UNIVERSITÀ DEGLI STUDI DI MILANO

DOTTORATO DI RICERCA IN INFORMATICA
XXXVIII CICLO

DIPARTIMENTO DI INFORMATICA
"GIOVANNI DEGLI ANTONI"

SETTORE SCIENTIFICO DISCIPLINARE INF/01

Tesi redatta con il contributo finanziario dell'Unione Europea
Next Generation EU e Enginerring Ingegneria Informatica S.p.A.

Data Governance Framework for ML-Based, Data-Intensive Distributed Systems

Tesi di Dottorato di Ricerca di:
Antongiacomo Polimeno

Matricola: R13812
ORCID: 0009-0003-1197-568X
CUP: G43C22002330004

Tutor:
Prof. Claudio Agostino Ardagna

Co-Tutor:
Prof. Marco Anisetti

Coordinatore del Dottorato:
Prof. Roberto Sassi

A.A. 2024/2025

Acknowledgements

Completing this doctoral journey would not have been possible without the support, guidance, and encouragement of many remarkable people.

First and foremost, I wish to express my deepest gratitude to my advisor, Claudio Agostino Ardagna, for his tireless guidance through the challenging landscape of academia. His mentorship has been invaluable throughout this journey. I am equally grateful to Marco Anisetti for his precious counsel and insights that have shaped my research path.

I owe special thanks to Chirine Ghedira-Guegan for her warm hospitality and for welcoming me as a member of her laboratory in Lyon. Her expert perspectives helped redirect and refine the trajectory of my research at crucial moments, for which I am profoundly grateful.

My sincere appreciation goes to Susanna Bonura, Salvatore Cipolla, Nicola Masi, and the entire team at ENG for their invaluable assistance and genuine kindness. You made me feel at home, and that meant more than you know.

To my colleagues at SESAR Lab, thank you for creating such a stimulating and supportive environment. I am particularly indebted to Nicola Bena for his guidance and inspiration, to Marco Luzzara for his incomparable support, both technical and beyond, and to Ruslan for keeping me on track with deadlines and examinations.

To Rachele, thank you for standing by me during the most difficult moments and for constantly reminding me of my worth. Your presence made all the difference.

Finally, to my family: thank you for being there from the very first steps of this long journey. Your unwavering support has been my foundation.

Abstract

The data processing landscape has been fundamentally transformed in recent decades, evolving from monolithic systems to complex distributed architectures where multiple services operated by different parties collaborate to deliver sophisticated analytics capabilities. Machine Learning (ML) and Internet of Things (IoT) technologies are increasingly integrated into these systems, enabling advanced data processing pipelines that span multiple organizational boundaries. Modern distributed data processing systems support critical operations across various domains, from healthcare analytics to financial services, making their trustworthiness and compliance with data protection regulations a concern.

This transformation has created significant challenges for data governance, particularly in balancing data quality preservation with privacy protection requirements. Current approaches suffer from fundamental limitations: existing governance frameworks focus primarily on static privacy configurations, neglecting the dynamic nature of distributed service environments and failing to optimize quality-privacy trade-offs systematically. Traditional access control models are inadequate for multi-dimensional optimization scenarios where data utility must be maximized while ensuring regulatory compliance. Despite increasing regulatory requirements and organizational needs for privacy-preserving data processing, there is a lack of governance frameworks that can systematically optimize quality-privacy trade-offs in modern distributed systems while maintaining computational tractability and industrial applicability.

In this thesis, we propose a data governance framework that addresses this impasse. Our framework operates at the service composition layer, providing policy-driven optimization for data processing pipelines built as compositions of distributed services. It implements a multi-dimensional approach that balances data quality preservation with privacy protection through systematic service selection and dynamic adaptation mechanisms. The framework integrates seamlessly with existing distributed data processing platforms and has been validated through industrial deployment and experimental evaluation.

Our contribution is manifold. First, we design and implement a policy-

driven service selection framework. The framework defines three complementary quality metrics that capture different aspects of data utility preservation: a quantitative metric based on weighted Jaccard coefficients (M_J), a qualitative metric using Jensen-Shannon divergence (M_{JSD}), and an entropy-based metric (M_H) that quantifies information content retention. Then, we develop sliding window heuristic algorithms that address the NP-hard problem of optimal service selection by employing moving optimization windows that balance local and global objectives. Moreover, we provide a methodology for integrating our governance framework within existing distributed data processing platforms, demonstrating practical deployment through successful integration with the ALIDA (Advanced Laboratory for Interactive Data Analytics) platform. To validate our framework and demonstrate its industrial applicability, we conduct experimental evaluation across multiple datasets and deployment scenarios, proving the framework's effectiveness in maintaining data quality while ensuring privacy compliance with acceptable operational overhead.

Contents

List of Figures

List of Tables

1

Introduction

Modern distributed data processing systems are characterized by heterogeneous service-oriented architectures where multiple autonomous organizations contribute computational resources, datasets, and analytical capabilities through federated infrastructures that span cloud and edge computing environments. The increasing adoption of these collaborative data ecosystems across industries has fundamentally transformed how organizations manage and extract value from their data assets, creating unprecedented opportunities for cross-domain analytics while simultaneously raising complex challenges regarding data governance, privacy preservation, and quality assurance.

These distributed service-based data pipelines enable organizations to access and process datasets that would otherwise remain isolated within organizational boundaries, facilitating data-driven decision making at unprecedented scales through dynamic composition of existing services. Data spaces transformed data processing into services, basic computing units composed and orchestrated by developers to create complex analytical pipelines in a fully declarative manner. Around the same time, cloud computing introduced the as-a-service paradigm, where data processing, analytical platforms, and even data infrastructure are completely outsourced to remote data centers and cloud providers, and accessed on demand.

However, this architectural evolution introduces a fundamental tension between maximizing data utility through data sharing and analytical processing, and maintaining data protection through privacy-preserving mechanisms that safeguard sensitive information throughout the entire data lifecy-

cle. The challenge becomes particularly acute in service-based data pipelines where data flows through multiple processing stages operated by different providers, each potentially implementing distinct privacy mechanisms that can cumulatively degrade data quality while offering varying levels of protection. Traditional approaches to this privacy-utility trade-off often rely on static configurations that apply uniform protection measures across entire datasets, failing to account for the dynamic nature of modern data processing workflows where optimal privacy-utility balance depends on contextual factors including data characteristics, processing requirements, regulatory constraints, and stakeholder policies.

This thesis addresses the fundamental challenge of dynamically optimizing service selection and composition in distributed data pipelines to achieve maximum data utility while ensuring appropriate data protection, developing a policy-driven framework that enables systematic, context-aware optimization of privacy-utility trade-offs throughout the execution of complex analytical workflows in heterogeneous service ecosystems. In the remainder of this chapter, we discuss the motivations of our thesis and present its outline.

1.1 Motivations

The last two decades have witnessed the widespread diffusion of distributed data processing systems, driven by the emergence of collaborative data sharing paradigms such as data spaces [16] and the evolution toward service-oriented ecosystems [63, 49]. This transformation has been motivated by increasing complexity of data processing requirements, the need for scalable analytics, and recognition that valuable insights often emerge from combining datasets across organizational boundaries. Data processing applications evolved from monolithic centralized systems to elastic multi-layer distributed systems, where orchestration platforms dynamically adapt service compositions based on observed data characteristics, quality requirements, and privacy constraints.

Modern distributed data processing systems are implemented as loose federations operated by different stakeholders within data spaces [16]. The practical implementation of data spaces, exemplified by the International

Data Spaces (IDS) initiative [60], demonstrates how architectural principles translate into operational frameworks through IDS Connectors, Metadata Brokers, and supporting services that enable trustworthy data sharing while maintaining data sovereignty. These systems drive data-centric transformation across sectors from healthcare and finance to smart cities, delivering increased quality and scope of analytical insights through large-scale collaborative data processing and machine learning models. However, they significantly impact the data governance landscape, raising concerns about privacy preservation, data quality degradation, and regulatory compliance across multiple jurisdictions.

Traditional approaches to data governance in distributed environments focus on static, uniform privacy protection that fails to account for dynamic data processing workflows. These approaches suffer from key limitations that can be characterized according to critical governance features: they lack comprehensive support for service-based pipeline operations in cloud-edge continuum environments (F1), where modern systems span multiple infrastructure layers; they evaluate data protection and quality independently rather than optimizing service selection to ensure both simultaneously (F2), leading to suboptimal trade-offs; they remain tightly coupled to specific data protection techniques (F3), limiting adaptability across diverse privacy-preserving methods; and they provide inadequate policy frameworks (F4) for systematic specification of governance requirements in heterogeneous service ecosystems. A detailed analysis of how existing approaches address these governance features is provided in Section 2.5. Moreover, governance is typically applied after pipeline development, with quality and privacy requirements neglected during service composition, leading to increased costs and suboptimal results.

1.1.1 Objectives

The objective of this thesis is to bridge the gap between current data governance practices and the requirements of modern distributed data processing systems, addressing the critical governance features identified in the state-of-the-art analysis (Section 2.5).

First, we aim to provide support for service-based pipeline operations

across cloud-edge continuum environments, enabling effective governance in distributed infrastructures that span multiple deployment paradigms. This requires frameworks that can operate seamlessly across hybrid environments without being constrained to specific deployment models.

Second, we face challenges in implementing quality-aware service selection that ensures data protection simultaneously, moving beyond approaches that treat quality and privacy as independent concerns. This translates to integrated optimization algorithms that balance both objectives during service composition and execution, rather than applying them sequentially or independently.

Third, we aim to develop framework-agnostic approaches to data protection that can work with diverse privacy-preserving techniques without being tied to specific methods. This enables adaptability to different regulatory requirements, organizational preferences, and technological constraints across various deployment scenarios.

Finally, we seek to provide effective policy frameworks that support systematic specification of governance requirements for heterogeneous service ecosystems. This requires policy models that can capture complex quality-privacy trade-offs, contextual requirements, and stakeholder relationships in distributed environments.

These objectives collectively aim to make quality-privacy optimization integral to service composition rather than an a posteriori activity, implementing a proactive, policy-driven approach that considers governance requirements throughout pipeline instantiation and execution.

1.2 Contribution of the Thesis

The contribution of this thesis is the definition of a data governance framework for modern distributed data processing applications that addresses the objectives in Section 1.1. The framework consists of a policy-driven service selection and composition approach that dynamically optimizes quality-privacy trade-offs, and is fully integrated within the data pipeline lifecycle, from template design and service discovery to instance optimization and execution monitoring.

The framework has been validated through experimental evaluation and

real-world application scenarios, including integration with an industrial platform for advanced data analytics. The framework enables practical deployment of quality-aware, privacy-preserving data processing pipelines in distributed service environments. In the remainder of this section, we discuss the thesis contributions in detail.

1.2.1 Policy-Driven Service Selection Framework

We define a policy-driven framework that enables systematic specification and enforcement of quality and privacy requirements in distributed data processing pipelines. This framework permits quality-privacy optimization while maintaining the flexibility needed for heterogeneous service ecosystems. This contribution can be detailed as follows.

Definition of policy-driven service composition. We define a policy model that captures both functional requirements (data transformations, processing capabilities) and non-functional requirements (quality metrics, privacy constraints) for service-based data pipelines. Our approach extends traditional attribute-based access control (ABAC) models to incorporate data quality considerations alongside privacy protection requirements.

Multi-dimensional optimization process. We define a service selection process that simultaneously optimizes multiple, often conflicting objectives including data quality preservation, privacy protection strength. The selection process operates on pipeline templates that specify the abstract structure of data processing workflows, enabling reusable pipeline definitions that can be instantiated with optimal service combinations based on current requirements and available services.

Quality-privacy trade-off management. We define systematic approaches for managing the fundamental tension between data utility maximization and privacy protection. This includes metrics that quantify both quality degradation and privacy protection strength, enabling objective evaluation of different service combinations.

1.2.2 Industrial Integration and Validation

Our data governance framework defines practical approaches for integration with existing data processing platforms and industrial systems. We demon-

strate the applicability of our approach through integration with real-world platforms and experimental validation. This contribution can be detailed as follows.

ALIDA platform integration. We integrate our framework with ALIDA [80], an industrial data analytics platform developed by Engineering Ingegneria Informatica. This integration demonstrates practical deployment of quality-privacy optimization in real-world scenarios through development of service catalogs, policy specification interfaces, and optimization algorithms that operate within existing industrial data processing infrastructures.

Experimental validation framework. We develop experimental evaluation approaches that assess the effectiveness of our quality-privacy optimization across multiple dimensions including optimization quality.

Performance and scalability analysis. We provide analysis of the computational and operational overhead introduced by our governance framework, demonstrating that quality-privacy optimization can be achieved with acceptable performance characteristics for practical deployment in distributed data processing environments.

1.3 Organization of the Thesis

This chapter presented the motivations and the main objectives of this thesis, and outlined its major contributions. The remaining chapters are structured as follows.

Chapter 2 presents the state of the art in data governance for distributed systems. It discusses the regulatory landscape affecting data processing systems, examines current approaches to data protection including anonymization techniques and access control mechanisms, analyzes data quality management frameworks and metrics, and reviews existing approaches for integrating data quality and privacy protection. It also outlines the challenges faced by existing techniques that guide our thesis contributions.

Chapter 3 describes our data governance enabled pipeline framework at the foundation of this thesis. This chapter introduces the policy-driven approach for service selection and presents the ACL-mediated pipeline implementation that enables systematic control over data access and processing within service compositions.

Chapter 4 presents our approach for maximizing data quality in distributed data processing pipelines. The chapter details the pipeline template architecture, the service selection optimization process, and the instantiation mechanisms that enable dynamic adaptation of service compositions based on quality and privacy requirements.

Chapter 5 describes our optimization algorithms and heuristics for efficient service selection in large-scale distributed environments. It presents the sliding-window optimization approach and analyzes the computational complexity and scalability characteristics of our quality-privacy optimization framework.

Chapter 6 shows experimental evaluation and application scenarios where our framework has been validated. The chapter includes performance analysis, scalability studies, and comparison with existing approaches for data governance in distributed systems, demonstrating the effectiveness of our policy-driven service selection approach.

Chapter 7 presents the integration of our framework with real-world industrial platforms, specifically the ALIDA platform for advanced data analytics. The chapter describes the practical deployment considerations, system architecture adaptations, and optimization algorithm implementations that enable industrial adoption of our governance framework.

Chapter 8 summarizes the contributions of this thesis and outlines future research directions for distributed data processing systems.

2

Related Work

2.1 Reference Architecture for Data Processing Systems

The evolution of data-intensive applications has fundamentally transformed how organizations approach data governance, moving from traditional centralized architectures toward distributed, service-oriented ecosystems [63, 49]. The transformation has been driven by the increasing complexity of data processing requirements, the need for scalable and flexible data analytics, and the emergence of collaborative data sharing paradigms such as *data spaces*[16]. This architectural shift enables organizations to break down complex data workflows into independent, manageable services that can be developed and maintained separately, facilitating horizontal scaling. Individual components expand based on demand rather than requiring system-wide resource allocation. In this context data pipelines are dynamically composing existing services to meet specific analytical requirements, improving the reusability.

A data space can be conceptualized through four interconnected layers that define its operational structure: i) *Business Layer*: Establishes the governance framework, defining policies, regulations, and business rules that govern data sharing and usage within the ecosystem. This layer encompasses compliance requirements, data sovereignty principles, and stakeholder agreements. ii) *Information Layer*: Ensures semantic coherence through shared ontologies, vocabularies, and metadata standards. This layer addresses data interoperability challenges by providing common data mod-

els and facilitating data lineage tracking across the ecosystem. iii) *System Layer*: Maps conceptual requirements onto concrete technological components, including data connectors, service brokers, identity management systems, and orchestration platforms that enable practical data sharing and processing. iv) *Process Layer*: Describes the workflows and data processing pipelines that transform raw data into actionable insights. This layer encompasses both standardized data processing patterns and customizable analytical workflows.

Within this layered architecture, the operational complexity of data spaces necessitates the coordination of diverse stakeholders, each with complementary roles in the ecosystem. The business and information layers establish the governance and semantic foundations that enable three primary categories of stakeholders to interact effectively: i) *Data Producers*: Organizations or entities that generate, collect, and provide access to datasets. These stakeholders maintain sovereignty over their data while participating in the broader ecosystem. ii) *Data Consumers*: Entities that require access to data for analytical, operational, or research purposes. These stakeholders must comply with data usage policies while maximizing the utility derived from accessed data. iii) *Service Developers*: Organizations creating services that process and analyze data from multiple sources, bridging producers and consumers.

The inherently heterogeneous and multi-provider nature of data spaces fundamentally distinguishes them from traditional centralized data processing environments, making service-based architectures a necessary solution for managing this complexity. In traditional scenarios, a single organization controls the entire data processing pipeline, enabling centralized governance and unified technical standards. However, data spaces by definition involve multiple independent organizations, each with distinct technical capabilities, security frameworks, and business models, creating an environment where distributed, interoperable services become the only viable approach to enable collaborative data processing.

This architectural necessity introduces several considerations that reshape how data systems must be designed and operated. Provider heterogeneity emerges as the primary challenge, as services are inevitably of-

fered by different organizations with varying technical capabilities, security frameworks, and business models. The heterogeneity necessitates dynamic service selection mechanisms, where the optimal composition of services for analytical tasks must be determined based on evolving factors such as data characteristics, quality requirements, and privacy constraints. Furthermore, the multi-provider foundation of data spaces intensifies privacy preservation challenges beyond those encountered in centralized systems. Protecting sensitive information while enabling collaborative analytics requires privacy-preserving techniques to be integrated throughout the entire distributed pipeline [62, 49]. This extends far beyond traditional access controls to encompass advanced approaches such as differential privacy, federated learning, and secure multi-party computation, ensuring that valuable insights can be extracted without compromising individual privacy or organizational confidentiality across the heterogeneous service ecosystem.

The practical implementation of data spaces, as exemplified by the International Data Spaces (IDS) initiative, demonstrates how these architectural principles translate into concrete operational frameworks. Real-world deployments showcase the critical infrastructure components that enable trustworthy data sharing between autonomous organizations. The core architecture centers on *IDS Connectors*, which serve as secure gateways that facilitate controlled data exchange while enforcing usage policies defined by data owners. These connectors integrate directly with organizational IT systems through dedicated applications, enabling seamless data provisioning and consumption while maintaining data sovereignty.

Figure 2.1 illustrates this connector-based architecture operates within a supporting ecosystem composed of several specialized services with distinct operational roles [60]. *Metadata Brokers* provide service discovery and data catalog capabilities through both registration and search mechanisms. Connectors register their self-descriptions and resource catalogs with brokers, enabling participants to locate relevant datasets and services through SPARQL-based queries or full-text search interfaces. These brokers maintain complete metadata graphs including connector descriptions, resource catalogs, contract offers, and representation details, facilitating service discovery across the data space ecosystem. *Clearing Houses* serve as transaction management and audit systems that handle logging of data exchanges, con-

tract agreement management, and billing processes. They provide critical capabilities for querying transaction histories, tracking data usage patterns through process identifiers, and ensuring compliance with contractual obligations and regulatory requirements. *Identity Providers* manage authentication and authorization across the federated environment through certificate-based security mechanisms; *Vocabulary Services* ensure semantic interoperability through standardized data models and ontologies; and *App Stores* facilitate the distribution and deployment of data processing applications.

