message** — the human-readable `sh:message` defined in the shape.

Persisting both the boolean conformance and the full violation list is essential as the boolean gives a quick pass/fail indicator, while the detailed report supplies the input for the automated remediation mapping applied in later pipeline stages.

4.3.2. Issue Detection and Cleaning Phases

The **cleaning stage** is the core of the pipeline, where targeted logic is applied to repair the RDF graph and its associated TSV data. This stage addresses four key semantic quality dimensions:

- **Lexical Robustness**
  - **Detection**: Detects malformed or noisy literals, such as names that are too short or contain digits.
  - Ensure that names are well-formed and human readable. It checks the minimum length (3 characters) and only allows letters, spaces, underscores, hyphens, periods, and apostrophes.
  - So, it filters out bad entities that are corrupted like "???", "123", or "@!$". The script Iterates over all IMDB.name triples and flags names that fail the is_lexical_robust check.
  - **Cleaning**: Removes triples with invalid names to ensure readability and consistency.
- **Redundancy Elimination**
  - **Detection**: Identify duplicate RDF triples. The logic tracks seen triples using a set and flags any repeated triples as actual redundancy.
  - **Cleaning**: Identifies and removes duplicate triples to reduce noise.
- **Semantic Coherence**
  - **Detection**: Validate that professions and known titles conform to expected formats. It checks that the profession is in the predefined set category (e.g., actor, director, etc.) and flags any unknown or misspelled professions (e.g., "actresss", "cinematographer").
  - **Cleaning**: After identifying profession values that do not match a predefined whitelist (e.g., "composer" or "unknown,actor").
  - Splits compound literals like "actor,director" into separate triples.
  - Filters out invalid roles and fills missing or unrecognized ones with "unknown" to maintain domain integrity.

- **Temporal Consistency**
  - **Detection**: Catch implausible or logically inconsistent birth/death years. It checks if the birth year or death year beyond the current year (2025) or birth year greater than death year or non-numeric or missing values that cause parsing errors. Then it returns a list of nconst identifiers with temporal anomalies.
  - **Cleaning**: After detecting illogical or unrealistic birth and death years (e.g., birthYear $>$ deathYear, or deathYear $>$ 2025)
  - Removes triples with invalid temporal values.
  - Completes missing or invalid birthYear with a default "1900" and deathYear with "2100" using xsd:gYear datatypes to satisfy SHACL constraints.

Semantic Data Repair Strategies

The transition from a removal-only strategy to a repair-oriented approach marks a significant advancement in semantic data quality assurance. This shift addresses the intrinsic theoretical tension between Consistency (freedom from contradiction) and Completeness (presence of all required facts).

While removal eliminates invalid data, it often leaves gaps that break SHACL shapes or reduce the completeness of the graph. By contrast, filling/ completing / patching preserves structural integrity and ensures that entities remain valid and usable.

This approach is most impactful for semantic coherence and temporal consistency:

- Repair Justification: For issues like missing professions or illogically ordered dates, patching with defaults ("unknown" or default years) ensures the entity satisfies the SHACL completeness contract. This retains the entity and its associated valid facts while explicitly flagging the repaired elements as derived or uncertain, which is formally quantified in the confidence scoring layer.
- Removal Justification: Meanwhile, lexical and redundancy issues are safely handled through removal, as eliminating these specific defects simplifies and purifies the data without compromising the logical structure or breaking mandatory entity properties.

4.3.3. Post-Cleaning SHACL Validation

Post-cleaning SHACL validation is executed immediately after all semantic cleansing and repair operations have been applied to the RDF graph.

The cleaned RDF graph is loaded into memory and validated using the same SHACL Shapes file (shacl_rules.ttl) employed during the pre-cleaning stage. No modifications are made to the constraint definitions to ensure consistency between the validation phases.

The SHACL validation engine processes the cleaned graph and evaluates each relevant node against the defined shapes. The validation procedure produces:

- A boolean conformance result indicating whether the graph satisfies all constraints
- A detailed validation report enumerating any remaining violations

The implementation explicitly checks the violation count to confirm that all previously detected issues such as invalid profession values, missing required properties, or inconsistent temporal literals have been resolved.

The final validation report is logged and stored as part of the pipeline output. A successful post-cleaning validation (i.e., full conformance with zero violations) marks the completion of the semantic quality assurance process and signals that the RDF graph is ready for downstream processing and evaluation.

4.4. Data Quality Confidence Scoring Layer

The confidence scoring layer is implemented as a post-processing step executed after the RDF graph has successfully passed post-cleaning SHACL validation. Its role is to compute and attach a numerical confidence score to each entity based on the corrections applied during earlier pipeline stages.

The scoring module computes a confidence value for each entity using a weighted penalty model. Starting from a maximum score of 1.0, penalties are applied according to the recorded metadata. The weights are fixed and applied uniformly across all entities to ensure consistency.

The resulting score is clamped to the [0,1] interval to prevent negative values.

Scoring Dimensions

During pre-cleaning validation and semantic cleansing, the pipeline records entity-level metadata required for scoring. For each RDF entity, the following values are collected:

- **Constraint Violations**: The number of SHACL violations detected before cleaning reflects the extent to which an entity deviates from expected semantic structure.
- **Repair Effort**: The number of repair operations applied to an entity indicates how much artificial correction was required to enforce consistency, such as replacing invalid values with defaults.
- **Missing Properties**: The absence of required attributes prior to cleaning signals incompleteness and uncertainty in the original data source.

Each dimension captures a different aspect of data quality risk and collectively provides a holistic estimate of entity reliability.

Scoring Model

The methodology adopts a penalty-based weighting model that subtracts quality penalties from a perfect confidence score of 1.0. This design ensures interpretability and monotonic behavior: increasing violations or repairs always decrease confidence.

$$\text{Confidence Score} = 1 - (\alpha \cdot \text{Violations}) - (\beta \cdot \text{Repairs}) - (\gamma \cdot \text{Missing Properties}) \quad (4.1)$$

The weighting coefficients (α, β, γ) reflect the relative impact of each dimension on perceived data reliability.

Algorithm 4.1 Compute confidence scores

Require: set `all_people`, map `quality_stats`, weights α, β, γ
Ensure: map `confidence_scores`

for all person in `all_people` **do**
  $v \leftarrow$ quality_stats[person].violations
  $r \leftarrow$ quality_stats[person].repairs
  $m \leftarrow$ quality_stats[person].missing
  $score \leftarrow 1 - (\alpha \cdot v) - (\beta \cdot r) - (\gamma \cdot m)$
  $score \leftarrow \max(0, \min(1, \text{round}(score, 3)))$
  store score in `confidence_scores` and add triple to graph
end for
return `confidence_scores`

4.5. Automated Constraint Discovery

After post-cleaning SHACL validation completes successfully, the RDF graph is reloaded as the input for constraint discovery. At this stage, the graph is considered clean where:

- Previously detected inconsistencies have been repaired
- The graph represents a reliable and trustworthy snapshot of the current data state

Importantly, this initial iteration is invigilated by a domain expert, who reviews and validates the applied repairs and assumptions to ensure the highest possible data quality. This expert supervision guarantees that the resulting graph constitutes a high-quality baseline, suitable for automated constraint discovery.

4.5.1. Class and Property Profiling

Each RDF subject in the graph is associated with one or more rdf:type values. For each subject:

- If one or more explicit types exist, they are used directly
- If no type is present, a domain-specific fallback heuristic assigns the subject to imdb:Person

Subjects are grouped by class, and for each class:

- All observed properties are collected
- All values associated with each property are recorded

This produces a class-level statistical profile that captures which properties occur, how often they appear, and what types of values they hold.

4.5.2. Cardinality, Datatype, and Node Kind Inference

From the collected class profiles, structural constraints are inferred automatically:

- **Cardinality Inference:** For each property, the minimum and maximum number of occurrences per instance are computed across all entities of the same class.
- **Datatype Profiling:** Literal values are inspected to identify consistent datatypes (e.g., xsd:gYear).
- **Node Kind Detection:** Values are classified as literals, IRIs, or blank nodes to infer sh:nodeKind constraints.