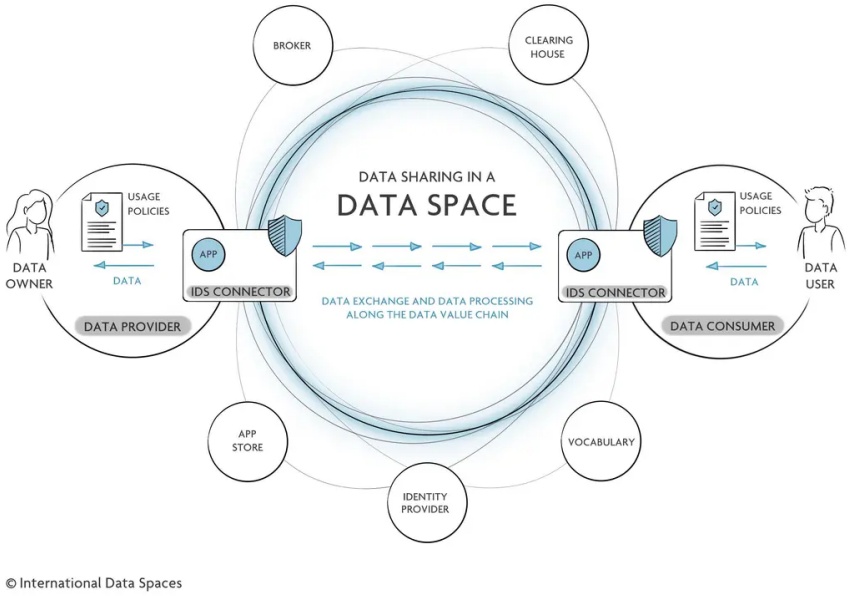


Figure 2.1: Data Space Conceptual Architecture [60]

The data governance framework developed in this thesis addresses the fundamental challenge of maximizing data quality while ensuring data protection in service-based data pipelines. The methodology centers on an optimization approach that dynamically selects and composes data processing services to achieve optimal quality-privacy trade-offs throughout the

pipeline execution. The framework aligns with the four-layer data space architecture previously outlined:

Business Layer The framework's policy-driven approach directly supports the business layer governance requirements. Some data protection annotations in the pipeline encode business rules and regulatory compliance requirements. On the other hand the service selection mechanism ensures adherence to data sovereignty principles and stakeholder agreements. The quality-privacy optimization addresses the business need to balance data utility with protection obligations.

Information Layer A functional annotations in pipeline contribute to semantic coherence by specifying service interfaces, expected inputs/outputs, and data transformation functions. This creates a foundation for maintaining data lineage tracking across service compositions, supporting the information layer's requirements for metadata standards and interoperability.

System Layer The service-based pipeline framework operationalizes the system layer by translating conceptual requirements into concrete technological components. Through its integrated service registry, policy enforcement mechanisms, and quality assessment modules, the framework delivers the essential infrastructure components required for practical data space implementation. The framework's sliding-window heuristic and optimization algorithms serve as the computational foundation that enables real-time dynamic service composition and automated orchestration across the distributed ecosystem.

Process Layer The pipeline instantiation process embodies the workflow management capabilities required by the process layer. The framework transforms abstract pipeline templates into executable instances, managing the complex dependencies and coordination required for multi-provider service compositions. The quality-aware selection process ensures that workflow execution optimizes for both functional requirements and governance constraints.

2.2 Regulation Landscape

The contemporary regulatory landscape of data protection and governance is characterised by an increasingly complex and multilayered architecture. The following section examines this landscape, focusing on major frameworks such as the GDPR, the EU Data Governance Act, and the CCPA, which collectively define the legal context within which technical solutions must operate.

The General Data Protection Regulation (GDPR) [34] constitutes the cornerstone of contemporary personal data protection law within the European Union and serves as a global reference point for privacy regulation. Adopted by the European Parliament and Council on 14 April 2016 and entering into force on 25 May 2018, the GDPR replaced the outdated 1995 Data Protection Directive to forge a harmonized legal framework for personal data protection across the EU and EEA. The GDPR have a dual mission: empowering individuals with robust rights over their personal information (including access, rectification, erasure, data portability, and objection) while obligating organizations to adopt principles of transparency, data minimization, security, and accountability in their data processing activities. As an EU regulation rather than a directive, it applies directly across member states without requiring national transposition, yet provides measured flexibility for local legal nuances. The regulation's territorial scope extends globally: any entity processing personal data of EU residents-regardless of geographical location-must comply with GDPR requirements. The GDPR's enforcement mechanisms feature independent supervisory authorities with administrative penalties reaching up to €20 million or 4% of global annual turnover, whichever is higher. These provisions have demonstrated practical impact through high-profile penalties on major technology companies, establishing the regulation's credibility beyond symbolic framework status.

The United Kingdom's Data Protection Act 2018 (DPA 2018) [95] operates in conjunction with the UK GDPR (the domestically retained version of the EU text) to provide a framework addressing national requirements, particularly law enforcement processing and public sector derogations. The DPA 2018 maintains the GDPR-style regulatory regime while incorporating UK-specific exemptions and enforcement mechanisms under

the UK Information Commissioner's Office. Although substantially aligned with the EU model, it incorporates domestic flexibilities for international data transfer adequacy determinations and regulatory oversight.

The EU Data Governance Act (DGA) [35] represents a complementary regulatory instrument designed to enhance trust and facilitate data sharing mechanisms within the single market, rather than fundamentally altering personal data rights. The Act focuses on enabling the reuse of specific categories of public sector data, regulating data intermediation services, and establishing data altruism frameworks to promote broader dataset sharing for research and innovation while maintaining appropriate legal safeguards. The DGA's emphasis on governance structures and marketplace mechanisms including registries, certified intermediaries, and data altruism frameworks-distinguishes it from the individual rights and core processing principles governed by the GDPR. The interaction between these instruments demonstrates a layered approach: the DGA establishes technical and institutional infrastructure for lawful data sharing, while privacy compliance remains governed by GDPR provisions and applicable national law.

The Digital Services Act (DSA) [36] addresses distinct regulatory challenges concerning the responsibilities of online intermediaries and platforms regarding content moderation, transparency obligations, advertising practices, and systemic risk management. While not primarily conceived as a privacy instrument, the DSA carries significant privacy implications through its requirements for large platforms to disclose algorithmic ranking methodologies, advertising targeting information, and risk mitigation measures that intersect with profiling and large-scale personal data processing activities. The DSA operates in parallel with privacy law, requiring platforms to satisfy DSA obligations while remaining bound by GDPR requirements for lawful processing and data subject rights. The enforcement architecture operates through designated Digital Services Coordinators and direct Commission oversight for very large platforms, creating a governance layer distinct from data protection authorities yet with overlapping practical implications.

The European Health Data Space (EHDS) [37] exemplifies sector-specific regulation that seeks to establish a coherent legal and technical framework for cross-border access and secondary use of electronic health data. The EHDS aims to enhance individual control over personal

health records for primary care purposes while enabling secondary uses for research, policymaking, and public health under strictly defined conditions. Given the particular sensitivity of health data, the EHDS combines health sector-specific provisions with existing GDPR protections, establishing health data access bodies, interoperability standards, and data permit schemes for secondary use.

Complementing these regulatory frameworks, technical infrastructure initiatives provide the operational foundation necessary for implementing such cross-sectoral data sharing requirements. **Gaia-X** constitutes a federated technical and governance initiative rather than a binding legal instrument. The project aims to construct a federated, interoperable data infrastructure emphasising data sovereignty, transparency, and federated control over services and metadata. Gaia-X provides architectural principles, certification criteria, and a community governance model intended to enable service providers and users to rely on interoperable, auditable services consistent with European values and regulatory requirements. Although Gaia-X does not substitute for legal frameworks, it provides technical infrastructure for regulatory compliance—enabling cross-border data sharing while reducing reliance on centralised platforms. The initiative bridges legal requirements, technical standards, and market competition.

These instruments illustrate three interconnected regulatory channels within the European ecosystem. The GDPR and national acts establish fundamental rights and obligations for personal data processing. The DGA constructs governance infrastructure for trustworthy data sharing. Sectoral regulations like the EHDS operationalise data reuse in specific domains with enhanced safeguards, while the DSA introduces platform governance with implications for profiling and algorithmic transparency. This creates a layered approach: foundational rights (GDPR/DPA), sectoral adaptations (EHDS), and horizontal governance instruments (DGA, DSA, Gaia-X) for data sharing and platform accountability.

This analysis reveals that the European regulatory framework has evolved into a sophisticated three-tier architecture where foundational data protection rights (GDPR/national acts), governance infrastructure mechanisms (DGA/Gaia-X), and sectoral applications (EHDS/DSA) operate through distinct yet coordinated enforcement paradigms. From enforcement perspec-

tives, these carry significant implications. The GDPR confers wide investigatory powers and substantial penalties upon independent regulators, with stringent international transfer requirements. The DGA introduces oversight through registries and certification for intermediaries, while the DSA establishes platform obligations and penalties for systemic risk management. The EHDS institutes sector-specific supervision requiring coordination between data protection and health authorities. Gaia-X relies on market adoption and certification rather than traditional enforcement. These create a regulatory environment with coexisting legal obligations, certification incentives, and supervised permit schemes. This layered approach attempts to balance fundamental rights protection with data sharing facilitation, though the practical effectiveness of this coordination across different enforcement mechanisms and sectoral authorities remains subject to ongoing evaluation and implementation challenges.

Beyond the EU, several regulatory systems merit consideration for comparative analysis, either reflecting aspects of EU law or presenting alternative regulatory models. **The California Consumer Privacy Act (CCPA) [89]** and its successor measures (CPRA) provide a US state-level framework granting consumer rights including access, deletion, and opt-out provisions while imposing obligations on businesses exceeding specified thresholds. Although narrower than the GDPR in certain respects, California law exercises significant extraterritorial effects for globally operating companies. **The Health Insurance Portability and Accountability Act (HIPAA) [94]** in the United States represents a sectoral statute focused on health information protection within healthcare contexts, emphasising covered entities and administrative, physical, and technical safeguards. **Brazil's Lei Geral de Proteção de Dados (LGPD) [40]** closely follows the GDPR model, incorporating a national regulator (ANPD) and comparable rights framework. **Canada's Personal Information Protection and Electronic Documents Act (PIPEDA) [78]** maintains a commercial activity-centric approach based on fair information practice principles. **China's Personal Information Protection Law (PIPL) [75]** sets formal privacy standards (e.g. explicit consent, data-localization and impact assessments) but remains rooted in a state-controlled enforcement framework that blends individual rights with national-security priorities, raising reasonable doubt about its effectiveness

in delivering genuine privacy protection. These non-EU regimes demonstrate two significant trends: convergence toward fundamental rights such as access and deletion, and divergence in territorial scope, sovereignty-driven controls, and enforcement models.

Table 2.1 provides a comparative analysis of the key regulatory instruments discussed above, highlighting their scope, territorial reach, and core obligations across both European and international jurisdictions.

Instrument	Scope	Territorial Reach	Core Rights/Obligations
European Regulations			
GDPR [34]	General personal data protection	EU + extraterritorial	Data-subject rights (access, rectification, erasure, portability), lawful bases
DPA 2018 (UK) [95]	UK-specific personal data law	UK domestic (with UK GDPR)	GDPR-style rights; law-enforcement exemptions
Data Governance Act (DGA) [35]	Re-use of protected public (and private) data	EU (also applies extraterritorially)	Conditions for data re-use, data-sharing intermediaries, data altruism
Digital Services Act (DSA) [36]	Platform governance (transparency, content)	EU (platforms operating)	Risk assessments, advertisement transparency, notice-and-action
European Health Data Space (EHDS) [37]	Health-sector data: primary and secondary use	EU cross-border	Patient control over health data; regulated reuse for research
International Regulations			
CCPA [89]	Consumer privacy rights	California + extraterritorial	Access, deletion, opt-out rights; business obligations for qualifying entities
HIPAA [94]	Health information protection	US healthcare entities	Administrative, physical, technical safeguards; covered entity obligations
LGPD [40]	General personal data protection	Brazil + limited extraterritorial	GDPR-style rights; national regulator (ANPD); comparable frameworks
PIPEDA [78]	Commercial personal information	Canada federal jurisdiction	Fair information practices; consent-based processing
PIPL [75]	Personal information protection	China + extraterritorial	Consent requirements, data localization, impact assessments

Table 2.1: Comparison of data protection and governance regulations across jurisdictions

2.3 Data Protection

This section examines the current state of the art in data protection across three dimensions. First, we investigate anonymization techniques, including k-anonymity, l-diversity, t-closeness, and differential privacy, which provide mathematical guarantees for releasing useful data while preventing individual re-identification. Second, we analyze access control mechanisms that govern authorized data interactions, encompassing traditional role-based systems and emerging privacy-aware models.

2.3.1 Data Anonymization

The increasing demand for sharing sensitive microdata for research, commercial, and public safety purposes, coupled with growing individual privacy concerns, has driven the development of data de-identification techniques [84, 90, 28, 32]. While the removal of explicit identifiers such as names or social security numbers is a basic first step, it is often insufficient to guarantee privacy [84, 90, 69]. Naive de-identification approaches are vulnerable to "linking attacks", where adversaries combine de-identified data with publicly available information (e.g., voter registries) to re-identify individuals through unique combinations of attributes like age, gender, and ZIP code [18, 84, 90, 69]. This section reviews classical data anonymization techniques designed to mitigate such risks, mechanisms, strengths, and limitations.

The concept of **k-anonymity**, proposed by Samarati [84, 85], emerged as a foundational solution to protect against linking attacks. A dataset is considered k-anonymized if each record is indistinguishable from at least $k-1$ other records within the dataset with respect to a set of "quasi-identifiers" . The quasi-identifiers are attributes whose values, when taken together, can potentially identify an individual (e.g., Zip-code, Birth-date, Gender). This property ensures that any individual associated with a released record cannot be uniquely identified with a probability greater than $1/k$ through linking attacks alone. The primary operations employed to achieve k-anonymity are generalization and suppression. Generalization involves replacing specific attribute values with more general, yet semantically consistent, ones (e.g., replacing a specific phone number with only its area code, or an age like '24'

with '20-29'). This is often achieved by defining generalization hierarchies for attribute domains. Suppression, on the other hand, involves deleting specific cell values or entire tuples from the dataset. Unlike other techniques that condense data or add noise, generalization and suppression preserve the truthfulness of the remaining information within a k-anonymized dataset [18, 28, 90].

Historically, early practical systems like Datafly [90, 84, 28] and μ-Argus [84, 28] employed greedy or iterative approaches to achieve k-anonymity. However, these heuristic methods often do not provide guarantees on the quality of the resulting anonymization and can lead to over-generalization or inadequate protection [18, 90, 84]. The problem of finding an optimal k-anonymization (i.e., one that perturbs the input dataset as little as necessary according to a given cost metric) is NP-hard [18, 32, 28, 67, 100]. Despite this, algorithms have been developed to better manage the computational challenges. Samarati proposed an algorithm exploiting binary search on domain generalization hierarchies to avoid exhaustive searches [84, 28]. Bayardo and Agrawal introduced an optimization algorithm that starts with a fully generalized dataset and systematically specializes it, employing cost-based pruning and dynamic search rearrangement to find optimal solutions for non-trivial datasets [18, 28]. While k-anonymity effectively protects against identity disclosure, it has been shown to be insufficient against attribute disclosure [67, 69, 102]. Two main types of attacks highlight these limitations: i) *Homogeneity Attack*: If all sensitive attribute values within a k-anonymous equivalence class are identical, an attacker can still infer the sensitive attribute of any individual in that class with 100% certainty, even without re-identifying the specific individual [67, 69]. For example, if all individuals in a group sharing the same quasi-identifiers are diagnosed with "heart disease", an attacker knowing someone belongs to that group can deduce their condition [67]. ii) *Background Knowledge Attack*: An attacker with external background knowledge about the distribution of sensitive attributes in the population can make strong inferences. For instance, if an attacker knows that an individual is in a specific equivalence class and also knows that this individual has a low risk for a common disease, they can narrow down the possible sensitive values, potentially compromising privacy [67, 69].

To address the shortcomings of k-anonymity, the notion of **l-diversity**

was introduced [69, 67, 102]. The core idea of *l*-diversity is to ensure that each equivalence class (defined by quasi-identifiers) contains at least *l* "well-represented" values for each sensitive attribute [69, 67]. This ensures that even if an attacker can identify a record as belonging to a particular equivalence class, they cannot infer the sensitive attribute with high confidence because there are at least *l* diverse sensitive values present [69]. Different interpretations of "well-represented" exist, including distinct values, entropy, or recursive definitions [69, 100]. Similar to *k*-anonymity, *l*-diversity is typically enforced using generalization and suppression techniques [69]. However, finding optimal *l*-diverse generalization is also NP-hard, even with a small number of distinct sensitive values [100].

Despite its improvements, *l*-diversity can still be vulnerable. For example, if the *l* diverse values are semantically close or if the overall distribution of the sensitive attribute in the dataset is skewed, an attacker might still make accurate inferences. To counter these issues, **t-closeness** was proposed [67, 102]. This privacy model requires that the distribution of a sensitive attribute within any equivalence class is "close" to the distribution of that attribute in the overall original table. The "closeness" is measured using a metric such as the Earth Mover Distance (EMD), which considers the semantic closeness of attribute values. By ensuring similar distributions, *t*-closeness prevents an attacker from gaining significant information about an individual's sensitive attribute by merely identifying their equivalence class. Generalization and suppression remain the primary mechanisms for achieving *t*-closeness [67].

The limitations of *k*-anonymity, *l*-diversity, and *t*-closeness, particularly their vulnerability to attribute disclosure and reliance on specific adversary models, led to the development of **differential privacy (DP)**, a more robust privacy paradigm introduced by Dwork around 2006 [33]. Differential Privacy provides mathematically provable privacy guarantees, in contrast to earlier heuristic-based approaches, by ensuring that the results of any data analysis remain nearly unchanged regardless of whether an individual's data is included in the dataset [54]. This strong guarantee implies that an individual incurs virtually no additional risk by having his/her data included in a differentially private analysis. Formally, a randomized mechanism M provides ϵ-differential privacy if, for any two neighboring datasets D_1 and D_2 differing by at most one record, and for any set of possible outputs S:

$Pr[M(D_1) \in S] \leq e^{\epsilon} \cdot Pr[M(D_2) \in S]$ [33]. A more general definition, (ϵ, δ)-differential privacy, allows for a small probability δ of a privacy breach, where $Pr[M(D) \in S] \leq e^{\epsilon} \cdot Pr[M(D') \in S] + \delta$ [83]. The parameter ϵ (privacy budget) quantifies the trade-off between privacy loss and data utility, with smaller values providing stronger privacy but potentially lower utility [83]. DP is primarily achieved through controlled addition of random noise to query results or data, calibrated based on the query function's sensitivity (Δf), which represents the maximum possible change in the query's output when a single record is added or removed from the dataset [54]. Common noise addition mechanisms include the Laplace mechanism for numerical query outputs, adding noise drawn from a Laplace distribution scaled by the sensitivity and ϵ; the exponential mechanism designed for non-numerical outputs, selecting an output with probability exponentially proportional to its utility score and inversely related to ϵ and sensitivity; and the Gaussian mechanism, which adds noise from a Gaussian distribution similarly scaled by sensitivity and ϵ [54, 108]. Local Differential Privacy (LDP) represents a significant variant where individuals perturb their own raw data before transmission to untrusted data aggregators, shifting privacy protection from trusted central curators to individuals using mechanisms like Randomized Response [102]. The strengths of DP include strong, provable privacy guarantees that hold regardless of adversary background knowledge, preservation of data truthfulness without modifying original records, composition theorems allowing privacy guarantees to be maintained across multiple queries, and relatively lightweight computational costs [54, 102]. However, DP faces challenges including the non-trivial task of determining optimal ϵ values, potential significant utility degradation when high query sensitivity necessitates substantial noise, complexity limitations for datasets that are not "internet scale", vulnerability to correlation-based attacks in real-world data despite DP protection, and susceptibility to specific attacks such as tracker or data reconstruction attacks [54, 83]. For LDP specifically, additional challenges include higher sample complexity, lower accuracy compared to centralized DP, and potential communication overhead [102].