4.5.3. SHACL Shape Generation

Using the inferred statistics, a new SHACL shapes graph is constructed:

- A sh:NodeShape is created for each RDF class
- For each property:
  - A sh:PropertyShape is generated
  - sh:path, sh:minCount, and sh:maxCount are assigned
  - sh:datatype or sh:nodeKind constraints are added where applicable
- Property shapes are linked to their corresponding node shapes

The resulting SHACL graph is serialized into a Turtle file representing the auto-discovered constraints of the cleaned knowledge graph.

4.5.4. Drift Detection via Shape Comparison

To detect constraint drift, the system compares two sets of SHACL Shapes:

- Original SHACL shapes (expert-defined baseline)
- Auto-discovered SHACL shapes (data-driven)

Both SHACL files are loaded as RDF graphs. A mapping is created from each property's sh:path to its associated property shape to enable efficient lookup.

For each property discovered in the auto-discovered shapes the system:

- Checks whether the property exists in the original SHACL rules
- Compares the following constraints dimensions:
  - Minimum cardinality
  - Maximum cardinality
  - Datatype
  - Node kind

Differences are recorded as explicit drift events. If a property does not exist in the original shapes, it is flagged as an emergent schema element.

4.5.5. Drift Detection Outcomes

Two outcomes are possible:

Case 1: No Drift Detected

The automatically discovered shapes exactly match the original SHACL rules.

This indicates that:

- No drift is reported as the cleaned RDF graph still follows the original structural assumptions
- The SHACL validation model remains valid
- No schema update is required

This is a stability condition: the feedback loop confirms that the model and the data are aligned.

Case 2: Drift Detected

When the automatically discovered shapes reveal differences from the original SHACL rules, the system identifies these as explicit drift events. Examples include:

- A property's maximum cardinality changing (e.g., $4 \rightarrow 3$)
- Datatypes becoming inconsistent (e.g., xsd:gYear $\rightarrow$ string)
- A newly observed property appearing in the RDF graph

Algorithm 4.2 Drift Detection Algorithm

Require: original_shapes.ttl, discovered_shapes.ttl

load original, discovered
build map originalMap: sh:path → PropertyShape
for all propShape in discovered where type = sh:PropertyShape **do**
  extract path, d_min, d_max, d_dtype, d_nodeKind
  if path ∉ originalMap **then**
    record emergent property(path)
  else
    extract o_min,o_max,o_dtype,o_nodeKind from originalMap[path]
    if any(d_min≠o_min, d_max≠o_max, d_dtype≠o_dtype, d_nodeKind≠o_nodeKind) **then**
      record drift(path, differences)
    end if
  end if
end for
if no drift recorded **then**
  output "No drift detected."
end if

4.5.6. Constraint Drift Severity Index

While detecting the presence of constraint drift is important, a binary indication of drift is insufficient for managing evolving knowledge graphs. Not all drift events have the same semantic impact as the introduction of a new optional property is far less severe than the removal of a mandatory constraint or a fundamental change in datatype. Constraint Drift Severity Index (CDSI) is a quantitative metric designed to measure the degree and nature of deviation between original SHACL constraints and those automatically discovered constraints. CDSI doesn't only measure whether a drift has occurred but also how severe that drift is.

The CDSI operates at the level of SHACL property shapes and evaluates drift along three complementary dimensions: cardinality drift, datatype drift, and node kind drift. Each dimension captures a distinct aspect of structural evolution in the knowledge graph and contributes to the overall drift score.

CDSI Computation

For each property associated with a given RDF class, the automatically discovered SHACL constraints are compared against the original SHACL specification. Drift scores are computed independently for the three dimensions and normalized to a value.

The final CDSI score for a property is calculated as the average of the three component scores:

$$\text{CDSI}_{\text{property}} = \frac{1}{3}\left(D_{\text{cardinality}} + D_{\text{datatype}} + D_{\text{nodeKind}}\right)$$

The overall CDSI for the knowledge graph is obtained by summing the per-property CDSI scores across all evaluated properties. This aggregation provides a single scalar value representing the cumulative severity of constraint drift in the graph.

$$\text{CDSI}_{\text{total}} = \sum \text{CDSI}_{\text{property}}$$

$$\text{CDSI}_{\text{normalized}} = \frac{1}{\text{property}} \sum \text{CDSI}_{\text{property}}$$

Each dimension produces a normalized score in the range $[0, 1]$, where higher values indicate more severe drift.

Cardinality Drift Scoring

The `cardinality_drift` function measures deviations between original and discovered cardinalities. The scoring rules prioritize semantic impact over numeric difference:

Missing cardinality values (i.e., `None`) are treated as zero to ensure consistent comparison.

- Removing a mandatory constraint (e.g., original `minCount` $>$ 0 and discovered `minCount` $= 0$) results in maximum severity (1.0).
- Converting an optional property into a mandatory one is scored as major drift (0.9).
- Small numeric deviations in cardinality bounds are assigned moderate scores in the range 0.2–0.6, proportional to the absolute difference in minimum and maximum counts.
- Identical cardinality constraints produce zero drift (0.0).

Datatype and Node Kind Drift Scoring

Drift is scored according to the following rules:

- Identical datatype and node kind constraints yield no drift (0.0).
- Datatype changes between literal values (e.g., `xsd:gYear` to `xsd:string`) are scored as low drift (0.25).
- Transitions between literals and IRIs are scored as high drift (0.6).
- Any change in node kind (e.g., `sh:Literal` to `sh:IRI`) results in maximum severity (1.0).

Node kind drift is evaluated explicitly because such changes alter the conceptual modeling of a property and can significantly affect downstream reasoning and query behavior.

Handling New and Removed Properties

Properties present in the discovered SHACL shapes but absent from the original schema are treated as schema extensions. These are assigned a moderate drift score (0.6), reflecting structural evolution rather than violation.

Conversely, properties that existed in the original SHACL rules but are absent in the discovered shapes are penalized more heavily when they were originally mandatory. This accounts for the loss of enforced structural guarantees and signals potentially destructive schema drift.

4.5.7. Constraint Update Policies

After constraint drift has been detected and quantified using the Constraint Drift Severity Index (CDSI), the system determines how the SHACL rules should be updated. Rather than applying all changes indiscriminately, the pipeline supports three distinct constraint update policies that control how automatically discovered constraints are integrated into the validation framework.

Strict Update Policy

Under the **Strict Update** policy, the original SHACL rules are fully replaced by the automatically discovered constraints derived from the validated RDF graph. This approach assumes that the current structure of the knowledge graph represents the most accurate and authoritative schema. All detected changes, including new properties, modified

cardinalities, and datatype updates, are applied without manual intervention.

While this policy maximizes adaptability, it may overwrite domain-specific modeling decisions and is therefore best suited for highly dynamic or weakly governed knowledge graphs.

Conservative Update Policy

The **Conservative Update** policy prioritizes schema stability by preserving all original SHACL constraints. Automatically discovered changes are only applied when they relax existing restrictions, such as increasing a maximum cardinality or allowing additional optional properties. Any changes that would strengthen constraints or remove mandatory requirements are ignored.

This policy minimizes the risk of unintended schema evolution but may fail to capture legitimate structural changes in the data.

Hybrid Update Policy

The **Hybrid Update** policy combines automated adaptation with controlled governance and is implemented as the default strategy in the pipeline. Under this policy, automatically discovered constraints are evaluated individually and categorized as either safe or risky updates.

Safe updates, such as the introduction of new properties, datatype consistency improvements, or relaxed `maxCount` values, are applied automatically. Potentially risky changes, including reductions in `minCount` or removal of mandatory constraints, are not applied directly. Instead, they are logged and flagged for manual review.

This hybrid strategy enables SHACL constraints to evolve in response to data-driven patterns while preserving expert-defined semantics and preventing uncontrolled schema degradation. It provides a balanced approach that supports long-term adaptability, accountability, and trust in evolving knowledge graph environments.

5 | Results and Evaluation

5.1. Robustness Stress Test

5.1.1. Synthetic Corruption

Real-world data sources often suffer from various forms of degradation due to human error, inconsistent integration, or evolving schema standards. To rigorously evaluate the effectiveness of a Data Quality Management framework, this phase introduces synthetic corruption as a controlled, measurable means of simulating data errors.

The goal of this phase is to intentionally inject corrupt or noisy data into a clean RDF graph with the intention of:

- Stress-test the robustness of cleaning algorithms.
- Evaluate the detection accuracy across multiple data quality dimensions.
- Analyze how injected corruption affects the structural integrity of the knowledge graph by comparing the results on clean versus degraded datasets.