While these classical data anonymization techniques have provided foundational frameworks for privacy preservation, they face significant chal-

lenges in the context of modern service-based big data and machine learning environments. The NP-hardness of optimal anonymization becomes increasingly intractable with the high dimensionality and vast volume of "big datasets", leading to substantial computational overhead and potentially unacceptably high information loss [32, 54, 100]. Furthermore, ensuring privacy for continuously generated, real-time data, such as from wearable health devices, is difficult due to the dynamic nature and inherent temporal correlations within the data. The static nature of many classical methods struggles to adapt to these evolving data streams. The utility-privacy trade-off is particularly challenging for machine learning applications, where models require high-quality, granular data for effective training, but anonymization can degrade data utility, impacting model accuracy and performance [83, 54, 33]. These limitations have spurred the development of more advanced privacy-preserving techniques, notably differential privacy, which offers stronger, provable privacy guarantees independent of an adversary's background knowledge [33, 54, 83, 102].

2.3.2 Access Control

Access Control (AC) systems are paramount for ensuring data security and privacy in a data-driven world. The exponential growth of data, the proliferation of cyber threats, and the shift to remote work underscore the critical need for robust AC mechanisms. Effective AC regulates how users and systems interact with data and resources, granting or denying access based on defined policies and user privilege levels. Research in AC has seen a significant upward trend over the past decade, driven by technological advancements and heightened security and privacy concerns across diverse domains [39].

Traditional Access Control Models (ACMs) form the foundational backbone of security systems, each offering distinct advantages and limitations. These models include *Mandatory Access Control (MAC)*, *Discretionary Access Control (DAC)*, *Role-Based Access Control (RBAC)*, *Rule-Based Access Control (RuBAC)*, *Organization-Based Access Control (OrBAC)*, *Team-Based Access Control (TmBAC)*, *Task-Based Access Control (TBAC)*, *Usage-Based Access Control (UCON)*, *View-Based Access Control (VBAC)*, and *Behavior-Based Access*

Control (BBAC) [39]. *Mandatory Access Control (MAC)* assigns security labels to users and resources, with administrators defining policies. Access is allowed only if a subject's clearance meets or exceeds an object's classification, making it suitable for high-security contexts such as military or government systems [39]. **Discretionary Access Control (DAC)**, in constrast, lets resource owners define and grant permissions. While flexible, it can create complex and insecure access patterns in large systems [26, 39]. **Role-Based Access Control (RBAC)** assigns permissions to roles rather than individuals, simplifying management and improving consistency in large organizations. Widely adopted for its balance of security and usability, it includes variants such as ARBAC for role administration [86, 26, 39]. **Attribute-Based Access Control (ABAC)** has emerged as a highly flexible model, granting permissions based on a set of conditions considering user attributes, object attributes, and environmental attributes [39, 27]. Its flexibility makes it particularly suitable for multi-user distributed environments and Big Data systems [51, 56]. Other models in literature include **Rule-Based Access Control (RuBAC)**, which leverages predefined rules [39]; **Organization-Based Access Control (OrBAC)**, focusing on an organization's structure [64, 39]; **Team-Based Access Control (TmBAC)**, designed for collaborative environments [93, 39]; **Task-Based Access Control (TBAC)**, where access is tied to specific tasks [39]; **Usage-Based Access Control (UCON)**, which offers continuous authorization monitoring [39]; **View-Based Access Control (VBAC)**, which defines access through database views [39]; and **Behavior-Based Access Control (BBAC)**, dynamically adjusting permissions based on user behavior [105, 39].

But Modern digital environments, characterized by large-scale distributed systems, streaming data, and complex interactions, demand more sophisticated and adaptable AC solutions than traditional models alone can provide [39, 7]. Many approaches for Big Data AC leverage Attribute-Based Access Control (ABAC) due to its flexibility in such dynamic, multi-tenant environments [7, 27, 51, 56]. These systems often integrate ABAC with other models or mechanisms. For instance, the HBD-Authority model for **Hadoop** combines ABAC with context support, leveraging streaming data analytics for distributed parallel processing to manage security policies efficiently and provide dynamic data protection [27]. Other solutions for Hadoop in-

clude object-tagged RBAC (OT-RBAC) and more generalized ABAC models (HeABAC) that introduce concepts like cross-Hadoop service trust [52, 51].

An important aspect of Big Data AC is the enforcement methodology. Query rewriting is a prominent technique used to ensure that only authorized data is released [5, 27, 26]. This solution involves modifying a user's query to align with defined access policies, effectively creating an "authorized view" of the data [27]. This approach is often contrasted using the *Truman model* (transparently modifying unauthorized queries, which can lead to misleading results) and the *Non-Truman* model (requiring original and modified queries to yield equivalent results, which can be computationally intensive) [82, 65, 27]. Hybrid approaches, like that in HBD-Authority, combine the benefits of both to provide transparent enforcement without interfering with query semantics or execution plan, while validating results for correctness, security, and completeness [27].

Building upon these foundational approaches, **NoSQL** databases exemplify the broader transition toward fine-grained access control, where specialized ABAC mechanisms are being developed to address the inherent limitations of coarse-grained RBAC permissions [56]. The **Attribute-Based Fine-Grained Access Control (AGAC)** mechanism proposed for **HBase** demonstrates this evolution, providing coverage across five granularity levels-global database, namespace, table, column family, and column-while supporting atomic operations and policy inheritance [56]. Similarly, **Apache Spark** environments are witnessing the investigation of the **purpose-aware access control (PAAC)** models, which extend conventional purpose-based access control by incorporating data processing and operation purposes. These systems enable automatic recognition of data processing intentions from analytics operations and queries, facilitating fine-grained access decisions directly within Spark's Catalyst optimizer [103].

The decentralized nature of **Web-based Social Networks** presents critical challenges that further underscore the inadequacy of traditional centralized access control systems in modern scenario [23, 39]. The emphasis on relationships and trust among users necessitates architectures where each participant can specify and enforce their own policies rather than relying on centralized authorities [23]. This requirement has catalyzed the development of the **collaborative multi-party access control** models, which enable all

users associated with a resource to participate collectively in specifying its access control policy. These innovative systems employ threshold-based secret sharing schemes and leverage existing social relationships to delegate partial enforcement responsibilities to trusted participants, thereby ensuring privacy preservation without complete dependence on service providers [58].

Data stream processing environments introduce temporal and computational efficiency considerations that demand specialized access control approaches. The predominant strategy involves query rewriting mechanisms that proactively modify user queries to prevent unauthorized tuple or attribute retrieval [21, 22]. This approach represents a significant improvement over post-processing methodologies, which inefficiently prune unauthorized data after query execution, resulting in substantial computational waste [68, 21, 22]. The effectiveness of query rewriting for data streams relies fundamentally on the design of secure operators including Secure Read, Secure View, Secure Join, and Secure Aggregate-that filter unauthorized data from non-secure counterparts during query definition rather than at runtime, thereby eliminating execution overhead [24, 22]. Contemporary frameworks further enhance this approach by maintaining independence from specific stream engines through abstract core query models and deployment modules that translate rewritten queries across different stream languages [22].

Graph databases represent another frontier where access control complexity is driving innovation, particularly through the application of RBAC principles. These systems must effectively manage sensitive data with varying clearance levels while maintaining efficient query processing capabilities, especially for aggregation operations within distributed environments. Consistent with developments in relational and Big Data contexts, graph database access control employs query rewriting techniques to restrict access to specific distributed storage components, ensuring that query responses utilize only data compatible with users' clearance levels and contextual requirements [26].

Despite significant advancements, AC systems face key challenges including performance overhead in large-scale systems and usability issues where complex policies lead to misconfigurations and security risks for non-

experts [39, 92]. Future research focuses on unified frameworks combining traditional and emerging models, actionable guidelines for policy management [39], advanced query rewriting for complex operations [5, 27], handling malformed policies and integrating inference control [22], privacy-preserving path discovery in social networks [58], and improving AC mechanisms for NoSQL systems [56].

2.4 Data Quality

The increasing generation of large volumes of data has paralleled a growing awareness among businesses, medical practitioners, and researchers of the intrinsic value of data and the potential to extract ever greater insights from them. This trend is further facilitated by service-based pipelines, which enable the efficient processing of these big data. One functionality that is increasingly integrated into these pipelines and is particularly sensitive to data quality is machine learning. As a domain of particular interest, machine learning plays a central role in enabling the development of accurate and reliable models, but it is fundamentally dependent on high-quality input data. This dependency stems from the inherent nature of ML algorithms, which learn patterns directly from training data and amplify any underlying quality issues through their predictive outputs. High-quality data serve as a prerequisite for informed decision-making, service improvement, risk prediction, and reliability of ML models. Consequently, data quality management has become not just a technical consideration but a critical business imperative, as poor data quality can cascade ultimately affecting the value extracted from data across all operational contexts [47, 53, 106].

Despite its critical importance, achieving high-quality data remains challenging. Datasets often contain inconsistencies, inaccuracies, incompleteness, outdated information, and duplication, leading to significant economic and technical consequences [20, 38]. Poor data quality directly affects analysis and machine learning processes, producing unreliable or biased conclusions and predictions, as evidenced by labeling errors in major AI benchmarks [47, 53]. However data quality assessment is inherently complex due to its stratified, multidimensional, and contextual nature. Data is considered high-quality if it is “fit for its intended uses in operations, decision mak-

ing, and planning" or if it accurately represents the real-world constructs to which it refers [99, 20, 47, 53, 104, 91]. This "fitness for use" perspective emphasizes that data quality cannot be evaluated in isolation but must be assessed relative to specific contexts and applications. Consequently, data quality measurement is distributed across multiple dimensions, each representing different aspects of how well data serve its intended purpose. These dimensions do not always indicate simplicity in measurement; therefore in literature often each dimension is associated with specific metrics and data curation steps necessary to address the respective measures. The dimensions presented in this review are collected from the literature and ordered by frequency of appearance.

2.4.1 Data Quality Dimensions and Metrics

Data quality is widely recognized as a multidimensional concept, with various frameworks and surveys proposing different dimensions and groupings tailored to specific contexts and applications [47, 53, 20, 104, 13, 87, 106]. The multidimensional nature reflects the complexity of modern data ecosystems, where data serve diverse purposes across heterogeneous applications, from traditional business intelligence to advanced machine learning pipelines. While there are differences across fields and individuals regarding the most suitable dimensions, a convergent set of characteristics frequently emerges across the literature, suggesting fundamental aspects that transcend domain-specific requirements.

The assessment of data quality involves measuring these dimensions through both quantitative and qualitative approaches [91, 109]. For large-scale ML/DL systems, quantitative approaches are generally prioritized due to their formal, objective, and systematic nature, providing numerical indicators that facilitate automated assessment and continuous monitoring [53, 20]. However, the selection and weighting of dimensions must be contextually appropriate, as different applications may emphasize distinct quality aspects—for instance, real-time systems prioritize timeliness, while regulatory compliance scenarios emphasize accuracy and auditability.

The following presents the fundamental data quality dimensions alongside their corresponding measurement metrics, organized by their preva-

lence in contemporary literature and practical importance in data-intensive systems:

Accuracy: This dimension quantifies the degree to which data correctly represents the true values or real-world entities to which it refers [99, 47, 53, 20, 104, 91, 87, 106]. Accuracy encompasses both syntactic correctness (adherence to format specifications) and semantic correctness (faithful representation of real-world phenomena). In machine learning contexts, accuracy extends to label correctness, feature representation fidelity, and ground truth alignment [47].

Metrics: Content correctness ratio (percentage of accurate values), syntactic validation scores, semantic consistency measures, label accuracy for supervised learning, cross-validation with authoritative sources, and error detection rates. Advanced metrics include statistical measures like mean absolute error for numerical data and edit distance for categorical data [47, 91].

Completeness: This dimension quantifies the extent to which data contains values for all expected attributes and entities, indicating the comprehensiveness of information capture [99, 47, 53, 20, 104, 91, 87, 106]. Completeness operates at multiple granularities: attribute-level (presence of expected fields), record-level (existence of complete entities), and population-level (coverage of the intended domain). For structured data, it includes schema completeness, while for linked data it encompasses interlinking completeness [104].

Metrics: Null value frequency, missing attribute ratio, record completeness percentage, population coverage metrics, schema compliance rates, and data density measures. For time-series data, temporal completeness and sampling rate adequacy are critical indicators [47, 91].

Timeliness/Currency: This temporal dimension quantifies the freshness and temporal relevance of data relative to its intended use, ensuring information reflects the current state of represented entities [99, 47, 53, 20, 104, 91, 87, 106]. Timeliness is context-dependent—financial

market data requires near real-time currency, while demographic data may remain valid for years. The dimension encompasses both absolute timeliness (time since creation/update) and relative timeliness (appropriateness for specific analytical tasks).

Metrics: Data age calculations (time since last update), update frequency adherence, staleness ratios, temporal validity windows, real-time ingestion latencies, and context-specific currency thresholds. For streaming data, metrics include processing delays and temporal ordering violations [47, 91].

Consistency: This dimension ensures data maintains uniformity and coherence across different representations, storage locations, and time periods, eliminating contradictions that could compromise analytical reliability [47, 20, 104, 91, 87, 106]. Consistency operates at multiple levels: syntactic (format standardization), semantic (meaning preservation), and temporal (state coherence over time). For distributed systems, consistency includes cross-system data synchronization and referential integrity maintenance [104].

Metrics: Format standardization compliance, referential integrity violations, duplicate detection rates, cross-system synchronization errors, temporal consistency checks, and logical contradiction identification. Advanced metrics include entropy-based consistency measures and graph-based coherence analysis for linked data [104].

Uniqueness/Deduplication: This dimension quantifies the absence of redundant records within datasets, addressing a critical challenge in big data environments where multiple sources and ingestion processes can introduce duplicates [47, 53, 104, 38, 101]. Uniqueness operates at entity-level (distinct real-world objects) and attribute-level (unique identifiers). The challenge intensifies with fuzzy duplicates, where records represent the same entity but contain variations in representation, formatting, or completeness.

Metrics: Duplicate detection ratios, entity resolution accuracy, fuzzy matching confidence scores, deduplication effectiveness measures, and unique identifier coverage. Advanced metrics include clustering-based

similarity scores and probabilistic duplicate identification confidence intervals [47, 91].

Validity: This dimension ensures data conforms to predefined constraints, formats, and business rules that define acceptable values and structures [47, 53, 104, 101]. Validity encompasses syntactic validation (adherence to data types and formats), semantic validation (compliance with domain-specific rules), and constraint validation (satisfaction of integrity constraints). For machine learning applications, validity extends to feature distribution compliance and target variable integrity.

Metrics: Format compliance rates, constraint violation frequencies, data type adherence percentages, range validation scores, pattern matching success rates, and business rule compliance measures. Statistical metrics include distribution conformity tests and outlier detection sensitivity [47, 91].

Unbiasedness/Fairness: This dimension quantifies the representational balance and demographic equity within datasets, particularly crucial for machine learning applications where biased training data can perpetuate or amplify societal inequities [47, 53, 87]. Unbiasedness encompasses statistical fairness (balanced feature distributions), demographic parity (equitable representation across groups), and outcome fairness (unbiased prediction patterns). The dimension extends beyond simple class balance to include intersectional fairness and historical bias correction.

Metrics: Class imbalance ratios, demographic parity measures, statistical parity differentials, equalized odds ratios, calibration fairness scores, and intersectional representation indices. Advanced metrics include algorithmic fairness indicators and bias amplification measurements across protected attributes [53, 96, 106].

Accessibility: This dimension encompasses the technical and organizational ease with which authorized users can discover, retrieve, and utilize data resources within appropriate security and privacy constraints [99, 47, 20, 104, 106]. Accessibility includes discoverability (metadata richness), retrievability (system availability), and usability (interface

effectiveness). For distributed systems, accessibility encompasses federation capabilities and cross-platform interoperability.

Metrics: System availability percentages, data retrieval response times, authentication success rates, metadata completeness scores, API performance indicators, and user satisfaction metrics. Technical metrics include bandwidth utilization efficiency and concurrent access handling capacity.

Representational Quality: This dimension quantifies how effectively data is structured, formatted, and presented to facilitate understanding and utilization by both human analysts and automated systems [99, 20, 104, 106]. Representational quality encompasses interpretability (clarity of meaning), conciseness (efficient representation without redundancy), and consistent formatting (standardized presentation). For machine learning contexts, this includes feature engineering quality and data encoding appropriateness.

Metrics: Schema consistency scores, documentation completeness ratios, metadata richness indicators, format standardization compliance, readability assessments, and semantic clarity measures. Technical metrics include serialization efficiency and cross-platform compatibility indicators.

Trustworthiness/Credibility: This dimension assesses the reliability and authoritativeness of data sources, incorporating both technical reliability (system stability) and content credibility (source reputation and verification) [47, 53, 104, 106, 109]. Trustworthiness encompasses provenance transparency (clear data lineage), source verification (authoritative origins), and temporal stability (consistent quality over time). In federated systems, this includes cross-institutional trust metrics and reputation-based quality indicators.

Metrics: Source reputation scores, provenance completeness indicators, verification success rates, cross-validation consistency measures, temporal reliability assessments, and institutional credibility ratings. Advanced metrics include blockchain-based provenance verification and distributed trust consensus measures.

Standardization: This dimension measures adherence to established industry standards, organizational conventions, and technical specifications that enable interoperability and consistent interpretation across systems and stakeholders [47, 87]. Standardization encompasses format standards (file types, encoding), semantic standards (ontologies, vocabularies), and quality standards (ISO frameworks). For data integration scenarios, standardization facilitates seamless information exchange and reduces transformation overhead.

Metrics: Standards compliance percentages, format adherence rates, ontology alignment scores, vocabulary consistency measures, and interoperability success rates. Technical metrics include transformation effort requirements and cross-system compatibility assessments.

Ease of Use: This dimension quantifies the and technical effort required for users to effectively discover, access, understand, and utilize data for their intended purposes [47]. Ease of use encompasses user interface quality, documentation adequacy, learning curve steepness, and tool integration effectiveness. For data science workflows, this includes API usability and programmatic access efficiency.

Metrics: User task completion times, learning curve measurements, documentation utilization rates, error frequency during usage, user satisfaction scores, and tool integration success indicators. Behavioral metrics include user retention rates and feature adoption patterns.

Confidentiality: This dimension ensures appropriate protection of sensitive information against unauthorized access, disclosure, or misuse while maintaining data utility for legitimate purposes [47]. Confidentiality encompasses access controls (authorization mechanisms), data protection (encryption and anonymization), and privacy preservation (compliance with regulations like GDPR). The dimension balances security requirements with analytical needs through techniques like differential privacy and secure multiparty computation.

Metrics: Encryption coverage percentages, access control effectiveness rates, privacy breach incident frequencies, anonymization success scores, differential privacy budget utilization, and regulatory com-

pliance assessment results. Advanced metrics include privacy-utility trade-off measurements and re-identification risk assessments.

Beyond direct measurements, the performance of machine learning models is often used as an indirect measure of data quality. For example, prediction accuracy of a classifier built from a dataset can indicate the quality of information discoverable within that dataset [42, 50].

Qualitative methods for data quality evaluation include expert reviews, interviews, and field observations. These methods rely on the subjective judgment of subject experts or professionals, especially for aspects that are difficult to quantify, such as user perception of quality. However, their application to large datasets, particularly in ML/DL systems, is often impractical [53, 104].

Beyond these individual dimensions, overarching categorical frameworks have been proposed. Wang and Strong's framework [99] groups dimensions into *Intrinsic DQ* (accuracy, objectivity, believability, reputation), *Contextual DQ* (value-added, relevancy, timeliness, completeness, appropriate amount), *Representational DQ* (interpretability, ease of understanding, consistency, concise representation), and *Accessibility DQ* (accessibility and access security) [99, 53, 106]. Cai and Zhu [20] identified five big data dimensions: *Availability, Usability, Reliability, Relevance*, and *Presentation Quality*. Gong et al. [47] proposed eight ML quality dimensions: *completeness, self-consistency, timeliness, confidentiality, accuracy, standardization, unbiasedness, and ease of use*. Chen et al. [53] suggested five critical ML dimensions: *comprehensiveness, correctness, variety, class imbalance, and duplication* [53]. For medical AI, the METRIC-framework proposes clusters like *measurement process, timeliness, representativeness, informativeness*, and *consistency* as awareness dimensions [87]. The choice and weighting of these dimensions often depend on the specific application scenario and user requirements [47, 53, 106].

Several frameworks and models are employed to assess data quality systematically:

Lifecycle-based Evaluation: This approach emphasizes analyzing data quality metrics at each stage of the data lifecycle, from data generation/collection to storage, processing, analysis, and visualization. This

ensures continuous quality management and assessment [47, 50, 53, 91, 106].

Systematic Evaluation and Improvement Frameworks: A comprehensive framework includes defining domain-specific ML requirements, constructing datasets (reusing, improving, or creating with semi-automatic approaches), evaluating data quality against predefined dimensions, improving quality through various techniques, developing appropriate ML models, evaluating model performance (as an indirect measure of data quality), and continuous monitoring with feedback mechanisms [53].

Cai and Zhu's Dynamic Assessment Process: This framework focuses on determining data collection goals, data cleaning, applying qualitative and quantitative assessment methods, and integrating a dynamic feedback mechanism to meet big data quality assessment needs [20].

Chen et al.'s ML-specific Framework: This research defines data quality as fit for the purpose of building a machine learning system and uses selected dimensions like comprehensiveness, correctness, variety, class imbalance, and duplication, evaluating them through experiments [53].

Big Data Quality Profile (DQP): Proposed by Taleb et al. [91], the DQP captures the quality outline, requirements, attributes, dimensions, scores, and rules for continuous quality management in big data environments. This profile is incrementally updated across the data lifecycle, providing a record of quality status and rules.

Data Quality Tools: Various tools facilitate data quality evaluation and improvement by offering functions for data profiling, issue detection, cleaning, transformation, and monitoring. These tools increasingly integrate AI, including Large Language Models (LLMs) and generative AI, to automate tasks such as data generation and anomaly detection [106].

2.4.2 Data Curation

Data curation is defined as the active management process from the point of data creation to extract value from it, encompassing tasks to clean and enrich data to ensure its fitness for user requirements. This process is important for improving ML model performance, especially within the complex and dynamic environments of big data and AI pipelines [47, 53, 106, 109].

Key methods and processes for improving data quality through curation include:

Data Cleaning (Data Scrubbing): This involves detecting and removing errors and inconsistencies from data to improve its quality [20, 38, 47, 50, 101, 91]. Common data quality issues addressed include duplicate data, inconsistent data, missing data, incorrect data, invalid data, and spelling errors [20, 47, 53]. Methods such as N-gram-based duplicate record detection are employed to enhance preprocessing efficiency [47]. Data cleaning can be implemented manually, through specific application programs, or via generic application-agnostic methods [20].

Preprocessing: A broader term, preprocessing includes various operations performed on raw data before model training, such as data cleaning, transformation, dimensionality reduction, and the application of robust statistics [50]. It is crucial for handling the sheer Volume, high Velocity, and diverse Variety of big data [20, 38].

Data Augmentation: This technique involves creating new, synthetic data from existing datasets to increase their size and diversity [53, 47, 101, 106]. It addresses data scarcity and class imbalance by using methods like pseudo-labeling, co-training, expectation-maximization, and Generative Adversarial Networks (GANs) [53, 47, 101].

Resampling Techniques: To address uneven class distribution or class imbalance, which can significantly reduce model performance, resampling techniques are applied [53, 47, 96]. Examples include Synthetic Minority Over-sampling Technique (SMOTE) to oversample minority classes [53].

Transfer Learning: This method involves fine-tuning pre-trained models on a target dataset, leveraging knowledge transferred from similar, often larger, datasets to improve performance, especially when domain-specific data is limited or expensive to obtain [53, 106, 101].

Data Integration and Fusion: This process combines multiple datasets, often from diverse and heterogeneous sources, to enrich the information base and resolve complex data structures and semantic integration difficulties [20, 50, 38, 101, 106, 109].

Data Transformation: This involves converting data into usable forms that meet the requirements of specific ML models [50, 101].

Data Validation: A critical step that ensures data conforms to predefined requirements and identifies invalid data points [47, 91].

Continuous Monitoring and Feedback Mechanisms: Ensuring high data quality is not a one-time task but requires a continuous, complex, and context-dependent approach across the entire data lifecycle. Dynamic feedback mechanisms are essential for adapting to evolving data characteristics and quality issues in big data and ML environments [20, 91].

The significance of data curation in contemporary big data and artificial intelligence pipelines arises from multiple interdependent factors. Machine learning models exhibit substantial dependence on dataset quality during both training and evaluation phases, with substandard data leading to inaccurate, unreliable, or biased predictions, as evidenced by documented failures in IBM's cancer treatment AI and Google Flu Trends [47, 53]. The inherent characteristics of big data—encompassing volume, velocity, variety, and value—further compound these quality challenges [20, 38, 91]. Large-scale datasets render traditional cleaning methodologies computationally expensive, while high-velocity data generation demands rapid processing that conflicts with thorough quality assurance. Additionally, the heterogeneous nature of data types complicates integration processes and quality assessment procedures. The field has witnessed a paradigmatic shift toward data-centric AI approaches, which prioritize data maintenance, understanding,

evaluation, and improvement over exclusive focus on algorithmic advancement [53]. This transition acknowledges that enhancing data quality often yields greater efficiency gains than merely expanding dataset size [106]. In federated learning contexts, noise heterogeneity arising from varied local dataset quality presents persistent challenges that compromise model performance [96], necessitating systematic data curation strategies across distributed environments. Consequently, systematic data curation—incorporating evaluation frameworks, multidimensional metrics, and advanced improvement techniques—represents an essential component for extracting value from data in modern big data and machine learning environments, ultimately ensuring the performance, fairness, robustness, safety, and scalability of AI systems [106].

2.5 Considering Both Data Quality and Protection

Achieving a balance between data protection and data quality is a fundamental challenge in the design of modern big data pipelines and distributed service-based systems. Recent advancements have shifted the focus beyond traditional trade-offs, promoting frameworks and methodologies that actively seek to reconcile, rather than simply choose between, these often competing objectives. As organizations increasingly realize the practical benefits and significant value of big data, they also acknowledge the limitations of current big data ecosystems, especially in terms of data governance. In this context, the need for privacy-aware systems enforcing sensitive data protection without compromising data quality throughout the entire data lifecycle arises. As organizations increasingly protect sensitive data, data quality assessment must also evaluate how closely sanitized values approximate their original counterparts, positioning post-anonymization quality measurement as fundamental to achieving both privacy and utility objectives.

As previously established, privacy-enhancing techniques inherently involve a trade-off between the level of data protection and the preservation of data utility. For example, differential privacy provides strong confidentiality guarantees, but the necessary introduction of statistical noise can diminish the precision and analytical value of the data. In contrast, methods like k-anonymity and l-diversity generally maintain higher data utility but are more

vulnerable to sophisticated re-identification attacks. To quantify the impact of these privacy transformations, a range of data quality metrics has been developed, such as generalized information loss (GenILoss), the discernibility metric, minimal distortions, and average equivalence class size (C_AVG) [70, 45, 88]. Nevertheless, no single metric has gained universal acceptance. This lack of consensus underscores the fact that measuring data quality in privacy-aware environments is a highly context-dependent and often subjective endeavor.

Both industry and academia have begun to investigate the issue, recognizing the need of new security requirements [29] and the importance of addressing the conflict between the need to share and the need to protect information [46, 97, 3, 55, 31, 76, 74, 72, 61, 59, 41, 19, 2, 1], from a data governance perspective [17, 4], and, more in general, to ensure compliance of big data pipelines with generic non-functional requirements [9, 14]. Table 2.2 provides a comparative analysis with relevant existing approaches, highlighting how few industrial solutions compare to our framework according to the following critical features that are the foundation of this thesis:

F1 – Service-Based Pipeline Support in the Cloud-Edge Continuum: The ability to effectively operate within distributed environments spanning cloud and edge infrastructure.

F2 – Quality-Aware Service Selection Ensuring Data Protection: The capacity to optimize service selection processes, maintaining data quality across the pipeline and ensuring robust data protection measures.

F3 – Framework-Agnostic Data Protection: The degree to which each solution is bound to specific data protection techniques.

F4 – Policy Framework Effectiveness: The degree to which each solution supports systematic specification of policies or privacy measures.

Most evaluated industry tools (e.g., Microsoft Presidio, Apache Ranger, Google Cloud DLP, AWS Macie, IBM Guardium, Apache Sentry) provide strong support for at least one of these features but typically fall short of full integration. Specifically, while F3 (agnosticism to protection technique)

is well-supported, F2 (simultaneous optimization of quality and protection) is rarely implemented in full. The discussed framework uniquely excels, offering a policy-driven, quality-aware service selection feature across the entire data pipeline and not being tied to a single deployment paradigm (cloud-only vs. hybrid).

Table 2.2: Comparative analysis with relevant existing approaches. Feature support is classified according to ✓(fully supported), ~(partially supported or limited in scope), ✗(not supported)

Solution	F1	F2	F3	F4
Microsoft Presidio [71]	✓, can integrate within cloud-edge pipelines	~, focuses on data redaction	✓, compatible with diverse techniques	~, pre-built PII detectors with configurable policies
Apache Ranger [11]	~, mostly limited to cloud settings	✗, provides access control rather than service optimization	✓, integrates with various techniques	✓; effective framework with systematic policy control
Google Cloud DLP [48]	✓, primarily within Google Cloud	~, focuses on redaction and anonymization	✓, works across data types	~, flexible templates for data masking and redaction policies

Continued on next page

Table 2.2 – continued from previous page

Solution	F1	F2	F3	F4
AWS Macie [6]	~, suited for AWS cloud infrastructure	~, prioritizes data protection	✓, AWS-centric	~, supports predefined PII types but less customizable
IBM Guardium [57]	✓, supports hybrid cloud and on-prem setups	✗, focuses on monitoring and access control	✓, adaptable to multiple frameworks	✓, extensive policy-based access control and monitoring
Apache Sentry [12]	~, Hadoop ecosystems	✗, static access control	✗, closely tied to Hadoop	~, supports column and row-level access control
Apache Atlas [10]	~, Hadoop-origin with on-prem/cloud deployability (VMs/containers)	✗, centers on metadata cataloging, lineage, and classification rather than service optimization	~, strong integrations in data lake ecosystems (Hive, HBase, Kafka, Spark) via hooks/APIs	~, expressive type system and tag-based governance; enforcement typically via Ranger

Continued on next page

Table 2.2 – continued from previous page

Solution	F1	F2	F3	F4
OpenMetadata [77]	✓, supports on-prem and multi-cloud deployments with agent-based ingestion	∼, enables governance and data-quality workflows but not automated service selection/optimization	✓, source- and technique-agnostic with broad connectors (warehouses, DBs, BI, ML)	∼, effective policies and automation; systematic enforcement depends on downstream integrations
Our work	✓, suitable for cloud-edge environments	✓, selection of services that optimize quality while ensuring protection	✓, data-protection techniques agnostic	✓, effective framework with systematic policy control

Despite notable progress, several limitations persist. Many approaches remain tightly coupled to specific platforms or database systems, restrict themselves to single privacy criteria, or define security only in terms of access control frequently ignoring post-protection data quality altogether. When attribute-based access control (ABAC) models are utilized, they typically remain at a theoretical level and lack concrete implementation that meaningfully incorporates data quality considerations. Batch-oriented or static protection (through access control or data sanitization) typically precludes systematic, context-aware optimization of privacy and utility along the entire data lifecycle, particularly in distributed or collaborative (multi-provider) settings. Furthermore, while effective policy frameworks and hybrid pipeline support are recognized as desirable, practical implementations

are still emerging in academic and industrial contexts.

In summary, sophisticated and systematic approaches are emerging to bridge the gap between data protection and quality in modern data pipelines. The state of the art increasingly calls for: i) dynamic, context-aware frameworks that adaptively balance privacy and utility based on specific use cases; ii) multi-criteria optimization techniques that consider a range of quality and protection metrics simultaneously; iii) policy-driven architectures that allow systematic control over data handling practices; and iv) hybrid deployment models that leverage the strengths of both cloud and edge computing environments. These trends underscore the growing recognition that effective data governance in the era of big data requires holistic solutions integrating both protection and quality considerations throughout the data lifecycle.

3

Data Governance for ETL Data Processing Pipelines

This chapter presents the development and implementation of a framework that addresses data protection requirements and data utility preservation through access control mediated data processing. We first present the ACL-mediated data processing pipeline, covering the architectural foundation, transformation-based governance mechanisms, and policy framework (Section 3.1). We then discuss the practical implementation of the data engine and present experimental evaluation results that demonstrate the effectiveness of our approach (Sections 3.2 and 3.3).

3.1 ACL-Mediated Data Processing Pipeline

We propose a data governance framework for data processing pipelines that integrates access control mechanisms with data transformation capabilities. Our approach shifts from traditional access denial paradigms to adaptive data transformation, ensuring that authorized users receive appropriately processed data based on their privilege levels and contextual requirements while maintaining privacy and security protections.

We demonstrate this framework through a practical implementation targeting machine learning-based anomaly detection in traffic and pollution monitoring for the city of Oslo. The system automatically adapts the training and inference of machine learning models based on user privilege levels,

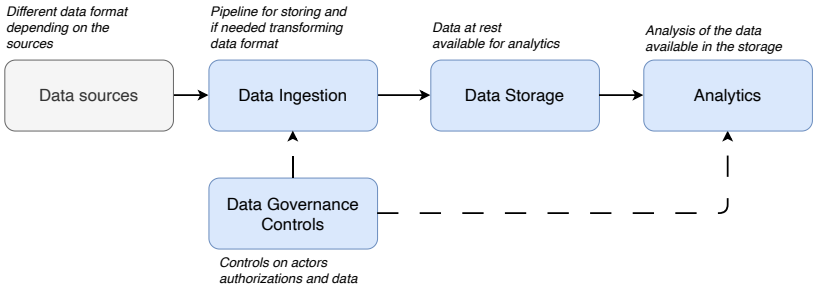


Figure 3.1: Architectural view of the ACL-mediated data processing engine

ensuring that both training data and inference results align with the user's permitted access level while maintaining privacy and security protections.

3.1.1 Architectural Foundation and Data Engine Design

Our approach centers on a data processing engine that enforces access control through adaptive data transformation, providing an ecosystem for managing the entire data lifecycle from collection to analytics. Figure 3.1 illustrates the architectural view of this integrated system, which serves as a middleware between diverse data sources and analytics procedures.

The data processing engine architecture is built upon four interconnected building blocks that work together to ensure secure and efficient data management through adaptive transformations:

Data Collection Pipeline: This component implements a multi-stage process that adapts to various data sources and formats while applying immediate policy-based transformations. The pipeline follows a structured approach where initial tasks collect data from heterogeneous sources, subsequent tasks perform both necessary format transformations and policy-driven transformations, and final tasks store appropriately transformed data. This staged approach ensures that access control policies are embedded throughout the collection process.

Policy-Aware Data Storage: The storage component maintains multi-

ple representations of data according to different access levels and transformation requirements. Rather than storing raw data with restricted access, the system maintains transformed versions that align with various user privilege levels, ensuring efficient access while preserving security properties.

Adaptive Data Governance: This layer implements dynamic access control policies that determine appropriate data transformations based on user identity, context, and operational requirements. The governance component continuously monitors access patterns and automatically adapts data representations to match changing user privileges and system conditions.

ML-Aware Analytics Integration: The analytics component provides intelligent interfaces between transformed data representations and machine learning procedures. This component ensures that ML models are trained and operate on data that matches the requesting user's access level, automatically selecting appropriate model variants based on the user's privileges.

The solid arrows in Figure 3.1 represent the standard data flow from sources through processing to analytics, while the dashed arrows highlight the governance controls that monitor and regulate all activities throughout the system. This architecture ensures that access control operates through data transformation rather than access denial, occurring at multiple critical points throughout the data processing lifecycle.

3.1.2 Transformation-Based Data Governance Through Enhanced ETL

Building upon this architectural foundation, we implement a workflow that extends traditional Extract, Transform, Load (ETL) approaches with dynamic, policy-driven data transformations. The solution [8] applies data transformations triggered by access control policies and applied to ensure that each user receives appropriately transformed data based on their identity, service characteristics, data annotation and context of execution.

Figure 3.2 demonstrates how traditional ETL processes are extended to address the security and privacy challenges of multi-tenant scenarios. The white boxes represent the traditional ETL components: connectors for establishing data source connections, connector handling tasks for managing different data formats and velocities, data gathering and transformation tasks,

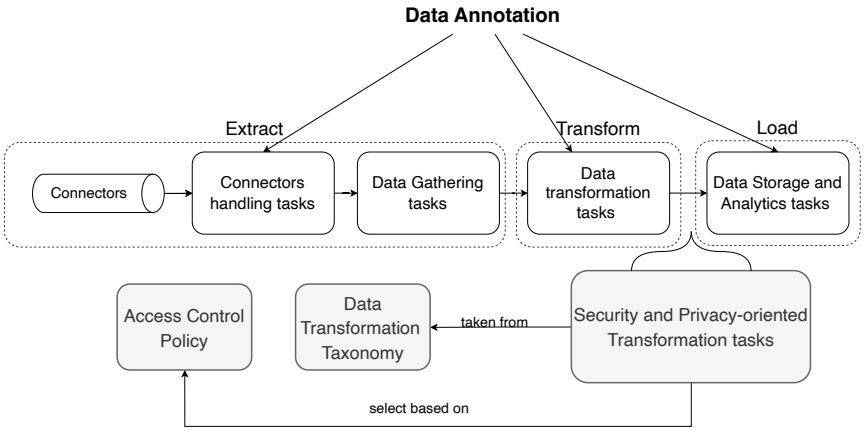


Figure 3.2: Policy-driven transformations and access control mechanisms applied throughout the ETL pipeline

and final storage or forwarding tasks.