Injection Implementation

Corruptions were injected based on four key data quality dimensions:

- **Lexical Robustness:** Malformed names were introduced. So having literals containing invalid characters (e.g., `Bad@Name#3`). These simulate typographical or encoding errors during data entry or ingestion.
- **Semantic Coherence:** Disallowed or nonsensical values were injected. For example, assigning a non-existent profession like “alien” alongside legitimate ones such as “actor violates domain rules.
- **Temporal Consistency:** Temporal inconsistencies were introduced, including birth years in the future (e.g., 3000) or logically impossible lifespan entries (e.g., birth after death). These mimic date parsing issues in temporal data integration.

- **Redundancy Detection:** Duplicate triples were introduced to simulate repeated ingestion. To test the system's ability to identify and handle redundant data.

Each type of corruption was injected in a way that preserved RDF compliance, making it more challenging to detect while still simulating realistic data quality issues.

Pre-Cleaning SHACL Results

Before applying any detection, cleaning, or correction routines. The RDF graph is validated against the SHACL rules defined in `shacl_rules.ttl` to assess its semantic quality. This step serves two purposes: first, to confirm that the injected corruption is declared as a formal SHACL violation. Secondly, to measure the degree of violation before any corrective intervention is taken.

The output of the pre-cleaning SHACL validation is summarized below:

```
PRE-CLEANING SHACL VALIDATION
SHACL Conforms (Pre-cleaning): False
Violations (Pre-cleaning): 99
```

The RDF graph does not conform to the SHACL rules defined in the shacl_rules.ttl file, so it is indicated as FALSE. A total of 99 individual constraint violations were detected in the RDF graph. Each violation corresponds to a specific rule in the SHACL shape that was broken by a particular node.

Example Violation Analysis

The SHACL validation report highlights a semantic inconsistency in the RDF graph with respect to the predefined constraints. In this instance, the focus node ns1:Gene_Tierney has the property ns1:profession set to the compound string

```
"actress,archive_footage,soundtrack, spaceship"
```

which encodes multiple roles in a single literal. The corresponding SHACL property shape enforces an sh:InConstraintComponent, requiring that each ns1:profession value be a single literal chosen from a predefined set of allowed professions. While most roles in the string are valid, the inclusion of "spaceship" violates the constraint. Consequently, the node fails validation and is reported as a semantic violation. This example demonstrates how SHACL can identify domain-level inconsistencies that are syntactically correct RDF but semantically invalid. The violation report provides detailed information, including the focus node, the property path (ns1:profession), the offending value, and the shape message.

This facilitates deterministic handling such as normalization, splitting, replacement, or removal of invalid entries.

Detection Summary

After applying the detection functions on the RDF graph, a contaminated version of `imdb.ttl` was analyzed. Table 5.1 summarizes the number of issues identified across different data quality dimensions.

Issue Type	**Detected Count**
Lexical issues	105
Semantic issues	122
Temporal issues	87
Redundant triples	100

Table 5.1: Summary of detected data quality issues in the contaminated RDF graph.

These results demonstrate the sensitivity of the detection functions to different forms of corruption. The temporal dimension highlights inconsistencies introduced by logically impossible dates, while lexical and semantic issues reflect malformed literals and invalid domain values. Redundant triples simulate repeated ingestion events. Overall, this detection step provides a controlled evaluation of the framework's robustness prior to constraint driven cleaning and normalization.

Corrupted knowledge Graph Visualization

The visualization in Figure 5.1 displays a sample of 50 triples from the corrupted RDF to highlight bad literals accompanied with a sample.

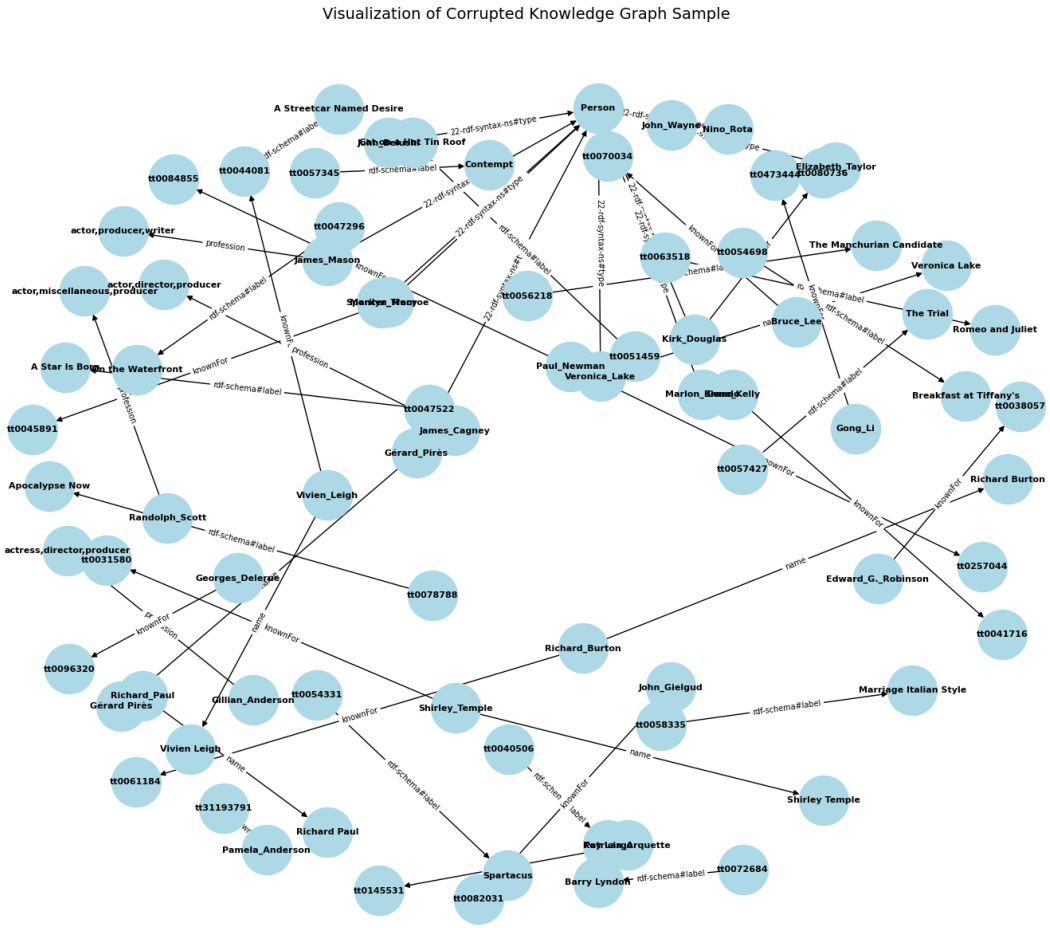

Figure 5.1: Visualization of Corrupted KG sample

`http://example.org/imdb/Bad_Semantic_Entity -- http://example.org/imdb/profession --> spaceship`

`http://example.org/imdb/Bad_Lexical_Entity -- http://example.org/imdb/name --> J@hn D0e!!!`

`http://example.org/imdb/Bad_Redundant_Entity -- http://example.org/imdb/name --> Duplicate Name`

`http://example.org/imdb/Bad_Temporal_Entity -- http://example.org/imdb/`

Constraint-Guided Cleaning Phase

The cleaning phase applies deterministic rule-based transformations to the contaminated RDF graph and the accompanying TSV source in order to remove lexical noise, resolve redundancy, normalize multi-valued literals, and patch temporal fields. The goal of this phase is to produce a semantically cleaner snapshot that can be re-validated with SHACL and used as the input for subsequent profiling and constraint induction.

```
The cleaned_imdb_after_cleaning.ttl file looks as follow:

RDF Graph loaded with 1012 triples.

Sample Triples:
1: http://example.org/imdb/Henry_Fonda -- http://example.org/imdb/knownFor --> http://example.org/imdb/tt0082846
2: http://example.org/imdb/Groucho_Marx -- http://example.org/imdb/knownFor --> http://example.org/imdb/tt0072322
3: http://example.org/imdb/Laurence_Olivier -- http://example.org/imdb/knownFor --> http://example.org/imdb/tt0060827
4: http://example.org/imdb/Stanley_Kubrick -- http://example.org/imdb/name --> Stanley Kubrick
5: http://example.org/imdb/James_Horner -- http://example.org/imdb/knownFor --> http://example.org/imdb/tt0499549
```

The cleaning routine follows three main steps:

1. **Remove detected lexical and redundant triples.** Triples flagged by the detection stage are removed from the graph to eliminate obvious syntactic and duplicate noise.

2. **Normalize profession values.** Compound profession literals (comma-separated lists) are split, trimmed, filtered against an `allowed_professions` vocabulary, and reinserted as separate profession triples. If any token is invalid, an `"unknown"` marker is added to preserve the fact that the original value contained problematic content.

3. **Patch temporal fields and TSV rows.** `birthYear` and `deathYear` values are validated for the expected datatype (`xsd:gYear`); missing or invalid values are replaced with conservative defaults.

```
Person: http://example.org/imdb/Bruce_Lee
  Name: Bruce Lee
  Professions: actor, miscellaneous, writer
  KnownFor: http://example.org/imdb/tt0067824, http://example.org/imdb/tt0068767, http://example.org/imdb/tt0068935, http://example.org/imdb/tt0070034

Person: http://example.org/imdb/Burt_Lancaster
  Name: Burt Lancaster
  Professions: actor, miscellaneous, producer
  KnownFor: http://example.org/imdb/tt0040823, http://example.org/imdb/tt0045793, http://example.org/imdb/tt0051036, http://example.org/imdb/tt0085859

Person: http://example.org/imdb/Buster_Keaton
  Name: Buster Keaton
  Professions: actor, director, writer
  KnownFor: http://example.org/imdb/tt0015163, http://example.org/imdb/tt0015324, http://example.org/imdb/tt0016332, http://example.org/imdb/tt0017925

Person: http://example.org/imdb/Cary_Grant
  Name: Cary Grant
  Professions: actor, producer, soundtrack, unknown
  KnownFor: http://example.org/imdb/tt0032599, http://example.org/imdb/tt0032904, http://example.org/imdb/tt0036613, http://example.org/imdb/tt0056923
```

Figure 5.2: Cleaned RDF tuple sample

Post-Cleaning SHACL Results

After the completion of the constraint-guided cleaning procedures, the repaired RDF graph and its associated tabular data were re-validated against the same SHACL rules defined in `shacl_rules.ttl`. This post-cleaning validation serves as a verification step to assess the effectiveness of the applied interventions and to confirm that all previously detected semantic violations have been resolved.

The output of the post-cleaning SHACL validation is summarized below:

```
POST-CLEANING SHACL VALIDATION
SHACL Conforms (Post-cleaning): True
Violations (Post-cleaning): 0
```

The validation outcome indicates that the RDF graph fully conforms to the defined SHACL constraints, as evidenced by the `True` conformance flag and the absence of any remaining violations. This confirms that the cleaning logic successfully addressed lexical, semantic, temporal, and structural inconsistencies identified in earlier phases.

Validation Report Preview

A preview of the validation report was also examined to ensure that no latent violations were overlooked. The report confirms full conformance, which demonstrates that the resulting RDF graph is semantically consistent and structurally complete.

The output of the validation report is summarized below:

```
Validation Report
Conforms: True
```

Confidence Scoring Layer

The confidence scoring layer transforms qualitative validation and repair signals into a quantitative measure of entity reliability. Rather than treating SHACL conformance as a binary outcome. This model interprets violations, repair interventions, and missing properties as indicators of semantic uncertainty and data lineage risk. Each detected issue contributes a proportional penalty to the ideal confidence value of 1.0, which measures the degree of intervention required for an entity to achieve compliance.

The resulting confidence score, therefore, represents a graded estimate of trustworthiness. So entities that naturally conform to constraints retain high confidence, while entities requiring extensive correction receive lower scores.

$$\text{Confidence}(e) = 1 - (0.1 \cdot V_e + 0.1 \cdot R_e + 0.05 \cdot M_e) \tag{5.1}$$

Dimension	Description	Weight Symbol	Default Weight
Violations	Number of SHACL violations detected before cleaning	α	0.1
Repairs	Number of triples repaired or replaced (e.g., with "unknown" values)	β	0.1
Missing Properties	Number of properties absent pre-cleaning (e.g., birth year, death year, and profession)	γ	0.05

Table 5.2: Dimensions and default weights used in the confidence scoring formula.

Confidence Score Results and Interpretation

Table 5.3 presents five representative entities sampled from the cleaned RDF graph, illustrating how the proposed confidence scoring layer differentiates entity reliability based on detected violations, applied repairs, and missing properties.

Entity	Violations	Repairs	Missing	Score
`imdb:Brad_Pitt`	1	1	0	0.80
`imdb:Bad_Redundant_Entity`	0	0	2	0.90
`imdb:Buster_Keaton`	1	0	0	0.90
`imdb:Gene_Kelly`	1	1	1	0.75
`imdb:J._Reifel`	0	0	0	1.00

Table 5.3: Five sample entities with confidence scores

Interpretation of Results

The assigned confidence scores reflect the cumulative impact on entity trustworthiness:

- **High Integrity (1.00):** Entities such as `J._Reifel` exhibit no violations, repairs, or missing attributes. These entities fully conform to all SHACL constraints and represent high-integrity data suitable for all analytical and decision-critical tasks.
- **Moderate Trust (0.90):** Entities with minor issues, such as a single violation or a missing optional attribute (e.g., `Buster_Keaton`) can achieve scores around 0.90. These entities remain reliable and are suitable for general-purpose recommendation and exploratory analysis.
- **Low Trust (0.80):** Entities that required both violation correction and repair, such as `Brad_Pitt`, receive moderate confidence scores. While usable, they reflect partial deviation from the ideal schema and should be treated with caution in high-stakes applications.
- **Unreliable (0.75 and below):** Entities exhibiting a combination of violations, repairs, and missing attributes, such as `Gene_Kelly`, are penalized more heavily. These entities are candidates for exclusion or manual remediation due to reduced trustworthiness.

This continuous confidence scoring framework provides a more refined assessment of data quality by quantifying degrees of trustworthiness. It enables selective data usage based on predefined confidence thresholds, such as restricting analytical queries to entities with a `dq:confidenceScore` of at least 0.9.

Automated Constraint Discovery

After the first iteration a synthetic individual(Person123) was injected into the cleaned IMDB graph with intentionally altered and unexpected characteristics designed to trigger specific forms of constraint drift. To evaluate the robustness of the automated constraint-drift detection.

This drift scenario serves as a stress test to determine whether the system can:

- Detect cardinality drift.
- Identify unexpected new properties not defined in the original SHACL shapes.
- Validate that the methodology is robust against subtle and explicit deviations.

The goal is to prove that the constraint-discovery and drift-detection pipeline can reliably flag deviations when the data evolves in ways the ontology did not originally anticipate.

Types of Drift Introduced

- Schema Drift: A new property (imdb:award) was added, even though it does not exist in the original SHACL shapes.
- Cardinality Drift: The imdb:knownFor property was given fewer values than usual, reducing its expected minimum count.
- Datatype/Ontological Drift: A new property (imdb:birthYear with datatype xsd:gYear) was introduced, even though it was not part of the original schema.

Drift Report Result

```
Constraint Drift Report
- http://example.org/imdb/Person | http://example.org/imdb/award: NEW property
  with constraints min=0, max=1,
  dt=http://www.w3.org/2001/XMLSchema#string, nk=None

- http://example.org/imdb/Person | http://example.org/imdb/birthYear: NEW property
  with constraints min=1, max=1,
  dt=http://www.w3.org/2001/XMLSchema#gYear, nk=None

- http://example.org/imdb/Person | http://example.org/imdb/deathYear: NEW property
  with constraints min=0, max=1,
  dt=http://www.w3.org/2001/XMLSchema#gYear, nk=None

- http://example.org/imdb/Person | http://example.org/imdb/knownFor: maxCount 0 → 4
- http://example.org/imdb/Person | http://example.org/imdb/knownFor: minCount 1 → 2

- http://example.org/imdb/Person | http://example.org/imdb/profession: maxCount 0 → 4
- http://example.org/imdb/Person | http://example.org/imdb/profession: minCount 1 → 0
```