We integrate data annotation and policy-driven transformations (shown in grey boxes) extending the traditional pipeline to create multiple data representations. Data annotations are applied at three critical stages: during connection handling to annotate raw data at gathering time, during data transformation to annotate specific fields before processing, and during storage tasks to provide annotations on transformed data results.

These annotations drive security and privacy-aware transformations that range from basic data reduction and restructuring operations to complex operations such as encryption, anonymization, and aggregation. The transformations are selected based on a taxonomy and triggered by access control policies, creating multiple transformed versions of the same data that match different privilege levels and operational contexts.

3.1.3 Access Control Policy Framework

The access control framework builds upon attribute-based access control (ABAC) principles [7], extending them with transformation obligations to

support adaptive data transformation.

Our policy model adapts traditional ABAC elements to address big data specific requirements. Each policy is defined as a 5-tuple <subject, object, action, environment, datatrans> where:

The **subject** component defines users or service providers issuing access requests, specified as <id, PC> where id represents a user class and PC contains policy conditions on subject attributes. This allows for flexible classification of requestors based on roles, departments, clearance levels, or other relevant characteristics.

The **object** component describes any data whose access is governed by the policy, structured as <type, PC> where type defines the data type (files, databases, tables, columns) and PC specifies conditions on object attributes such as creation date, sensitivity level, or data classification.

The **action** component encompasses operations from traditional database operations (CRUD) to complex big data procedures such as Apache Spark DAGs, Hadoop MapReduce operations, or complete analytics pipelines.

The **environment** component captures contextual conditions including temporal factors (time of day, emergency situations), spatial factors (location, weather conditions), and operational factors (network conditions, system load, threat levels).

The **data transformation** component specifies security and privacy-aware transformations aligned with the CIA triad principles. Confidentiality transformations include encryption, hashing, and anonymization techniques. Integrity transformations ensure data authenticity and protection from unauthorized modifications. Availability transformations focus on maintaining authorized access while implementing appropriate security controls.

Our framework includes specialized transformations for temporal and spatial data. Temporal transformations introduce controlled delays in data access, where data produced at time T becomes accessible at time T+δ, with δ determined by policy requirements and desired visibility levels. For example, as illustrated in Figure 3.3, police departments might access traffic data in real-time (green segments) while air quality officers receive the same data with a one-hour delay (red segments).

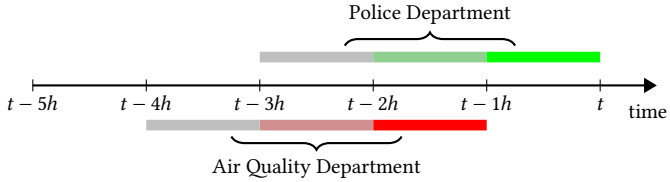


Figure 3.3: Example of Temporal Transformation

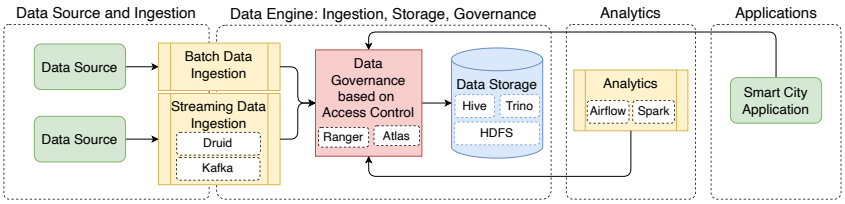


Figure 3.4: Apache-Based Big data engine.

Spatial transformations modify the granularity of geolocation information, providing different levels of spatial accuracy based on user roles and context. In emergency situations, law enforcement might receive precise location data while citizens receive information aggregated at the district or city level.

3.2 Data Engine Implementation

We implemented the data engine in Figure 3.1 using services and components of the Apache-based ecosystem, integrating the data governance approach in Section 3.1 within a complete Apache-based Big Data engine. Figure 3.4 highlights all the building blocks of the Big Data engine, their interactions and the technologies they are based on, as follows.

Data ingestion supports batch and streaming ingestion of the data produced by sources (e.g., sensors) or other providers (e.g., smarty city apps) and later deposited in the data storage via our data governance

approach based on access control. It supports batch and stream ingestion. For batch ingestion, the data collection process is executed through the source API, or via manual upload. Data may be collected either in an active (i.e., by directly picking up data – event-based) or passive (i.e., by a regular data receiving process – time-based) way. Batch ingestion builds on Apache *NiFi*, *Hive*, *Trino*, and *HDFS*. The automatic batch ingestion is implemented via ad hoc scripts or via specific data-flow collectors (i.e., Nifi). In a batch ingestion, data schema are defined. Hive uses HDFS to hold the data and a MySQL database (Metastore) to hold the schema and other information such as types, number of records and other metadata. Metastore can also be used by other tools (e.g., Trino), facilitating data centralization and favouring the diversification of access methods. Once the data and the schema are associated with each other, it is possible to query data through queries written in SQL/SQL-like languages. For stream ingestion, the data collection process relies on Apache *Kafka* and *Druid*. The latter offers functionalities to efficiently apply real-time transformations on a stream of data. It is used in combination with Kafka to manage the ingestion-based access control enforcement on stream data. Apache Kafka is used, with each data source sending data towards a specific queue, identified by a unique label called *topic*. Stream ingestion is implemented through a streaming endpoint in Kafka capable of accepting connections and receiving stream of data. Producers submit their data to a Kafka topic, which flows into the data engine. Kafka acts as an intercommunication channel between the components and as a buffer. Data can in fact be deposited inside Kafka to be consumed (even multiple times) later. We complemented Kafka with Apache Druid to apply real-time transformations on a stream of data.

Data storage provides functionalities to ensure data availability, fault tolerance, and reliable distribution of data on the nodes of the cluster. Data storage is indirectly involved in the processing procedure since nodes typically take part in the processing of local data. Any operations (read/write) in the data storage is mediated by the our data governance approach. It is implemented using components *Hive*, *Trino*,

and *HDFS*.

- *HDFS* is a distributed, scalable, and portable file system designed to deal with big data issues, based on blocks to be distributed across nodes.
- *Hive* is a structured data warehouse based on Hadoop, capable to manage large datasets that reside in a distributed storage.
- *Trino* is a big data distributed query engine that uses the SQL query language.

HDFS, Hive, Trino are responsible for storage and data access.

The storage of structured data is built on relational DBMS in Apache Hive. Data storing tasks handle procedures to connect to Hive and do the final checks on the data format to guarantee absence of incompatibilities (e.g., data types) between the storing technologies (i.e., Hive), and the data after transformation. As far as unstructured and semi-structured data is concerned (e.g., documents, folders, images), our engine adopts HDFS.

Data Governance implements the advanced ETL in Section 3.1 that enforces access to data at ingestion-time and when data is fed into the analytics pipeline for analysis and processing. It relies on components *Ranger* and *Atlas* as follows.

- *Ranger* provides a centralized platform to define, administer and manage security policies consistently across Hadoop components via specific plugins. It supports the specification, assignment, and enforcement of attribute-based access policies on resources (e.g., files, folders, databases, tables, or columns) or tags (annotations). These policies can be set for individual users or groups, and enforce security and privacy-oriented transformations.
- *Atlas* is used to implement the annotation process by means of tags and provides data lineage to improve the auditability of data flows. Data annotations (tags) are then used to define and enforce tag-based policies in Apache Ranger.

Policies defined in Ranger mediate access to resources or part of them (i.e., databases, tables, columns and cells), increasing data governance flexibility. The result of a policy execution is a data transformation that addresses specific needs emerging in the considered scenario. We note that data transformations refer to Hive transformations. Hive in fact provides several manipulation functions that the user can extend to define her own User Defined Functions (UDFs). A classic UDF takes a single value as input and provides a single value as output, that is, the UDF is called for each row of the query result. All functions, both those provided by the system and UDFs, can be used as custom masking option in Ranger.

Data analytics implements the big data pipelines analyzing data in the data storage. It relies on components *Spark* and *Airflow* as follows.

- *Spark* is an advanced, unified analytics engine for large-scale data processing. The core idea of Spark is to provide an expressive computing system (not limited to Hadoop MapReduce model) by exploiting in-memory processing of cached data to avoid saving intermediate results to disk and caching data for repetitive queries. Apache Spark framework enables four different programming languages for map-reduce programming (i.e., Java, Scala, Python, R). We considered Scala mainly thanks to the greater documentation support and community.

- *Airflow* is an orchestrator of processing pipelines aimed to provide scheduling and monitoring capabilities. Airflow pipelines are implemented in Python for better dynamic pipeline generation. The pipeline is modelled as a Direct Acyclic Graph (DAG) of tasks that can be Spark jobs.

We recall that each access to data by big data pipelines is mediated by our data governance approach.

3.3 Experimental Evaluation

We experimentally evaluated our approach in the smart city reference scenario presented in Section 3.1. In the following, we first present the considered dataset and our experimental settings (Section 3.3.2); we then present a concrete execution of our data governance approach in terms of access control policy enforcement based on spatial transformations (Section 3.3.3); we further explain the training of 5 anomaly detection models on 5 different spatial views of the considered dataset and the results of their execution at inference time (Section 3.3.4); we finally discuss how policy enforcement affects the privacy and quality of the overall analytics (Section 3.3.5).

3.3.1 ML-Adaptive Policy Enforcement: Oslo Traffic and Pollution Case Study

Our framework has been applied to ML-based anomaly detection in Oslo's traffic and pollution monitoring system. The implementation demonstrates how user privilege changes trigger automatic adaptation of both training data and inference models to maintain consistent access control throughout the ML pipeline.

Policy enforcement operates through a systematic evaluation and adaptation process that dynamically matches machine learning models to user privilege levels. When an access request is submitted, the system evaluates the request against subject, object, action, and environment conditions, then automatically selects or adapts machine learning models trained on appropriately transformed data that matches the user's access level.

The system operates with multiple machine learning model variants trained on different levels of data granularity and detail. When users change roles or permissions, the system automatically transitions to using models trained on data that matches their new privilege level. To show the framework's capabilities in the context of machine learning-based anomaly detection, consider three representative policies from the Oslo traffic and pollution monitoring system:

Policy 1 - Emergency Response Access: When emergency response coordinators request anomaly detection results during crisis situations, the

system provides access to models trained on complete, real-time data including precise location coordinates and individual sensor readings. This ensures rapid identification and response to traffic incidents or pollution events.

Policy 2 - Municipal Department Access: When municipal planning departments request the same anomaly detection capabilities under normal operating conditions, the system automatically switches to models trained on temporally delayed and spatially aggregated data. This provides useful planning insights while protecting operational security and individual privacy.

Policy 3 - Public Access: When citizens or researchers request access to anomaly detection results, the system uses models trained on anonymized, district-level aggregated data that can identify general trends and patterns while protecting sensitive operational details and individual activities.

These examples demonstrate how the same machine learning-based anomaly detection system can provide different levels of insight based on the user's role and context. Each user receives meaningful results appropriate to their legitimate needs while the system maintains consistent privacy and security protections through automatic model adaptation.

Scalability and ML Integration: The framework's design ensures that policy-driven transformations and model selection can be implemented as scalable ML pipelines, allowing for efficient processing of large-scale urban sensor data while maintaining security properties. The separation of concerns between data transformations and ML model variants enables independent optimization of both accuracy and security requirements.

Such an integrated approach provides a foundation for extending ACL-mediated access control beyond single services to complex multi-service pipelines, where the principles established in single-service scenarios can be propagated and adapted throughout distributed machine learning processing ecosystems.

3.3.2 Experimental Settings

We tested our methodology in a smart city scenario using a dataset taken from the open data of the local public transportation in Oslo. The dataset counts 9.88M rows and 25 columns (attributes). Each row contains informa-

Table 3.1: Attributes and related descriptions of the dataset used in the experiments.

Attribute	**Description**
DateTime	Timestamp specifying when the position/status was recorded/updated
LinkDistance	Distance in meters between the previous stop (or current, if located at stop) and the next stop
Percentage	How much of the total distance (percentage) that has been traversed at the time of the message
LineRef	Reference to the line in question
DirectionRef	Reference to the direction in question
PublishedLineName	Name describing the line in question
OriginRef	Reference to the Origin in question
OriginName	Name describing the origin of the departure
DestinationRef	Reference to the destination in question
DestinationName	Name describing the destination of the departure
OriginAimedDepartureTime	Origin aimed departure time
DestinationAimedArrivalTime	Destination aimed arrival time
VehicleRef	Reference to the vehicle in question
Delay	Delay-time, defined as "PT0S" (0 seconds) when there are no delays
HeadwayService	Field defining whether the service is a headway service
InCongestion	Field defining whether the traffic is in congestion or other circumstances which may lead to further delays
InPanic	Boolean field, indicating that the driver reported an "out of order" status
GPS (Latitude and Longitude)	Position of the public vehicle, according to the timestamp recorded
StopPointRef	Reference to the stop point in question
VisitNumber	Number of the stop point in question
StopPointName	Name of the stop point in question
VehicleAtStop	Field defining whether the vehicle is at the stop
DestinationDisplay	Name of the next destination

tion about the location of a single bus/tram and other information such as origin, destination, delay, malfunctioning. All attributes in the dataset are detailed in Table 3.1.

An anomaly detector was implemented to spot abnormal situations, that is, any situation diverging from a *normal* one. The goal of the detector is to retrieve early indicators of critical events such as accidents, unauthorized protests, or terrorist attacks.

In the above settings, we considered that the traffic anomaly detector could be used by three categories of users (i.e., policeman, air quality control officer, citizen) under two different environment conditions (i.e., *normal*, *emergency*), leading to five different access control policies mediating access to data by means of spatial transformations that limits the view of the detector on the data.

Our experiments were run on a single node virtual machine running Ubuntu 20.04.2 LTS, installing the big data engine in Section 3.2. The virtual machine was equipped with Intel Xeon E5 2620 - 2.095GHZ CPU and 64 GB of RAM.

3.3.3 Data Transformation: Aggregation and Filtering

The data transformation process consists of a preparation and a filtering phase, working as follows.

Aggregation. Data aggregation is a fundamental step for anomaly detection on time series and properly defines the concept of *time-unit-anomaly*. Our experiments represented and collected data on movable points (buses and trams) from a *"fixed point of view"* (i.e., a specific area of the city), and aggregated them in a 5-minute time window representing our time-unit-anomaly.

Starting from the vehicles traffic dataset in Table 3.1, data was prepared for analytics according to a time aggregation executed at ingestion time and producing the dataset in Table 3.2. Collected data has been aggregated in 5-minute time windows considering data coming from n distinct locations. $k < n$ was then selected, where k represents the number of spatial areas (spatial clusters) for spatial aggregation. Each spatial area is represented in terms of its cluster centroid, that is, a set of GPS (latitude and longitude) coordinates. Obviously, changing the value of k also changes the number

Table 3.2: Aggregated dataset.

Attribute	Description
Date	The timestamp of the current measurement/instance
ClusterLatitude	The latitude of the cluster centroid from where aggregate
ClusterLongitude	The longitude of the cluster centroid from where aggregate
Delay	Average of the busses
Percentage	Average total distance that has been traversed
InPanic	Average busses in panic mode
InCongestion	Average busses in congestion
DestinationAimedArrivalTime	Average Destination aimed departure time
OriginAimedDepartureTime	Average Origin aimed departure time
HeadwayService_False	Count of instances for non-headway services
Anomaly	0,1 indicating if the instance represents a normal case 0 or an anomaly 1 (for quantitative evaluation purpose only)

of zones and centroids considered, and of the level of spacial precision: a larger k gives larger granularity and thus more precise information, whereas a smaller k leads to smaller granularity and thus less precise information. Our experiments used five different predictive models based on *k-means* trained according to five different aggregated datasets with $k \in \{5, 10, 25, 50, 100\}$.

Properly modeling the spatial dimensions of the data considering spatial autocorrelation phenomena [30, 25] emphasizes possible agreements or discrepancies related to the different sensors measurements located in different positions. Therefore, according to the considered clustering model, we collected different aggregated data for training the final predictive models.

Filtering. We exploited the values of k to define the access control policies performing spatial filtering on data according to the role (i.e., policeman, air quality control officer, citizen) and the conditions of the environment (*normal*, *emergency*). Intuitively, a user with more permissions can use a larger k, whether a user with fewer privileges should use a smaller k. Filtering on the aggregated and tagged data was performed at ingestion time by changing k

as specified in the following policies.

Policy 1: ⟨Citizen, TrafficData, Read, Normal, AggregationClusters=5⟩, with *Citizen* ≡ ⟨user, {(department="Citizen")}⟩ and *TrafficData* ≡ ⟨file, {(tag="traffic")}⟩
This policy specifies that *citizens* can access low-accurate data in a *normal* situation. Data quantization is very low (5 centroids), thus the retrieved data serve only to indicatively show the anomalous area.

Policy 2: ⟨Citizen, TrafficData, Read, Emergency, AggregationClusters=10⟩, where *CitizenUser* ≡ ⟨user, {(department="Citizen")}⟩ and *TrafficData* ≡ ⟨file, {(tag="traffic")}⟩
With this policy, during an *emergency*, *citizens* can access medium-low-accurate data. Data quantization is low (10 centroids). Retrieved data allows to see the anomalous area.

Policy 3: ⟨AirQualityUser, TrafficData, Read, Normal, AggregationClusters=25⟩, where *AirQualityUser* ≡ ⟨user, {(department="AirQuality")}⟩ and *TrafficData* ≡ ⟨file, {(tag="traffic")}⟩
In this case, the policy states that *air quality control officers* can access medium-accurate data both in *normal* and *emergency* situations. Data quantization is medium (25 centroids). Retrieved data is still useful to outline the zone where an anomaly occurred.

Policy 4: ⟨Policeman, TrafficData, Read, Normal, AggregationClusters=50⟩, with *Policeman* ≡ ⟨user, {(department="Police")}⟩ and *TrafficData* ≡ ⟨file, {(tag="traffic")}⟩
The policy states that *policemen* can access accurate data in a *normal* situation. Data quantization is high (50 centroids).

Policy 5: ⟨Policeman, TrafficData, Read, Emergency, AggregationClusters=100⟩, where *Policeman* ≡ ⟨user, {(department="Police")}⟩ and *TrafficData* ≡ ⟨file, {(tag="traffic")}⟩
During an *emergency*, *policemen* can access the most accurate data. Data quantization is very high (100 centroids) resulting in a more accurate model guaranteeing a more reliable identification of the

anomaly location.

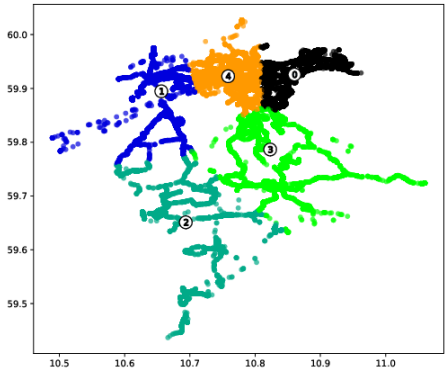
Policy enforcement led to the generation of the 5 different datasets in Figure 3.5.