Auto-discovered SHACL

```
dq:PersonShape a sh:NodeShape ;
   sh:property dq:PersonShape_award,
               dq:PersonShape_birthYear,
               dq:PersonShape_deathYear,
               dq:PersonShape_knownFor,
               dq:PersonShape_name,
               dq:PersonShape_profession ;
   sh:targetClass imdb:Person .

dq:PersonShape_award
   sh:datatype xsd:string ;
   sh:maxCount 1 ;
   sh:path imdb:award .

dq:PersonShape_birthYear
   sh:datatype xsd:gYear ;
   sh:maxCount 1 ;
   sh:minCount 1 ;
   sh:path imdb:birthYear .

dq:PersonShape_deathYear
   sh:datatype xsd:gYear ;
   sh:maxCount 1 ;
   sh:path imdb:deathYear .

dq:PersonShape_knownFor
   sh:maxCount 4 ;
   sh:minCount 2 ;
   sh:nodeKind sh:IRI ;
   sh:path imdb:knownFor .

dq:PersonShape_name
   sh:datatype xsd:string ;
   sh:maxCount 1 ;
   sh:minCount 1 ;
   sh:path imdb:name .

dq:PersonShape_profession
   sh:datatype xsd:string ;
   sh:maxCount 4 ;
   sh:path imdb:profession .
```

CDSI Normalization and Aggregation

The drift analysis between the original SHACL schema and the automatically discovered schema reveals that several new properties were introduced for the class imdb:Person. Specifically, the properties award, birthYear, deathYear, and rdfs:label appear in the discovered shapes but were not present in the original schema. Because these properties are entirely new, the system assigns each of them a moderate drift score of approximately 0.20, shows that the schema has expanded but without violating any mandatory constraints from the original model.

For the properties that existed in both schemas the drift varies. The property imdb:name shows a drift score of 0.0, meaning it remained completely unchanged in terms of cardinality, datatype, and node kind. In contrast, imdb:knownFor and imdb:profession each receive a score of 0.3333, indicating that at least one of their structural constraints has changed enough to produce a moderate level of drift. These changes do not represent severe schema breakage, but they do signal meaningful evolution in how these properties are defined.

```
Drift Report:
- http://example.org/imdb/Person | http://example.org/imdb/award: NEW property
- http://example.org/imdb/Person | http://example.org/imdb/birthYear: NEW property
- http://example.org/imdb/Person | http://example.org/imdb/deathYear: NEW property
- http://example.org/imdb/Person | http://www.w3.org/2000/01/rdf-schema#label: NEW property

Per-property scores:
('http://example.org/imdb/Person', 'http://example.org/imdb/award') -> 0.19999999999999998
('http://example.org/imdb/Person', 'http://example.org/imdb/birthYear') -> 0.19999999999999998
('http://example.org/imdb/Person', 'http://example.org/imdb/deathYear') -> 0.19999999999999998
('http://example.org/imdb/Person', 'http://example.org/imdb/knownFor') -> 0.3333333333333333
('http://example.org/imdb/Person', 'http://example.org/imdb/name') -> 0.0
('http://example.org/imdb/Person', 'http://example.org/imdb/profession') -> 0.3333333333333333
('http://example.org/imdb/Person', 'http://www.w3.org/2000/01/rdf-schema#label') -> 0.19999999999999998

TOTAL CDST = 1.4666666666666666

CDSInormalized = 1/7 * 1.4667 = 0.21
```

When all property-level drift scores are combined, the total CDSI is 1.4667. Since this value depends on the number of properties, it is normalized by dividing the total number of evaluated properties.

The resulting normalized CDSI is approximately 0.21, which falls within the "Minor Drift" severity range. This indicates that while the discovered schema introduces new information and modifies some constraints, the overall structure remains largely consistent with the original SHACL model. The schema has evolved, but not in a way that threatens compatibility or semantic integrity.

Hybrid Update Policy

After constraint drift is detected and quantified using the Constraint Drift Severity Index (CDSI), the pipeline updates the active SHACL model according to the Hybrid Update policy. The policy interprets the CDSI drift report and determines which updates are applied automatically and which are deferred to manual review by being placed in the review queue. The resulting SHACL model is then used as the pre-SHACL validation schema for subsequent ingested data. This policy enables automatic incorporation of low-risk structural changes while preventing uncontrolled schema strengthening.

Decision Criteria

Update decisions were derived deterministically from the CDSI output and observed data statistics:

- Newly detected properties were classified as safe when their datatype distribution was stable or when their per-property CDSI exceeded the auto-apply threshold.
- Relaxations of existing constraints (e.g., increases in `sh:maxCount`) were applied automatically.
- Strengthening updates (e.g., the introduction or increase of `sh:minCount`) were deferred for manual inspection.
- All update decisions were logged together with the corresponding CDSI scores.

Auto-applied SHACL update

The following Turtle fragment shows the subset of SHACL constraints that were automatically applied and persisted as the current validation model. Deferred (pending-review) updates are excluded.

```
dq:PersonShape a sh:NodeShape ;
sh:targetClass imdb:Person ;
sh:property dq:PersonShape_award,
dq:PersonShape_birthYear,
dq:PersonShape_deathYear,
dq:PersonShape_knownFor,
dq:PersonShape_name,
dq:PersonShape_profession .

dq:PersonShape_award
sh:path imdb:award ;
sh:datatype xsd:string ;
sh:maxCount 1 .
```

```
dq:PersonShape_birthYear
sh:path imdb:birthYear ;
sh:datatype xsd:gYear ;
sh:maxCount 1 .

dq:PersonShape_deathYear
sh:path imdb:deathYear ;
sh:datatype xsd:gYear ;
sh:maxCount 1 .

dq:PersonShape_knownFor
sh:path imdb:knownFor ;
sh:nodeKind sh:IRI ;
sh:maxCount 4 .

dq:PersonShape_name
sh:path imdb:name ;
sh:datatype xsd:string ;
sh:minCount 1 ;
sh:maxCount 1 .

dq:PersonShape_profession
sh:path imdb:profession ;
sh:datatype xsd:string ;
sh:maxCount 4 .
```

Pending Review Updates

Updates classified as potentially strengthening were recorded but not applied. In this run, the pending review queue included:

- `imdb:DeathYear`: proposed `sh:minCount 1`
- `imdb:knownFor`: proposed `sh:minCount 2`
- `imdb:profession`: any proposed `sh:minCount`.

These items remain inactive until explicitly approved and promoted.

Closed-loop Effect

By feeding the auto-applied SHACL updates back into the Pre-SHACL validation stage, newly ingested RDF data is validated against a schema that reflects observed structural changes. This configuration reduced false positives in pre-SHACL validation for incoming data while preserving expert control over potentially breaking schema changes.

5.2. Pre-cleaning Recommendation Behavior under Semantic Noise

A simulated recommendation scenario test case is performed to demonstrate the practical implications of poor data quality in knowledge graphs.

"A knowledge graph of a movie recommendation system that suggests similar actors based on shared professional roles and temporal proximity. The seed entity selected for this case was Grace Kelly, a well-known actress, with the expectation that the system would identify and recommend contemporaries with matching professions and overlapping life spans."

Recommendation Query	Expected Results

`SELECT actor.name`
`FROM actors AS actor`
`WHERE actor.profession IN ('actor', 'actress')`
`AND actor.birthYear BETWEEN 1920 AND 1940`
`AND (actor.deathYear IS NULL`
`OR actor.deathYear >= 1950);`
	• Audrey Hepburn
	• Joan Fontaine
	• Elizabeth Taylor

Table 5.4: Recommendation query and expected results under ideal data conditions

Generated Results

However, the model recommends questionable results: a profile named "Bad@Name#1", another called "Bad Future Actor" who is inexplicably born in the year 3000, and Joan Fontaine, whose listed profession is the nonsensical "perlactor" due to a typo.

These faulty recommendations stem from various data-quality issues in the knowledge graph. Lexical errors, such as invalid characters in names, semantic errors (e.g., incorrect or misspelled professions), temporal inconsistencies (e.g., birth dates in the future or death preceding birth), and redundant entries all contribute to misleading or irrelevant suggestions.

Generated Results Visualization

Expanded Version:

- Red nodes = lexical issues (invalid characters in names). The name contains special characters such as `@` or `#`, which violate the lexical validation rules (only letters, spaces, underscores, hyphens, and apostrophes are allowed).
- Orange nodes = semantic issues (invalid profession values). For example, "perlactor" is not a valid profession within the allowed set (actor, director, writer, etc.).
- Purple nodes = temporal issues (birth/death year inconsistencies). Temporal contradictions, such as a birth year of 3000 and a death year of 2999 (before birth), introduce logical errors in the timeline.
- Green edges = redundant (duplicate) triples. Duplicate relationships can skew ranking or weighting mechanisms within the knowledge graph.
- Light blue nodes = clean entities. These represent correctly validated entries that conform to lexical, semantic, and temporal rules.

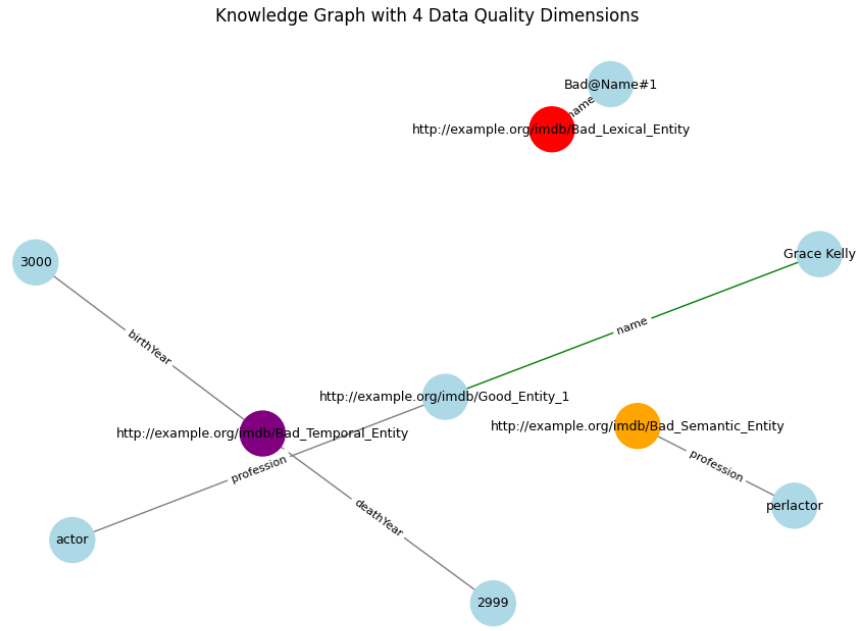

Figure 5.3: Visualization of the unclean query knowledge graph

Client's Recommendation

From a client-facing perspective, these issues manifest as the following recommendation failures:

- **Omission of relevant actors** due to lexical and semantic inconsistencies.
- **Inclusion of irrelevant or impossible entities** caused by temporal anomalies.
- **Over-representation of certain actors** resulting from duplicated graph assertions.

5.3. Impact of Data Cleaning on Recommendation Input Quality

To illustrate the impact of data quality on recommendation systems, we consider a movie recommendation scenario in which users discover films based on directors they admire.

The underlying knowledge graph represents entities and relationships among users, directors, and films. In this setting, candidate films are retrieved by traversing the following path:

User → Likes Director → Known For Films

For example, a user visits a movie recommendation platform and selects **Akira Kurosawa** as their favorite director. The system queries the knowledge graph to retrieve the films associated with the director via the `imdb:knownFor` relationship. These films form the candidate set that is subsequently consumed by recommendation or ranking models.

Candidate Retrieval Query	Expected Results

`SELECT object AS film`
`FROM triples`
`WHERE subject =`
`'http://example.org/imdb/Akira_Kurosawa'`
`AND predicate =`
`'http://example.org/imdb/knownFor';`
	• tt0051808
	• tt0057565
	• tt0080979
	• tt0089881

Table 5.5: Candidate film retrieval query and expected results for Akira Kurosawa after data cleaning.

Generated Results

After applying the data cleaning workflow, which systematically addressed lexical, semantic, temporal, and redundancy-related inconsistencies, the candidate retrieval query returned the expected and semantically valid results. The cleaned RDF graph produced a coherent and complete set of films associated with the director **Akira Kurosawa**.

As a result, the recommendation pipeline operates on a high-quality candidate set, ensuring that downstream recommendation and ranking components receive accurate and reliable inputs.

Prompt Outcome

```
Movies known for Akira Kurosawa:
http://example.org/imdb/tt0051808
http://example.org/imdb/tt0057565
http://example.org/imdb/tt0080979
http://example.org/imdb/tt0089881
```

This experiment demonstrates that data cleaning plays a critical role in stabilizing the input layer of recommendation systems by ensuring that candidate retrieval from the knowledge graph is both correct and complete.

5.4. Impact of Data Fusion and Entity Resolution on Knowledge Graph Integration

A data fusion simulation is assessed to show how entity resolution errors affect integrated knowledge graphs. The findings shows the final graph state acquired after running the whole pipeline, which included pre- and post-SHACL validation, violation detection, data cleaning, confidence scoring, and automated constraint enforcement.

To assess the influence of entity resolution on semantic integrity, a data fusion experiment was carried out by matching IMDb person identifiers with their corresponding Wikidata entities. IMDb and Wikidata have considerable differences in schema structure, identifier precision, and attribute accuracy, making them typical of diverse integration scenarios in real domain.

Fusion Task and Expected Results

The anticipated result of the fusion process is the merging of all IMDb identifiers associated with the same human in the real-world into a single Wikidata object, while maintaining semantic distinctions between various individuals.

In particular, the integrated graph is intended to:

- Eliminate duplicated entities
- Merge similar attributes into a single uniform profile
- Avoid conflation of unrelated individuals

Observed Results

Initial results anticipated multiple IMDb identifiers mapping to the same Wikidata entity. For example, three distinct IMDb identifiers (nm0018187, nm0018188, nm0018189) were associated with Wikidata entity Q3609386. Similar patterns were observed for other IMDb entities that include nm0074193, nm0074194 and nm0074195 are linked to Q61670.

Without semantic resolution the yielded patterns raises the risk of both under merging that results in entity fragmentation and over-merging which results in inflated counts of entity based queries.

Post-fusion Results

After performing the intended pipeline workflow, all IMDb and Wikidata mappings obtained full validation scores across lexical, semantic, temporal, and redundancy dimensions. These scores show that the matched IDs consistently correspond to the same individuals in the real-world.

Validated fusion results include:

```
Resolved entity: Q3609386
  - nm0018187 -> merged (Score: 4)
  - nm0018188 -> merged (Score: 4)
  - nm0018189 -> merged (Score: 4)

Resolved entity: Q61670
  - nm0074193 -> merged (Score: 4)
  - nm0074194 -> merged (Score: 4)
  - nm0074195 -> merged (Score: 4)
```

Impact on Integrated Graph Quality

Following the fusion and integration between IMDb and Wikidata, the knowledge graph showed significant improvements in structural and semantic quality across many data quality metrics. In order to reduce ambiguity and noise, heterogeneous IDs were combined into unified representations and traits that had previously been inconsistent or misaligned across multiple sources are now resolved into coherent entity profiles.

Clean entity integration enhances the dependability of graph driven applications from a user viewpoint. While lexical normalization stopped incorrect identifiers from contaminating query output, entity oriented searches produced consistent and non-redundant results. Temporal checks removed unrealistic dates that may potentially skew downstream behavior, while semantic validation guaranteed that objects and types maintained uniformity throughout fusion. This guaranteed that recommendation models ran on coherent inputs and decreased the number of biased or missing outcomes.

The established connection also helped with analytical and decision support roles. The aggregates and statistics obtained from the graph are no longer biased by redundant entities, semantically inconsistent traits, or contradictory temporal results. The trust in graph based insights increased as a result of analytics reflecting real-world patterns rather than products of unresolved fusion problems.

6 | Strengths and Weaknesses

6.1. Reversing the Induction Sequence

6.1.1. Limitations of Conventional Induction Pipelines

The central conceptual contribution of this work lies in a deliberate inversion of the conventional relationship between constraint induction and validation in RDF knowledge graphs. In prevailing approaches, structural or semantic constraints are inferred directly from the observed graph and subsequently used as a reference model for validation. This ordering implicitly assumes that the current state of the data provides a reliable basis for defining what constitutes valid data.

However, knowledge graphs are rarely pristine. They evolve through heterogeneous sources, incremental updates, and varying editorial standards. Constraints derived from such unrefined data risk encoding noise, inconsistencies, or temporary anomalies as normative characteristics of the domain. This leads to regulating data quality problems rather than correcting them.