3.3.4 Data Analytics: Model Training and Inference

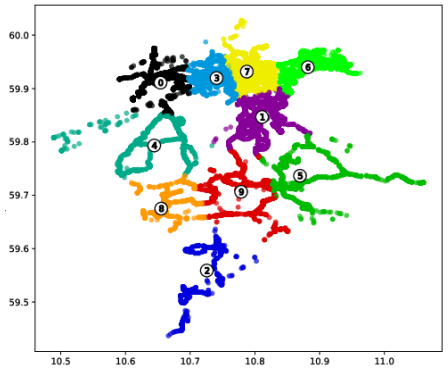
Our experiments used 5 time series data collected from March, 1st, 2022 at 17:40 to October, 18th, 2022 at 14:45. The training set included time series from March, 1st, 2022 at 17:40 to the end of September 2022; the test set included the remaining 18 days of October. The testing time series were randomly perturbed by considering for each test instance 100 times the real value of average delay, average in panic, and the average in congestion buses. 1% of the testing instances were randomly selected and perturbed. The output files of the experiments are available online.[1]

Our training process differs from traditional processes where a single model is trained on the original dataset and the inference results are aggregated and filtered according to users' privileges. It rather builds on 5 different models trained on the 5 different datasets generated at ingestion time, which natively address the access control requirements. Each model (M1–M5 in Table 3.3) corresponds to a policy (Policy 1–Policy 5 in Section 3.3.3). In particular, our analytics pipeline performs a training phase on the 5 subset of data obtained after spatial filtering (representing normal situations) and generates 5 predictive models. At this point, each time a specific user decides to perform monitoring through the anomaly detector, it uses the model corresponding to his/her role. In this way, the model natively addresses data protection because it is built with data aggregated according to the access control policies based on the user's privileges and only responses that can be accessed by the user requesting the analytics are used. In other words, the system presents as much sensitive data as possible and the detected anomalies at the level of detail granted by the user's privileges, thus balancing operativity and GDPR compliance.

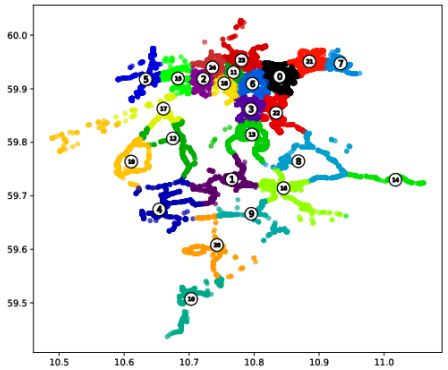
[1]Output files are available at: `http://www.di.uniba.it/~mignone/materials/MBDAaaS/anomdet.7z`



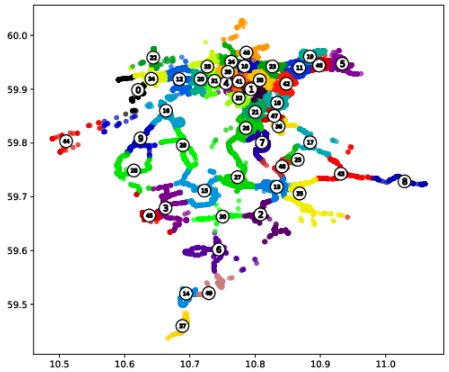
(a) Policy 1 (user: Citizen, env: Normal, k: 5)



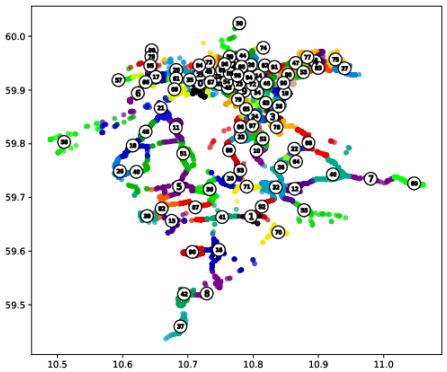
(b) Policy 2 (user: Citizen, env: Emergency, k: 10)



(c) Policy 3 (user: Air Quality, env: Any, k: 25)



(d) Policy 4 (user: Policeman, env: Normal, k: 50)



(e) Policy 5 (user:Policeman, env:Emergency, k:100)

Figure 3.5: Results of the enforcement of policies in Section 3.3.3. Spatial transformation (number k of clusters) depends on the user role (*user*) and the status of the environment (*env*).

Table 3.3: Results in terms of precision, recall, and f1-score. We also report the training time (in seconds), the training set cardinality, as well as the value for $fp\%$ that maximizes the f1-score.

model	prec	rec	f1-score	fp%	train time	data card
M_1 (5_clus)	0.995	0.993	0.994	0	140	198312
M_2 (10_clus)	0.998	0.997	0.997	0	362	399677
M_3 (25_clus)	0.990	0.990	0.986	0	1997	910724
M_4 (50_clus)	0.981	0.983	0.982	0.1	1488	1577192
M_5 (100_clus)	0.982	0.990	0.985	0.01	7888	2546608

Table 3.3 shows the results in terms of weighted precision, recall, and f1-score w.r.t. the support of the classes normal and anomaly. f1-score is particularly suitable for the imbalanced task at hand due to the rare anomalous items and abundant normal cases. We considered the different predictive models M_1–M_5 generated according to the number of data clusters (5, 10, 25, 50, 100, resp.) imposed by Policy 1–Policy 5 transformations in Section 3.3.3. The results in Table 3.3 show that the number of clusters considered has a very low impact on the predictive performance (precision between 0.981 and 0.998, recall between 0.983 and 0.997), while it has a high impact on the amount of retrieved information. M_1 and M_2 being built on 5 and 10 clusters, resp., provide a rough idea of the area affected by an anomaly (e.g., north-south-east-west-downtown); M_3 being built on 25 clusters provides a more detailed idea of the area affected by an anomaly (e.g., district); M_4 and M_5 being built on 50 and 100 clusters, resp., provide a very detailed idea of the area affected by an anomaly (e.g., street, precise location).

Finally, we observed the results of the ablation analysis aimed to identify the best value for $fp\%$ and, therefore, of t. The experiments showed that, the more the clusters, the data sensitivity, and the data volume, the higher the value of $fp\%$ that maximizes the f1-score. Indeed, with $k = 5$, $k = 10$ and $k = 25$, the best results are obtained with a threshold t that is identified by assuming no false positives on the training set. On the contrary, with $k = 50$ and $k = 100$, it is necessary to admit a small percentage of false positives on the training set to avoid too conservative models that do not detect abnormal cases properly. This behavior was somehow expected since a larger number

of clusters requires the algorithm to consider cluster boundaries that are not clearly identifiable.

3.3.5 Discussion

Our main research objective focused on how to achieve better governance for big data processing pipelines, validating it in the domain of anomaly detection. That is, we aimed to ensure that the sensitive data used in the pipeline was protected, still producing high quality results. Our key findings are as follows.

Full Privacy Scenario. It considers models M1 and M2 to achieve a high degree of anonymisation at the expense of a substantial loss of quality in the data. What is important to note is that aggregation, performed before the training phase, actually changes the work and results of anomaly detection. In the case study, the system monitors areas of different sizes for each level of aggregation (Figures 3.5(a) and 3.5(b)). The citizen's view, for example, reduces the sensitivity of the model as small anomalies have less impact in a larger area. This behavior prevents the citizens from detecting anomalies he/she should not have access to and which are not of interest to him/her anyway.

Full Quality Scenario. It considers models M5 and M6 to greatly reduce anonymisation, while supporting a higher quality of data. As already discussed, aggregation carried out upstream changes the behavior of anomaly detection. In this case, the model is much more sensitive, making it possible to detect even point anomalies of greater interest to a security officer. We note that it is possible to take into account other signals (e.g., an emergency situation) to alter the granularity of the system, making it more flexible and adaptive.

Full Post-Aggregation Scenario. It considers the common approach with a single prediction model and a posteriori aggregation to anonymize the results. The model underlying such a system must be as granular as possible, avoiding the risk of removing information that is fundamental for the most privileged users. On the other hand, it is necessary to

find an aggregation heuristic that is not a simple average over more or less large aggregation windows, so as to avoid an anomaly detected at higher granularities being visible also in those at lower granularities. This approach could also breach the rules imposed by the GDPR: although not all data is visible to the final end users, during the analytics they are managed by possibly unauthorized services or users.

3.4 Chapter Summary

This chapter demonstrated how access control policies can be integrated into data processing workflows through transformation-based mechanisms at ingestion time. The Oslo traffic monitoring case study showed that policy-driven spatial aggregation maintains analytical performance while providing role-appropriate data granularity. This establishes the foundation for extending protection mechanisms to multi-service pipeline environments.

4

Data Governance for Service-Based Data Processing Pipelines

Building upon the foundational work in Chapter 3, this chapter addresses the challenge of maximizing data quality across entire service pipelines while maintaining compliance with data protection requirements. The work extends single-service transformation mechanisms to tackle orchestration and optimization challenges in multi-service data processing environments.

This chapter presents a framework that enables quality-aware service composition through policy-driven pipeline design and systematic service selection algorithms. The framework ensures that adaptive data transformation principles can be effectively applied at scale in distributed service ecosystems through three key abstractions: pipeline templates, pipeline instances, and optimization algorithms.

We first discuss the system model and present a reference scenario illustrating core challenges in service-oriented data processing (Sections 4.1 and 4.2). We then introduce pipeline templates as annotated specifications capturing functional requirements and data protection policies (Section 4.3). Finally, we present the pipeline instantiation process that transforms templates into executable instances through systematic service selection (Section 4.4).

4.1 System Model

We consider a service-based environment where a service-based data pipeline (service pipeline in the following) is designed to analyze data. Our service pipeline is enriched with metadata specifying data protection requirements and functional specifications, and models the data flow among component services, without posing any restrictions on the control flow. It is composed of the following parties:

- *Service*, software distributed by a service provider that performs a specific task;
- *Service Pipeline*, a sequence of connected services that collect, prepare, process, and analyze data in a structured and automated manner;
- *Data Governance Policy*, a structured set of privacy guidelines, rules, and procedures regulating data access, sharing, and protection;
- *User*, executing a service pipeline on the data. We assume the user is authorized to perform this operation, either as the data owner or as a data processor with the owner's consent.
- *Dataset*, the data target of the service pipeline. We assume all data is ready for analysis, that is, they underwent a preparatory phase addressing issues such as missing values, outliers, and formatting discrepancies.

A service pipeline is a graph formally defined as follows.

Definition 1 (Service Pipeline)**.** A Service Pipeline is as a direct acyclic graph G(V,E), where V is a set of vertices and E is a set of edges connecting two vertices v_i,$v_k \in V$. The graph has a root ($\bullet$) vertex $v_r \in V$, a vertex $v_i \in V_S$ for each service s_i, an additional vertex $v_f \in V$ for each parallel ($\oplus$) structure modeling the contemporary execution (*fork*) of services.

Modeling the pipeline as a directed acyclic graph ensures a sound representation of its data flow. This standard approach to representing workflows, including service pipelines, closely mirrors real-world systems.

We note that $V=\{v_r,v_f\}\cup V_S$, with vertices v_f modeling branching for parallel structures, and root v_r possibly representing the orchestrator. To simplify the explanation and maintain clarity, we model alternative execution paths as distinct service pipelines, rather than embedding alternative structures within a single pipeline. This representation is equivalent to having alternative structures within a single pipeline, as each distinct pipeline corresponds to one possible execution path. By separating these paths into individual pipelines, we avoid the additional complexity of modeling alternatives within the same structure, while fully capturing all execution possibilities.

We refer to the service pipeline annotated with both functional and non-functional requirements, as the **pipeline template**. It acts as a skeleton, specifying both the structure of the pipeline, that is, the chosen sequence of desired services, and the functional and non-functional requirements for each component service. We note that, in our multi-tenant cloud-based ecosystem, each element within the pipeline may have a catalog of candidate services. A pipeline template is then instantiated in a **pipeline instance** by selecting the most suitable candidates from the pipeline template.

This process involves retrieving a set of compatible services for each vertex in the template, ensuring that each service meets the functional requirements and aligns with the policies specified in the template. Since we also consider security policies that may necessitate security and privacy-aware data transformations, compatible services are ranked based on their capacity to fulfill the policy while preserving the maximum amount of information (*data quality* in this work). Indeed, our data governance approach, while applicable to a generic scenario, operates under the assumption that *preserving a greater quantity of data is correlated with enhanced data quality*, a principle that reflects many real-world scenarios [44, 15]. However, we acknowledge that this assumption may not universally apply and remain open to exploring alternative solutions in future works. The best service is then selected to instantiate the corresponding component service in the template. Upon selecting the most suitable service for each component service in the pipeline template, the pipeline instance is completed and ready to be compiled in an executable pipeline.

4.2 Reference Scenario

Our approach targets application domains involving sensitive data, such as Personally Identifiable Information (PII), that must be securely shared and protected across diverse and complex analytical processes involving multiple stakeholders. It is applicable across industrial use cases based on cloud-edge infrastructures, where data from third-party (IoT) devices are injected and shared via the cloud, as well as in data ecosystems across sectors such as healthcare, finance, law enforcement, and justice.

Our reference scenario draws on commonly used dataspaces, such as dataspace on public administration, focusing specifically on the law enforcement domain. Using open data, we selected a scenario that includes real sensitive records of individuals detained in Connecticut Department of Correction facilities while awaiting trial. [1] Various stakeholders may use this data for different objectives: public health agencies to monitor inmate health trends, judicial bodies to track case processing efficiency, advocacy groups to identify disparities in detention, policymakers to analyze the impacts on the criminal justice system, social services to prepare post-release support, researchers to study the broader social effects of pre-trial detention, and correctional departments to compare admission trends across facilities.

To streamline the use case, we focused on a subset of this real-world scenario, envisioning three Department of Correction (DOC) partners - Connecticut, New York, and New Hampshire - sharing data according to their privacy policies. In this scenario, a user from the Connecticut DOC seeks to compare admission trends in Connecticut's facilities with those in New York and New Hampshire to evaluate, for instance, possible discrimination and unfair treatment of individuals awaiting trial. Additionally, the policy requires that all service execution remains within the Connecticut DOC environment, mandating data protection measures if data transmission extends beyond Connecticut's borders.

Our reference scenario aligns with the latest regulations on data governance (e.g., the European AI Data Governance Act[2]) and artificial intelli-

[1] https://data.ct.gov/Public-Safety/Accused-Pre-Trial-Inmates-in-Correctional-Faciliti/b674-jy6w

[2] https://digital-strategy.ec.europa.eu/en/policies/data-governance-act

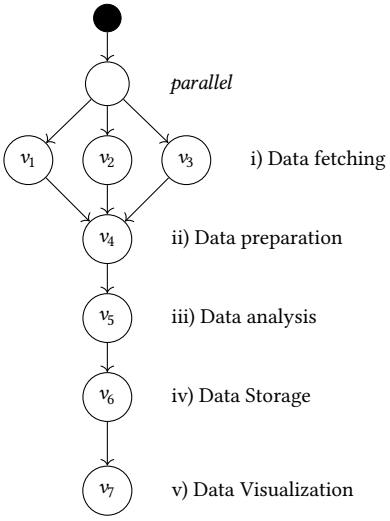


Figure 4.1: Service pipeline in the reference scenario

gence (e.g., the EU AI Act[3]). In particular, the EU AI Act identifies law enforcement and administration of justice as high-risk domains where proper data governance, risk management, and quality management systems must be employed in AI training and operation following the requirements on data quality and protection set in this work.

The user's objective aligns with the predefined service pipeline in Figure 4.1 that orchestrates the following sequence of operations: (i) *Data fetching*, including the download of the dataset from other states; (ii) *Data preparation*, including data merging, cleaning, and anonymization; (iii) *Data analysis*, including statistical measures like average, median, and clustering-based statistics; (iv) *Data storage*, including the storage of the results; (v) *Data visualization*, including the visualization of the results.

[3] `https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai`

4.3 Pipeline Template

Our approach integrates data protection and data management into the service pipeline using annotations. To this aim, we extend the service pipeline in Definition 1 with: *i)* data protection annotations that express transformations on data, ensuring compliance with data protection requirements, *ii)* functional annotations that express data manipulations carried out during service execution. These annotations enable the implementation of an advanced data lineage, tracking the entire data lifecycle by monitoring changes that result from functional service execution and data protection requirements.

In the following, we first introduce the annotated service pipeline, called pipeline template (Section 4.3.1). We then present both functional annotations (Section 4.3.3) and data protection annotations (Section 4.3.2), providing an example of a pipeline template in the context of the reference scenario in Section 4.2.

4.3.1 Pipeline Template Definition

Given the service pipeline in Definition 1, we use annotations to express data protection requirements and functional requirements on the services to be integrated into the pipeline. Each service vertex in the service pipeline is labeled with two mapping functions forming a pipeline template: *i*) an annotation function $\lambda : V_S \rightarrow P$ that associates a set of data protection requirements to be enforced on data, in the form of policies $p \in P$, with each vertex $v_i \in V_S$; *ii*) an annotation function $\gamma : V_S \rightarrow F$ that associates a functional service description $F_i \in F$ with each vertex $v_i \in V_S$.

The template is formally defined as follows.

Definition 2 (Pipeline Template)**.** Given a service pipeline G(*V*,*E*), a Pipeline Template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ is a direct acyclic graph extended with two annotation functions:

i) Data Protection Annotation λ that assigns a label $\lambda(v_i)$ to each vertex $v_i \in V_S$. Label $\lambda(v_i)$ corresponds to a set P_i of policies p_j to be satisfied by service s_i represented by v_i;

ii) Functional Annotation γ that assigns a label $\gamma(v_i)$ to each vertex $v_i \in V_S$. Label $\gamma(v_i)$ corresponds to the functional description F_i of service s_i represented by v_i.

We note that, at this stage, the template is not yet linked to any service. We also note that policies $p_j \in P_i$ in $\lambda(v_i)$ are combined using logical OR, meaning that the access decision is positive if at least one policy p_j evaluates to *true*.

4.3.2 Data Protection Annotation

Data Protection Annotation λ expresses data protection requirements in the form of access control policies. We consider an attribute-based access control model that offers flexible systematic authorization and adapts its standard key components to address the unique characteristics of a big data environment. Access requirements are expressed in the form of policy conditions that are defined as follows.

Definition 3 (Policy Condition)**.** A *Policy Condition pc* is a Boolean expression of the form (*attr_name* op *attr_value*), with op$\in\{<,>,=,\neq,\leq,\geq\}$, *attr_name* an attribute label, and *attr_value* the corresponding attribute value.

Built on policy conditions, an access control policy is then defined as follows.

Definition 4 (Policy)**.** A *policy* $p \in P$ is 5-uple $<subj, obj, act, env, T^P>$ that specifies who (*subject*) can access what (*object*) with action (*action*), in a specific context (*environment*) and under specific obligations (*data transformation*).

More in detail, *subject subj* specifies a service s_i issuing an access request to perform an action on an object. It is a set $\{pc_i\}$ of *Policy Conditions* as defined in Definition 3. For instance, (classifier="SVM") specifies a service providing an SVM classifier. We note that *subj* can also specify conditions on the service owner (*e.g.*, owner_location="EU") and the service user (*e.g.*, service_user_role="DOC Director").

Object obj defines the data governed by the access policy. It is a set $\{pc_i\}$ of *Policy Conditions* on the object's attributes. For instance, {(type="dataset"), (region="CT")} refers to an object of type dataset whose region is Connecticut.

Action act specifies the operations that can be performed within a big data environment, from traditional atomic operations on databases (e.g., CRUD operations) to coarser operations, such as an Apache Spark Direct Acyclic Graph (DAG), Hadoop MapReduce, an analytics function call, and an analytics pipeline.