6.1.2. Post-SHACL Induction as a Normative Process

This work challenges the above assumption by asserting that constraint induction should be performed on repaired data. Therefore, the framework creates a fundamentally new epistemic foundation for reasoning by prioritizing data cleansing and SHACL-based validation above statistical constraint detection.

The validated graph is treated not merely as another snapshot of the data, but as a curated representation that reflects intended domain semantics rather than incidental artifacts. As a result, constraint induction operates over a population in which malformed literals, contradictory assignments, and implausible values have already been removed or repaired.

This reordering has direct implications for the nature of the discovered constraints. Post-SHACL constraint discovery captures what is consistent after semantic normalization, rather than what is clearly frequent in the data. Datatype regularities, cardinality bounds, and structural patterns converge toward stable semantic invariants rather than being weakened by outliers or systematic errors.

From a governance perspective, this reframes constraint discovery as a normative process. Constraints are no longer descriptive summaries of imperfect data as they become prescriptive rules that formalize the conditions under which data is considered acceptable going forward.

6.2. Impact on Recommendation Performance

The effects of semantic data cleaning and closed-loop governance are observable in downstream recommendation behavior. Table 6.1 summarizes the qualitative impact of data cleaning on recommendation performance.

Metric	**Noisy Data (Open-Loop)**	**Cleaned Data (Closed-Loop)**
Precision	Low (includes impossible or malformed nodes)	High (only semantically valid nodes)
Recall	Low (missed entities due to typos and duplicates)	High (normalized and consolidated entities)
Trustworthiness	Low (hallucinated or contradictory facts)	High (validated and repaired facts)
Governance	Weak (static or drifting constraints)	Strong (adaptive and evidence driven constraints)

Table 6.1: Impact of semantic data cleaning on recommendation quality

The closed-loop approach improves recommendation reliability by ensuring that candidate entities satisfy structural and semantic constraints before being consumed by downstream models. This leads to higher precision and recall while reducing the risk of misleading or logically inconsistent recommendations.

6.3. Applicability Beyond Recommendation Systems

Although this thesis evaluates semantic data quality primarily through recommendation scenarios, the proposed framework is generalized across a wide spectrum of knowledge graph applications.

- **Semantic Search.** Improved lexical robustness and redundancy elimination can directly enhance entity retrieval and ranking accuracy as it prevents result fragmentation and improves recall.
- **Question Answering (QA).** Temporal consistency and semantic coherence reduce the risk of spurious or logically contradictory answers, so they are less likely to suggest implausible facts.
- **Explainable AI (XAI).** Stable and well-governed constraints provide a reliable foundation for explanation generation. Closed-loop governance ensures that explanations are grounded in facts that conform to validated and current semantic rules.

Furthermore, the confidence scoring layer enables selective data consumption across applications. Systems can dynamically adjust trust thresholds depending on risk tolerance. To use only high-confidence entities for critical tasks while enabling exploratory analytics on lower-confidence data. This flexibility makes the framework particularly suitable for multipurpose and enterprise knowledge graph environments.

6.4. Human-in-the-Loop Governance

The proposed pipeline emphasizes automation, but it is not designed to eliminate human oversight. Instead, it repositions domain experts from manual rule authors to semantic supervisors who intervene only when automated discovery detects meaningful drift.

The hybrid SHACL update strategy explicitly supports this interaction by automatically applying low-risk evidence-backed updates. The high-impact or strengthening changes are flagged for expert review. This design aligns with modern data governance principles to automatically handle routine consistency enforcement while enabling humans to arbitrate semantic intent.

By reducing the operational burden of continuous manual schema maintenance. The framework allows experts to focus on higher-level modeling decisions rather than simple validation logic.

6.4.1. Dependency on Initial Constraint Quality

The pipeline reduces reliance on manually authored constraints over time; however, its behavior in early governance cycles is influenced by the quality of the initial SHACL model. If the initial constraints are overly permissive or misaligned with domain semantics, then the first validation and cleaning stages may fail to remove certain classes of noise.

This dependency is mitigated by the iterative nature of the closed-loop process. As constraint discovery operates over progressively cleaner graphs, the induced rules become more precise and informative. Nevertheless, the cold-start phase remains a potential weakness, especially in domains with ambiguous schema definitions.

Providing even a minimal set of high-confidence seed constraints can significantly improve convergence speed and stability during early iterations.

6.5. Threats to Validity

Despite its strengths, the proposed closed-loop semantic governance framework is subject to several threats to validity.

Internal validity may be influenced by the use of synthetic corruption and drift scenarios. Although these scenarios are designed to mimic common real-world data quality issues, they cannot fully capture the complexity and unpredictability of naturally evolving knowledge graphs.

External validity is constrained by the use of the IMDb dataset and its person-centric domain. Results may vary in domains with stricter ontological constraints, richer semantics, or different update dynamics.

Nevertheless, IMDB provides a realistic and widely recognized benchmark that supports meaningful evaluation of semantic quality governance under realistic conditions.

6.6. Scalability and Performance Considerations

The closed-loop governance pipeline incurs additional computational overhead from SHACL validation, statistical profiling, and constraint comparison. Looking at this limitation, it shows that this framework is intended for environments where semantic correctness is prioritized over immediate responsiveness. So, in domains with strictly controlled schemas, low data volatility, and minimal integration from heterogeneous sources, a closed-loop governance may outweigh its benefits.

However, this overhead is incurred during governance cycles rather than at query time, which makes the approach well-suited for batch-oriented or periodic validation workflows. Moreover, post-SHACL constraint discovery operates on a reduced and normalized graph that results to significantly lowering profiling costs compared to open-loop induction over raw data. The modular design of the pipeline supports parallel execution across graph partitions, enabling the framework to scale to large, continuously evolving knowledge graphs.

6.7. Design Trade-offs and Governance Balance

The proposed framework reflects a deliberate set of design trade-offs between automation, stability, and semantic control. Fully automated constraint induction systems prioritize responsiveness but risk encoding transient noise as schema rules. Conversely, strictly manual governance ensures semantic precision but does not scale to large or rapidly evolving knowledge graphs.

The closed-loop hybrid strategy adopted in this work lies in the middle ground. By automatically applying low-risk updates while deferring potentially breaking changes for review. Therefore, the framework sacrifices maximal automation in favor of controlled evolution. This choice reduces the likelihood of unintended schema degradation at the cost of occasional human intervention.

Importantly, this trade-off is intentional rather than incidental. The framework is designed for environments where semantic correctness and long-term trust outweigh the benefits of fully autonomous adaptation.

6.8. Partial Provenance and Constraint Evolution

The designed architecture embraces implicit provenance at the entity level through a confidence score mechanism. To preserve a concise record of the semantic interventions applied to specific entities, each score accounts for prior SHACL violations, repair activities, and missing information. In this way, the confidence score quantifies the adjustment required for an entity to reach compliance, serving as a lightweight provenance signal.

Nonetheless, this provenance capability does not capture constraint changes across multiple governance iterations. Automatically generated SHACL constraints are regarded as the current legitimate validation model. This is achieved without maintaining a clear history of their origin, modification rationale, or previous versions, regardless of whether they are applied directly or reviewed and updated through human-in-the-loop interaction. Earlier constraint parameters and the empirical metadata data that drove their induction to convergence are discarded rather than maintained as the pipeline iterates.

This limitation becomes more pronounced over extended iterations of governance. When constraints are regularly adjusted in response to evolving data distributions, the framework lacks a means to explain why a particular constraint was introduced, reinforced, or rejected in previous rounds. As a result, although the system guarantees semantic consistency in the last iteration, it cannot support continuous monitoring of governance progress or analysis of constraint decisions.

The trade-off between past traceability and ease of use is reflected in this pipeline design. The framework eliminates extra storage and management complexity, but it comes with a cost of not keeping explicit provenance for constraint modifications, which weakens the transparency. Enhancing the system to include versioned SHACL shapes will overcome this restriction and allow for full transparency across iterative closed-loop governance cycles.

6.9. Prioritizing Data Validity over Logical Inference

The framework's design intentionally gives validation-oriented semantics precedence over explicit ontological logic. Semantic layers such as RDFS and OWL are primarily intended to enable inference, allowing systems to infer hidden truths from clearly expressed facts, as covered in Section 2.1.3. While this feature is helpful for reasoning tasks, it does not immediately resolve the challenge of determining whether data are trustworthy in practice.

The main cause of failure in dynamic knowledge graphs is the frequent availability of inconsistent and illogical data rather than the absence of implied facts. Even in cases where the underlying ontology appears logically valid, issues such as invalid literals, incorrect role assignments, temporal inconsistencies, and duplicate entities directly compromise the dependability of downstream applications. It is not possible to identify or fix these weaknesses using ontological reasoning alone.

Given that this pipeline governance process is centered on SHACL-based validation, it underscores the importance of data quality over logical deduction. Constraints specify what can be located within the graph rather than what can be deduced from it. This approach enables systematic identification and correction of semantic violations that reasoning methods may overlook.

This design decision also accounts for scalability. Comprehensive OWL reasoning over dynamic knowledge graphs is affected by data quality and is computationally costly. On the other hand, validation and profiling processes are integrated into regular governance iterations, making the architecture more resilient in real-world settings where rapid updates and data quality aren't guaranteed.

However, this approach has a limitation where higher-order logical conflicts and other fundamental logical inconsistencies that require ontological reasoning are not intended to be captured by the framework. Rather, it focuses on the categories of semantic faults that significantly affect functional consistency.

7 | Conclusions and Future Developments

The primary objective of this research is to design and practically evaluate a robust semantic quality management framework for RDF-based knowledge graphs that remains effective under data evolution and schema drift. Therefore in order to tackle this challenge a closed-loop governance approach is introduced that joins SHACL-driven validation, targeted data repair, confidence assessment, and automated constraint evolution within a unified pipeline.

A comprehensive literature review of existing open-loop and closed-loop approaches to semantic data validation was conducted prior to highlighting their limitations in dynamic knowledge graph environments. In particular, the research has revealed that typical pipelines tend to rely on static and manually defined constraints or induce rules directly from raw data. It often leads to constraint drift and the reinforcement of noise. These observations encouraged the invention of a pipeline in which semantic validation and cleaning precede constraint discovery.

The proposed framework centralizes SHACL-based validation and cleaning stages as fundamental to the pipeline. By implementing semantic coherence, temporal consistency, lexical robustness, and redundancy removal prior to any inductive process, the framework creates a semantically clean and normalized graph. This validated graph provides a solid foundation for automated constraint discovery, ensuring that derived rules represent consistent semantic patterns rather than random results arising from noisy data. Thus, constraint learning becomes a normative process rather than merely descriptive.

To demonstrate how semantic issues arise in practice and how they can be systematically identified, resolved, and controlled over time, the framework was evaluated using real-world IMDb data. It illustrates that the closed-loop design enhances robustness against shifting data distributions and preserves alignment between constraints and graph structure through a series of stress tests and drift scenarios. A scaled concept of trust is enabled by the implementation of a confidence score layer, which extends the framework and al-

lows downstream systems to distinguish between highly reliable and minimally acceptable entities.

Experimental findings indicate that downstream recommendation tasks are directly and favorably affected by semantic cleaning and adaptive governance. Precision, recall, and overall trustworthiness all improved, indicating that semantic data quality is an important consideration in real-world knowledge graph applications rather than merely a modeling issue. The framework was demonstrated to be useful not only for recommendation systems but also for other applications that depend on coherent, tightly controlled semantics.

Despite its advantages, the architecture has some drawbacks. The quality of the initial SHACL limitations shapes its early behavior and may affect the initial cycles of governance. Domains with poorly defined schemas still face challenges during the cold-start phase, even though repeated refinement helps reduce this dependency. Furthermore, results may differ in domains with more complex ontological structures or more stringent semantic requirements because the evaluation was carried out on a person-centric dataset.

Several directions for future work emerge from this thesis. Extending the framework to support incremental and streaming validation would improve its suitability for continuously evolving knowledge graphs by enabling validation and repair on incoming updates rather than full graph snapshots. Furthermore, combining rule-based cleaning with learning-based repair methods may help address ambiguous or context-dependent semantic errors that are challenging to fix solely with constraints. Including external ontologies or cross-graph alignment data could improve constraint discovery, providing a firmer semantic foundation in heterogeneous settings.

This thesis concludes by showing that a closed-loop governance approach enhances semantic quality assurance in evolving knowledge graphs. The proposed paradigm provides a sustainable method for preserving semantic integrity over time by tightly coupling validation, cleaning, confidence assessment, and constraint evolution. This study introduces a conceptual change in how semantic constraints are learnt, enforced, and regulated in RDF knowledge graphs.

Bibliography

[1] A. Cropper and S. Dumancic. Inductive logic programming at 30: A new introduction. *Artificial Intelligence*, 298:103487, 2021.

[2] R. Cyganiak, D. Wood, and M. Lanthaler. Rdf 1.2 concepts and abstract syntax. W3c recommendation, W3C, 2023.

[3] J. Doe and J. Smith. Qse and shactor: Incremental shape extraction for evolving rdf graphs. *Journal of Web Semantics*, 90:101–120, 2025.

[4] L. Ehrlinger and W. Wöß. Towards a definition of knowledge graphs. *Semantic Web*, 13(1):1–15, 2022.

[5] M. Färber, F. Bartscherer, C. Menne, and A. Rettinger. Linked data quality of knowledge graphs. *Semantic Web*, 12(2):249–274, 2021.

[6] L. Galárraga and F. Suchanek. Knowledge graph mining. In *Reasoning Web. Declarative Artificial Intelligence*, volume 12660 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2021.

[7] P. Hitzler, K. Janowicz, and S. Schlobach. Foundations of ontology engineering with owl. *Semantic Web*, 13(1):1–24, 2022.

[8] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, et al. Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37, 2021.

[9] E. Huaman, D. Kärle, et al. Knowledge graph validation. *arXiv preprint*, 2005.01389, 2020. URL `https://arxiv.org/abs/2005.01389`. Overview of methods and challenges in validating quality and correctness of knowledge graphs.

[10] X. Jiang, C. Xu, Y. Shen, X. Sun, L. Tang, S. Wang, Z. Chen, Y. Wang, and J. Guo. On the evolution of knowledge graphs: A survey and perspective. *arXiv preprint*, 2310.04835, 2023. URL `https://arxiv.org/abs/2310.04835`. Survey on dynamic and temporal knowledge graph evolution and maintenance issues.

[11] H. Knublauch and D. Kontokostas. Shapes constraint language (shacl). W3C Recommendation, 2017. URL `https://www.w3.org/TR/shacl/`. Accessed: 2026-01-18.

[12] D. Kontokostas and H. Knublauch. Shacl advanced features and validation of rdf graphs. *Semantic Web Journal*, 14(3):503–527, 2023.

[13] J. E. Labra Gayo and H. Rodriguez. Shaclearner: Learning shacl shapes from rdf data. *Journal of Web Semantics*, 70:100613, 2021.

[14] A. Lee and R. Kumar. Prism: Closed-loop constraint governance for knowledge graphs. *Semantic Web Journal*, 2025.

[15] M. Ortiz and J. Smith. Automatic shacl constraint generation for large knowledge graphs. In *Proceedings of the 21st International Semantic Web Conference (ISWC)*, pages 245–260. Springer, 2022.

[16] H. Paulheim. Knowledge graph quality: A survey. *Semantic Web*, 12(3):417–447, 2021.

[17] H. Paulheim. Knowledge graph refinement: Recent advances and challenges. *Semantic Web Journal*, 12(3):345–370, 2021.

[18] R. Zhang, B. D. Trisedya, M. Li, Y. Jiang, and J. Qi. A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. *VLDB Journal*, 31(5):1143–1168, 2022. doi: 10.1007/s00778-022-00747-z.

[19] R. Zhang et al. Knowledge graph embedding methods for entity alignment: experimental review. *Data Mining and Knowledge Discovery*, 37:2070–2137, 2023. doi: 10.1007/s10618-023-00941-9.

List of Figures

List of Tables

Acknowledgements

This thesis marks the end of a long journey fulfilled with challenges and achievements that have significantly contributed to my personal and intellectual development.

I would like to express my sincere gratitude to Prof. Cinzia Cappiello for introducing me to this topic specially when I had few alternatives. I am also grateful for her invaluable guidance, patience, and encouragement that she offered throughout this work.

I am here thanks to my family who have always been certainties in my life. They have never made me doubt myself or the choices I've made. Also I would like to thank them for the endless support from the first moment in Italy up to the last phase in my master's program.