Environment env defines a set of conditions on contextual attributes, such as time of the day, location, IP address, risk level, weather condition, holiday/workday, and emergency. It is a set $\{pc_i\}$ of *Policy Conditions* as defined in Definition 3. For instance, (time="night") refers to a policy that is applicable only at night.

Data Transformation T^P defines a set of security and privacy-aware transformations on *obj* that must be enforced before any access to data is given. Transformations focus on data protection, as well as on compliance with regulations and standards, in addition to simple format conversions. For instance, let us define three transformations that can be applied to the dataset, each performing different levels of anonymization: i) level *l0* (t_0^p): no anonymization; ii) level *l1* (t_1^p): partial anonymization with only first and last name being anonymized; iii) level *l2* (t_2^p): full anonymization with first name, last name, identifier, and age being anonymized.

Access control policies $p_j \in P_i$ annotating a vertex v_i in a pipeline template $G^{\lambda,\gamma}$ specify the data protection requirements that a candidate service must fulfill to be selected in the pipeline instance. Section 4.4 describes the selection process and the pipeline instance generation.

4.3.3 Functional Annotations

A proper data management approach must track functional data manipulations across the entire pipeline execution, defining the functional requirements of each service operating on data. To this aim, each vertex $v_i \in V_S$ is annotated with a label $\gamma(v_i)$, corresponding to the functional description F_i of the service s_i represented by v_i. F_i describes the functional requirements,

such as API, inputs, and expected outputs. It also specifies a set T^F of data transformation functions t_i^f, which can be triggered during the execution of the corresponding service s_i.

Function $t_i^f \in T^F$ can be: *i*) an empty function t_ϵ^f that applies no transformation or processing on the data; *ii*) an additive function t_a^f that expands the amount of data received, for example, by integrating data from other sources; *iii*) a transformation function t_t^f that transforms some records in the dataset without altering the domain; *iv*) a transformation function t_d^f (out of the scope of this work) that changes the domain of the data.

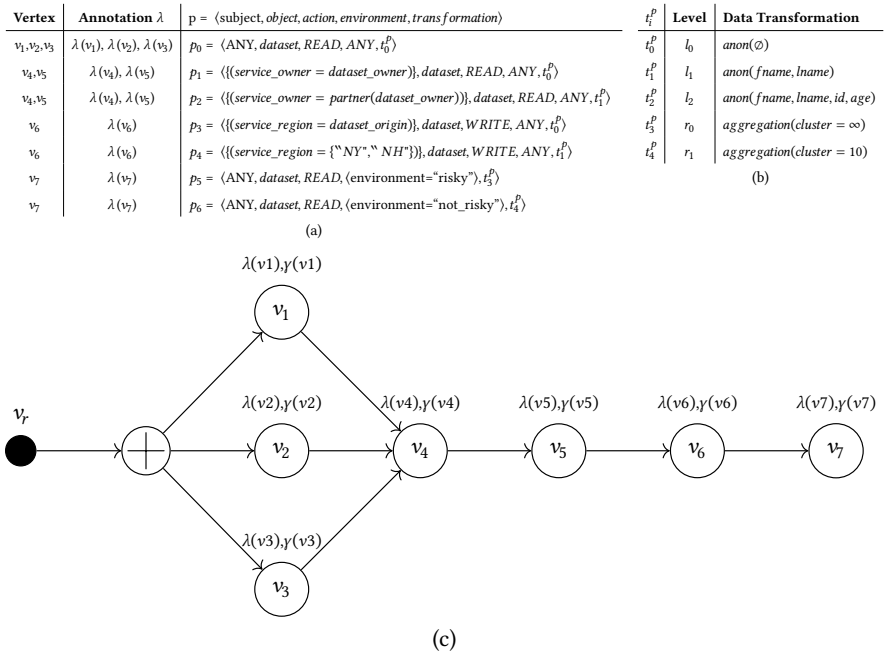
For simplicity but with no loss of generality, we assume that all candidate services meet functional annotation F and that $T^F = t^f$. As a consequence, all candidate services apply the same transformation to the data during the pipeline execution.

Example 4.3.1 (Pipeline Template). Let us consider the reference scenario introduced in Section 4.2. Figure 4.2(c) presents an example of a pipeline template consisting of five stages, each one annotated with a policy in Figure 4.2(a) and corresponding data transformations in Figure 4.2(b).

The first stage in Figure 4.2(c) consists of three parallel vertices v_1, v_2, v_3 for data collection. Data protection annotations $\lambda(v_1)$, $\lambda(v_2)$, $\lambda(v_3)$ refer to policy p_0 in Figure 4.2(a) with an empty transformation t_0^p in Figure 4.2(b). Functional requirements F_1, F_2, F_3 prescribe a URI as input and the corresponding dataset as output.

The second stage in Figure 4.2(c) consists of vertex v_4, merging the three datasets obtained at the first stage. Data protection annotation $\lambda(v_4)$ refers to policies p_1 and p_2 in Figure 4.2(a), which apply different data transformations depending on the relation between the dataset and the service owner. If the service owner is also the dataset owner (i.e., ($service\_owner = dataset\_owner$)), the dataset is not anonymized (t_0^p). If the service owner is a partner of the dataset owner (i.e., ($service\_owner = partner(dataset\_owner)$)), the dataset is anonymized at *level1* (t_1^p). If the service owner has no partner relationship with the dataset owner, no policy applies. Functional requirement F_4 prescribes n datasets as input and the merged dataset as output.

The third stage in Figure 4.2(c) consists of vertex v_5 for data analysis. Data protection annotation $\lambda(v_5)$ refers to policies p_1 and p_2 in Figure 4.2(a),



Vertex	Annotation λ	p = ⟨subject, *object*, *action*, *environment*, *transformation*⟩
v_1,v_2,v_3	$\lambda(v_1)$, $\lambda(v_2)$, $\lambda(v_3)$	p_0 = ⟨ANY, *dataset*, *READ*, *ANY*, t_0^p⟩
v_4,v_5	$\lambda(v_4)$, $\lambda(v_5)$	p_1 = ⟨{(*service_owner* = *dataset_owner*)}, *dataset*, *READ*, *ANY*, t_0^p⟩
v_4,v_5	$\lambda(v_4)$, $\lambda(v_5)$	p_2 = ⟨{(*service_owner* = *partner*(*dataset_owner*))}, *dataset*, *READ*, *ANY*, t_1^p⟩
v_6	$\lambda(v_6)$	p_3 = ⟨{(*service_region* = *dataset_origin*)}, *dataset*, *WRITE*, *ANY*, t_0^p⟩
v_6	$\lambda(v_6)$	p_4 = ⟨{(*service_region* = {"*NY*","*NH*"})}, *dataset*, *WRITE*, *ANY*, t_1^p⟩
v_7	$\lambda(v_7)$	p_5 = ⟨ANY, *dataset*, *READ*, ⟨environment="risky"⟩, t_3^p⟩
v_7	$\lambda(v_7)$	p_6 = ⟨ANY, *dataset*, *READ*, ⟨environment="not_risky"⟩, t_4^p⟩

(a)

t_i^p	Level	Data Transformation
t_0^p	l_0	*anon*(∅)
t_1^p	l_1	*anon*(*fname*, *lname*)
t_2^p	l_2	*anon*(*fname*, *lname*, *id*, *age*)
t_3^p	r_0	*aggregation*(*cluster* = ∞)
t_4^p	r_1	*aggregation*(*cluster* = 10)

(b)

(c)

Figure 4.2: Anonymization policies (a) and data transformations (b) Pipeline Template Example (c)

as for the second stage. Functional requirement F_5 prescribes a dataset as input and the results of the data analysis as output.

The fourth stage in Figure 4.2(c) consists of vertex v_6, managing data storage. Data protection annotation $\lambda(v_6)$ refers to policies p_3 and p_4 in Figure 4.2(a), which apply different data transformations depending on the relation between the dataset and the service region. If the service region is the dataset origin (condition (*service_region=dataset_origin*) in p_3), the dataset is anonymized at level l_0 (t_0^p). If the service region is in a partner region (condition (*service_region*={"*NY*","*NH*"}) in p_4), the dataset is anonymized at level l_1 (t_1^p). Functional requirement F_7 prescribes a dataset as input and the URI of the stored data as output.

The last stage in Figure 4.2(c) consists of vertex v_7, responsible for data visualization. Data protection annotation $\lambda(v_7)$ refers to policies p_5 and p_6

in Figure 4.2(a), which anonymize data according to the environment where the service is executed. A *risky* environment is defined as a region outside the owner or partner facility. If the environment is risky (p_5), the data is anonymized at level r_0 (t_3^p). If the environment is not risky (p_6), the data is anonymized at level r_1 (t_4^p). Functional requirement F_8 prescribes a dataset as input and a data visualization interface (possibly in the form of a JSON file) as output.

4.4 Pipeline Instance

A Pipeline Instance $G'(V', E, \lambda)$ instantiates a Pipeline Template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ by selecting and composing services according to data protection and functional annotations in the template. It is formally defined as follows.

Definition 5 (Pipeline Instance)**.** Let $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ be a pipeline template, a Pipeline Instance $G'(V', E, \lambda)$ is an isomorphic directed acyclic graph where: *i*) $v'_r = v_r$; *ii*) for each vertex v_f modeling a parallel structure, there exists a corresponding vertex v'_f; *iii*) for each $v_i \in V_S$ annotated with policy P_i (label $\lambda(v_i)$) and functional description F_i (label $\gamma(v_i)$), there exists a corresponding vertex $v'_i \in V'_S$ instantiated with a service s'_i, such that:

1) s'_i satisfies data protection annotation $\lambda(v_i)$ in $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$;

2) s'_i satisfies functional annotation $\gamma(v_i)$ in $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$.

Condition 1 requires that each selected service s'_i satisfies the policy requirements P_i of the corresponding vertex v_i in the Pipeline Template, whereas Condition 2 is needed to preserve the process functionality, as it simply states that each service s'_i must satisfy the functional requirements F_i of the corresponding vertex v_i in the Pipeline Template.

We then define a *pipeline instantiation* function that takes as input a Pipeline Template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ and a set S^c of candidate services, and returns as output a Pipeline Instance $G'(V', E, \lambda)$. We note that S^c is partitioned in different sets of services S^c_i, one for each vertex $v_i \in V_S$. Recall from

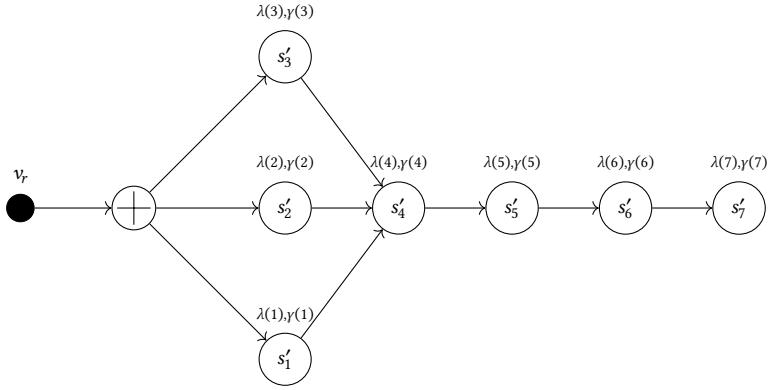


Figure 4.3: Service composition instance

Section 4.3.3 that all candidate services meet the functional annotation in the template, meaning that Condition 2 in Definition 5 is satisfied for all candidate services. The pseudocode of the pipeline instantiation process is presented in Figure 4.4. The Pipeline Instance is generated by traversing the Pipeline Template with a breadth-first search algorithm (line 4-10), starting from the root vertex v_r. Then, for each vertex v_f in the pipeline template, the corresponding vertex v'_f is generated (line 5). Finally, for each vertex $v_i \in V_S$, a two-step approach is executed as follows.

i) Filtering Algorithm – It checks whether profile prf_j of each candidate service $s_j \in S_i^c$ satisfies at least one policy in P_i (line 16). If yes, service s_j is compatible, otherwise it is discarded (line 17). The filtering algorithm finally returns a subset $S'_i \subseteq S_i^c$ of compatible services for each vertex $v_i \in V_S$ (line 19).

ii) Selection Algorithm – The selection algorithm selects one service s'_i for each set S'_i of compatible services, which instantiates the corresponding vertex $v'_i \in V'$ (line 8-9). There are many ways of choosing s'_i, Section 5.3 presents our approach based on the maximization of data *quality Q*.

When all vertices $v_i \in V$ in $G^{\lambda,\gamma}$ have been visited, the Pipeline Instance

Table 4.1: Instance example

Vertex→Policy	Candidate	Profile	Filtering	Instance
$v_5 \rightarrow p_1, p_2$	s_{51}	service_owner = "CT"	✓	✓
	s_{52}	service_owner = "NY"	✓	✗
	s_{53}	service_owner = "CA"	✗	✗
$v_6 \rightarrow p_3, p_4$	s_{61}	service_region = "CA"	✗	✗
	s_{62}	service_region = "CT"	✓	✓
	s_{63}	service_region = "NY"	✓	✗
$v_7 \rightarrow p_5, p_6$	s_{71}	visualization_location = "CT_FACILITY"	✓	✓
	s_{72}	visualization_location = "CLOUD"	✓	✗

(a) Valid Instance example

Candidate	Ranking
s_{51}	1
s_{52}	2
s_{53}	–
s_{61}	–
s_{62}	1
s_{63}	2
s_{71}	1
s_{72}	2

(b) Best Quality Instance example

G' is generated (line 11), with a service instance s_i' for each $v_i' \in V'$. Vertex v_i' is annotated with policies in P_i according to λ, because policies in P_i are evaluated and enforced at runtime, only when the pipeline instance is triggered and before any service is executed. When policy evaluation returns *true*, data transformation $T^P \in P_i$ is applied, otherwise a default transformation that removes all data is applied.

Example 4.4.1 (Pipeline Instance). Let us consider a subset $\{v_5, v_6, v_7\}$ of the pipeline template $G^{\lambda,\gamma}$ in Example 4.3.1.

As presented in Table 4.1(a), each vertex is labeled with policies (column *Vertex→Policy*) and is associated with different candidate services (column *Candidate*) and corresponding profile (column *Profile*). The filtering algorithm matches each candidate service profile against the policies annotating the corresponding vertex (Table 4.2). It returns the set of services whose profile satisfies a policy (column *Filtering*): *i*) for vertex v_5, the filtering algorithm produces the set $S_1 = \{s_{51}, s_{52}\}$. Assuming that the dataset owner is "CT", the service profile of s_{51} matches p_1 and s_{52}'s one matches p_2. For s_{53}, there is no policy match and, thus, it is discarded; *ii*) for vertex v_6, the filtering algorithm returns the set $S_2' = \{s_{62}, s_{63}\}$. Assuming that the dataset region is "CT", the service profile of s_{62} matches p_5 and the one of s_{63} matches p_6. For s_{61}, there is no policy match and, thus, it is discarded; *iii*) for vertex v_7, the filtering algorithm returns the set $S_3' = \{s_{71}, s_{72}\}$. Since policy p_7 matches against any subject, the filtering algorithm keeps all services.

For each vertex v_i', we select a matching service s_j' from S_i' and incorpo-

rate it into a valid instance. For instance, we select s_{51} for v_5; s_{62} for v_6, and s_{71} for v_7 as depicted in Table 4.1(a) (column *instance*). We note that to move from a valid to an optimal instance, it is mandatory to evaluate candidate services based on specific quality metrics that reflect their impact on data quality, as discussed in the following of this chapter.

4.5 Chapter Summary

This chapter presented the core abstractions that enable quality-aware, privacy-preserving service composition in distributed data processing environments. *Pipeline Definition* provides the system model for capturing the multi-dimensional requirements of service-oriented data processing. *Pipeline Template* formalizes the specification layer through policy and functional annotations, enabling declarative definition of data protection requirements and service capabilities while maintaining separation between abstract requirements and concrete implementations. *Pipeline Instance* materializes these specifications through service selection and composition, guided by filtering algorithms that ensure policy compliance and optimization heuristics that maximize data quality across the entire pipeline.

INPUT
$G^{\lambda,\gamma}$: Pipeline Template
S^c: Candidate Services

OUTPUT
G': Pipeline Instance

Instantiate_Pipeline($G^{\lambda,\gamma}$, S^c)

```
1  /* Initialize the pipeline instance*/
2  G' = {};
3  /* Traverse the pipeline template using BFS*/
4  for each v in G^{λ,γ}
5      v' = Generate_Vertex(v);
6      G' = G' ∪ v';
7      S' = Filter_Services(S^c[v], v.policies);
8      selectedService = Select_A_Service(S');
9      v'.service = selectedService;
10 endfor;
11 return G';
```

```
12 Filter_Services(S^c[v], policies)
13 /* Filter candidate services based on policies*/
14 S' = {};
15 for each service s in S^c[v]:
16     if s.profile satisfies any policy:
17         S' = S' ∪ s;
18     endif;
18 endfor;
19 return S';
```

Figure 4.4: Pseudocode of the pipeline instantiation process.

5

Dynamic Pipeline Instance Optimization

Building upon the policy-driven service selection framework and template-instance architecture presented in previous chapters, this chapter addresses the computational challenges of optimizing service compositions in distributed data processing environments.

This chapter presents our sliding-window optimization approach that enables scalable quality-privacy optimization for complex data processing pipelines. We introduce metrics for evaluating quality-privacy trade-offs in service compositions and develop heuristic algorithms that provide near-optimal solutions with computational efficiency suitable for real-time pipeline instantiation. Our goal is to generate pipeline instances with maximum quality while addressing data protection requirements throughout execution. We first discuss the quality metrics used to measure and monitor data quality, then prove that the problem of generating optimal pipeline instances is NP-hard, and finally introduce a parametric heuristic designed to tackle the computational complexity of enumerating all possible service combinations. The heuristic approximates optimal paths for service interactions and transformations, focusing on understanding quality changes introduced during transformation processes within complex pipelines consisting of numerous vertices and candidate services.

5.1 Quality Metrics

Ensuring data quality is mandatory to implement service-based data pipelines that provide accurate results and decision-making along the whole pipeline execution. Different definition of quality exists (e.g., [98, 107]) according to different dimensions such as completeness, timeliness, and accuracy, to name but a few. Quality metrics measure the data quality preserved at each step of the pipeline according to the selected quality dimensions, and can be classified as *quantitative* or *qualitative*.

Quantitative metrics monitor the amount of data lost during data transformations to model the quality difference between datasets X and Y. Qualitative metrics evaluate changes in the properties of datasets X and Y. For instance, qualitative metrics can measure the changes in the statistical distribution of the two datasets.

It is important to note that providing a taxonomy of all possible dimensions and metrics is beyond the scope of this work. In our future work, we will examine the conceptual and practical aspects of classifying and defining relevant quality metrics, such as timeliness and consistency, as well as their impact on our methodology.

5.1.1 Quantitative metric

We propose a quantitative metric M_J based on the Jaccard coefficient that assesses the similarity between two datasets. The Jaccard coefficient is defined as follows [81]:

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

where X and Y are two datasets of the same size.

The coefficient is calculated by dividing the cardinality of the intersection of two datasets by the cardinality of their union. It ranges from 0 to 1, with 0 indicating no similarity (minimum quality) and 1 indicating complete similarity (maximum quality) between the datasets. It has several advantages. Unlike other similarity measures, such as Euclidean distance, it is not affected by the magnitude of the values in the dataset. It is suitable for datasets with categorical variables or nominal data, where the values do not

have a meaningful numerical interpretation.

Metric M_J extends the Jaccard coefficient with weights that model the importance of each element in the dataset as follows:

$$M_J(X,Y) = \frac{\sum_{i=1}^{n} w_i(x_i \cap y_i)}{\sum_{i=1}^{n} w_i(x_i \cup y_i)}$$

where $x_i \in$X ($y_i \in$Y, resp.) is the i-th feature of dataset X (Y, resp.), and w_i the weight modeling the importance of the i-th feature.

It is computed by dividing the cardinality of the intersection of two datasets by the cardinality of their union, weighted by the importance of each feature in the datasets. It provides a more accurate measure of similarity.

5.1.2 Qualitative Metric

We propose a qualitative metric M_{JSD} based on the Jensen-Shannon Divergence (JSD) that assesses the similarity (distance) between the probability distributions of two datasets.

JSD is a symmetrized version of the KL divergence [43] and is applicable to a pair of statistical distributions only. It is defined as follows:

$$JSD(X,Y) = \frac{1}{2}\left(KL(X||M) + KL(Y||M)\right)$$

where X and Y are two distributions of the same size, and M=0.5*(X+Y) is the average distribution. JSD incorporates both the KL divergence from X to M and from Y to M.

To make JSD applicable to datasets, where each feature in the dataset has its own statistical distribution, metric M_{JSD} applies JSD to each column of the dataset. The obtained results are then aggregated using a weighted average, thus enabling the prioritization of important features that can be lost during the policy-driven transformation in Section 5.3, as follows:

$$M_{JSD} = 1 - \sum_{i=1}^{n} w_i \cdot \mathrm{JSD}(x_i, y_i)$$

where $\sum_{i=1}^{n} w_i$=1 and each JSD(x_i, y_i) accounts for the Jensen-Shannon Divergence computed for the i-th feature in datasets X and Y. It ranges from 0 to 1, with 0 indicating no similarity (minimum quality) and 1 indicating complete similarity (maximum quality) between the datasets.

M_{JSD} provides a weighted measure of similarity, which is symmetric and accounts for the contribution from both datasets and specific features. It can compare the similarity of the two datasets, providing a symmetric and normalized measure that considers the overall data distributions.

5.1.3 Entropy Metric (M_H)

We propose a metric M_H based on Shannon entropy that assesses the information content preserved in datasets after privacy-preserving transformations.

Shannon entropy is a fundamental concept in information theory that measures the uncertainty or information content in a probability distribution [73]. For a dataset feature X with probability distribution $P(X)$, it is defined as:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where p_i represents the probability of the i-th unique value in feature X, and n is the number of distinct values.

To make Shannon entropy applicable to multi-feature datasets, metric M_H applies entropy computation to each feature of the dataset. The obtained results are then aggregated using feature-wise comparison, thus enabling the measurement of information loss across all dataset attributes, as follows:

$$M_H = 1 - \frac{1}{|F|} \sum_{j=1}^{|F|} \left| \frac{H(X_j) - H(Y_j)}{H(X_j)} \right|$$

where $|F|$ is the number of features, X_j and Y_j represent the j-th feature in the original and transformed datasets respectively. It ranges from 0 to 1, with 0 indicating no similarity (minimum quality) and 1 indicating complete similarity (maximum quality) between the datasets.

The entropy metric is particularly valuable when dealing with categor-

ical data and discrete attributes, as it directly measures the uncertainty and diversity within each feature space. Unlike other similarity measures, it captures the information-theoretic properties of the data distribution. It is suitable for assessing the impact of generalization and anonymization operations that affect the diversity of attribute values.

5.2 Pipeline Quality

Metrics M_J, M_{JSD}, and M_H contribute to the calculation of the pipeline quality Q as follows.

Definition 6 (*Pipeline Quality*)**.** Given a metric $M \in \{M_J, M_{JSD}, M_H\}$ modeling data quality, the pipeline quality Q is equal to $\sum_{i=1}^{|S|} M_{ij}$, with M_{ij} the value of the quality metric computed at each vertex $v'_i \in V'_S$ of the pipeline instance G' with respect to the service instance s'_j, with $1 \leq j < |S^c_i|$.

We also use the notation Q_{ij}, with $Q_{ij}=M_{ij}$, to specify the *quality* at vertex $v'_i \in V'_S$ of G' for service s'_j. The process of computing a pipeline instance (Definition 5) with maximum quality Q can be formally defined as follows.

Definition 7 (Max-Quality Problem)**.** Given a pipeline template $G^{\lambda,\gamma}$ and a set S^c of candidate services, find a max-quality pipeline instance G' such that:

- G' satisfies conditions in Definition 5,
- $\nexists$ a pipeline instance G'' that satisfies conditions in Definition 5 and such that quality $Q(G'') > Q(G')$, where $Q(\cdot)$ is the pipeline quality in Definition 6.

The Max-Quality problem is a combinatorial selection problem and is NP-hard, as stated by Theorem 5.2.1. However, while the overall problem is NP-hard, the filtering step of the process, is solvable in polynomial time. It can be done by iterating over each vertex and each service, checking if the service matches the vertex policy. This process takes polynomial time complexity $O(|V_S| * |S|)$.

Theorem 5.2.1. The Max-Quality problem is NP-Hard.

Proof: The proof is a reduction from the multiple-choice knapsack problem (MCKP), a classified NP-hard combinatorial optimization problem, which is a generalization of the simple knapsack problem (KP) [66]. In the MCKP problem, there are t mutually disjoint classes $N_1, N_2, \dots, N_t$ of items to pack in some knapsack of capacity C, class N_i having size n_i. Each item $j \in N_i$ has a profit p_{ij} and a weight w_{ij}; the problem is to choose one item from each class such that the profit sum is maximized without having the weight sum exceed C.

The MCKP can be reduced to the Max-Quality Pipeline Instantiation Process in polynomial time, with $N_1, N_2, \dots, N_t$ corresponding to the sets of compatible services $S_1^c, S_2^c, \dots, S_u^c$, with $t=u$ and n_i also the size of each set S_i^c. The profit p_{ij} of item $j \in N_i$ corresponds to quality Q_{ij} computed for each candidate service $s_j \in S_i^c$, while w_{ij} is uniformly 1 (thus, C is always equal to the cardinality of V_C). It is evident that the solution to one problem is also the solution to the other (and vice versa). Since the reduction can be done in polynomial time, the Max-Quality problem is also NP-hard.

Example 5.2.1 (Max-Quality Pipeline Instance)**.** We extend Example 4.4.1 with the selection algorithm in Section 4.4 built on pipeline quality Q. The selection algorithm is applied to the set S'_* of compatible services and returns three service rankings, one for each vertex v_4, v_5, v_6, according to quality metrics M_J (measuring the amount of preserved data after anonymization), M_{JSD} (measuring distributional similarity), and M_H (measuring entropy-based information content). The ranking is presented in Table 4.1(b), according to the transformation function in the corresponding policies. We assume that the more restrictive the transformation function (i.e., it anonymizes more data), the lower is the service position in the ranking. For example, s_{11} is ranked first because it anonymizes less data than s_{12} and s_{13}, that is, $Q_{11} > Q_{12}$ and $Q_{11} > Q_{13}$. The same applies to the ranking of s_{22} and s_{23}. The ranking of s_{31} and s_{32} is affected by the environment state at the time of the ranking. For example, if the environment where the visualization is performed is a CT facility, then s_{31} is ranked first and s_{32} second because the facility is considered less risky than the cloud, and $Q_{31} > Q_{32}$.

5.3 Heuristic

We design and implement a heuristic algorithm built on a *sliding window* for computing the pipeline instance maximizing quality Q. At each iteration i, a window of size $|w|$ selects a subset of vertices in the pipeline template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$, from vertices at depth i to vertices at depth $|w|+i-1$. Service filtering and selection in Section 4.4 are then executed to maximize quality Q_w in window w. The heuristic returns as output the list of services instantiating all vertices at depth i. The sliding window w is then shifted by 1 (i.e., $i=i+1$), and the filtering and selection process is executed until $|w|+i-1$ is equal to length l (max depth) of $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$, that is, the sliding window reaches the end of the template. In the latter case, the heuristic instantiates all remaining vertices and returns the pipeline instance G'. This strategy ensures that only services with low information loss are selected at each step, maximizing the pipeline quality Q.

The choice of window size represents a fundamental trade-off between optimization quality and computational complexity. A window of size 1 (Figure 5.2) provides local optimization with minimal computational overhead, selecting services independently for each vertex. Conversely, larger windows such as size 3 (Figure 5.4) or size 5 (Figure 5.6) enable better global optimization by considering multiple consecutive vertices simultaneously, though at increased computational cost. The red dashed rectangles in these figures illustrate how the optimization scope expands with larger window sizes, while the different line styles (thick for selected services, dashed for candidates) show the progressive service selection process.

The pseudocode of the heuristic algorithm is presented in Figure 5.1. The function **SlidingWindowHeuristic** implements our heuristic; it takes the pipeline template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ and the window size $|w|$ as input and returns the pipeline instance G' and corresponding metric M as output. The function **SlidingWindowHeuristic** retrieves the optimal service combination composing G', considering the candidate services associated with each vertex in $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ and the constraints (policies) in *verticesList*.

It iterates all sliding windows w step 1 until the end of the pipeline template is reached (**for cycle** in line 2). Adding the service(s) selected at step i to G' by the function **SelectService** (defined in line 10).

The function **SelectService** takes as input index i representing the starting depth of the window and the corresponding window size $|w|$. It initializes the best combination of services to *empty* (line 11). It iterates through all possible combinations of services in the window using the Cartesian product of the service lists (**for cycle** in lines 13-16). If the current combination has quality metric $M(G'_w)$ higher than the best quality metric $M(G^*_w)$, the current combination G'_w updates the best combination G^*_w (lines 14-15). This combinatorial selection process is visualized in Figures 5.3 and 5.5, where the thick lines indicate the winning service combinations within each optimization window. The function **SelectService** then checks whether it is processing the last window (line 18). If yes, it returns the best combination G^*_w (line 19). Otherwise, it returns the first service in the best combination G^*_w (line 21).

Within each window, the function **SlidingWindowHeuristic** finally iterates through the selected services to calculate the total quality metric M (**for cycle** in lines 6-8). This metric is updated by summing the quality metrics of the selected services. The function concludes by returning the best pipeline instance G' and the corresponding quality metric M (line 9).

5.4 Chapter Summary

This chapter addressed the computational challenges of optimizing service compositions in data processing pipelines while balancing quality preservation and privacy protection requirements. We introduced quality metrics to evaluate quantitative and qualitative aspects of data transformations. Our sliding window optimization heuristic provides a more computationally efficient solution to the NP-hard problem of optimal service selection.

INPUT
$G^{\lambda,\gamma}$: Pipeline Template
$|w|$: Window Size

OUTPUT
G': Pipeline Instance
M: Quality Metric

Sliding_Window_Heuristic ($G^{\lambda,\gamma}$, $|w|$)

```
1  /* For each window frame choose the best combination of services*/
2  for i = 0 to l - |w|+ 1;
3      G' = G' ∪ Select_Service(j, |w|);
4  endfor;

5  /* Calculate the total quality metric*/
6  for j = 0 to |V'_S|;
7      M=M+M(s'_j);
8  endfor;

9  return G', M;
```

Select_Service (j,$|w|$)

```
11 G*_w= best combination (empty);
12 /* Select the best combination of services*/
13 for G'_w∈ ⨂_{k=j}^{j+|w|−1} verticesList[k]
14     if M(G'_w) < M(G*_w)
15         G*_w= G'_w
16 endfor;

17 /* If it is the last window frame, return all services */
18 if isLastWindowFrame()
19   return G*_w
20 else
21   return G*_w[0]
```

Figure 5.1: Pseudocode of the sliding window heuristic algorithm.

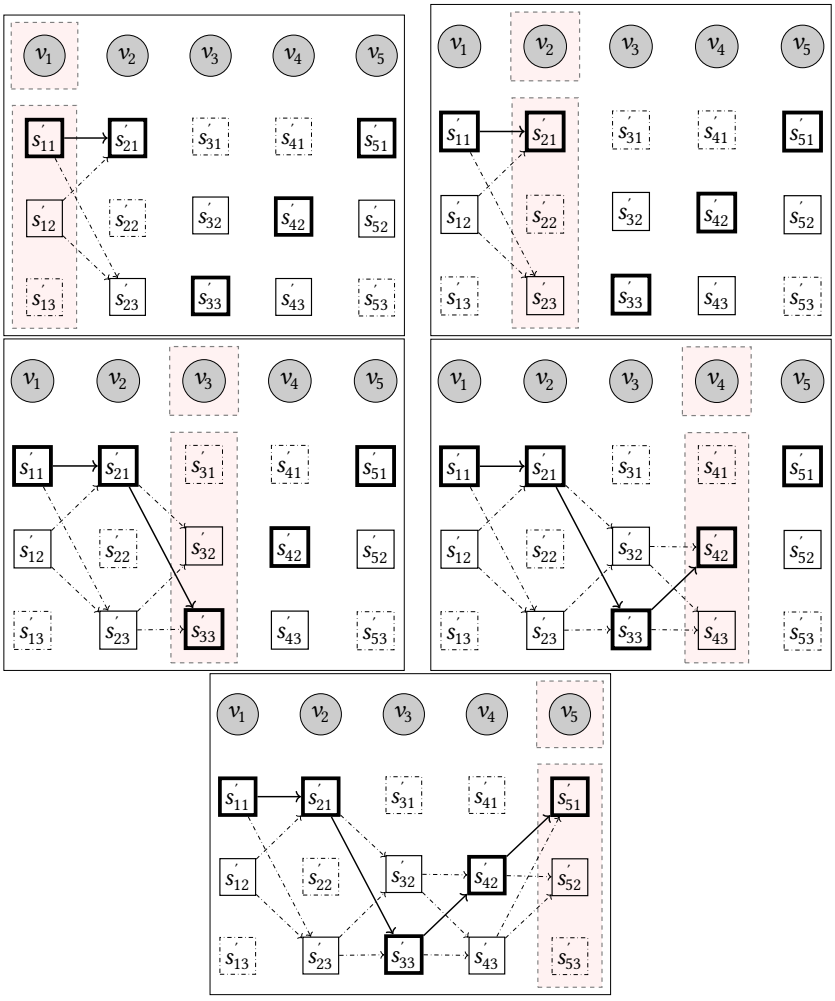


Figure 5.2: Pipeline instantiation with window size 1. Each step optimizes service selection for individual vertices independently, providing maximum flexibility but potentially missing global optimization opportunities.

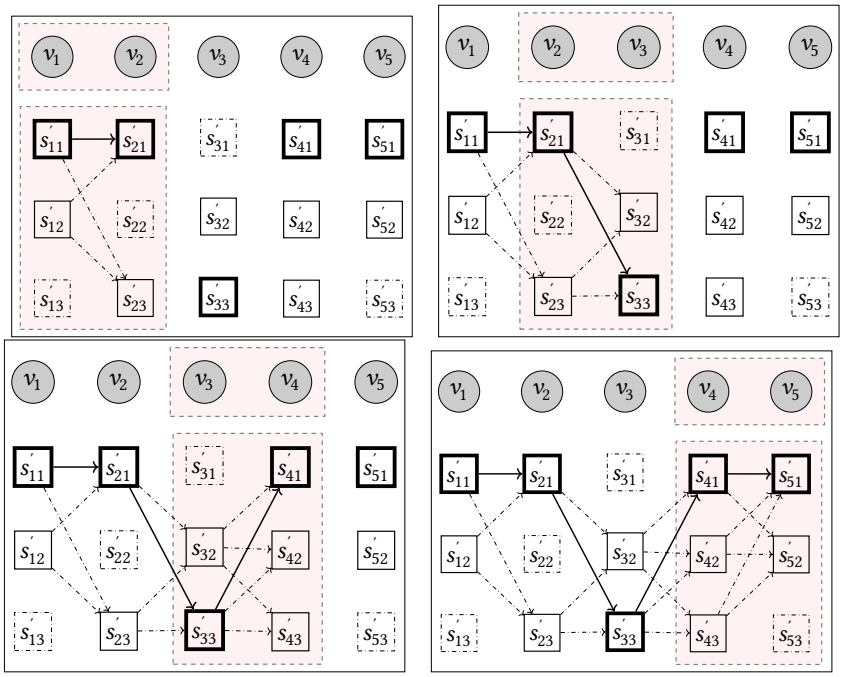


Figure 5.3: Pipeline instantiation with window size 2. The optimization considers pairs of consecutive vertices, enabling local optimization while maintaining computational tractability.

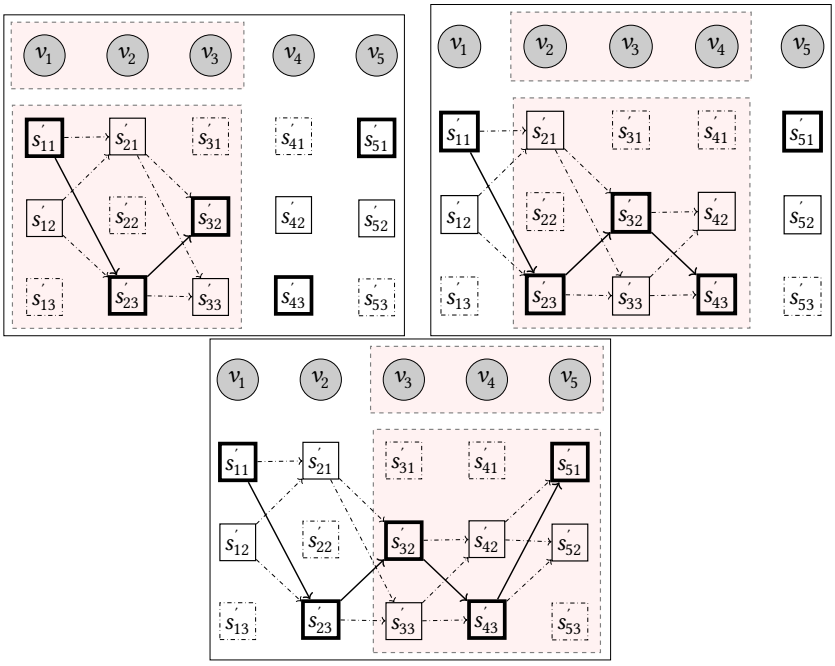


Figure 5.4: Pipeline instantiation with window size 3. Three consecutive vertices are optimized together, balancing between local and global optimization with moderate computational complexity.

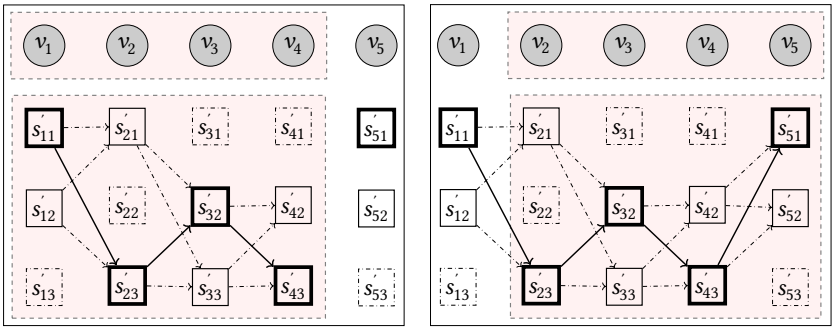


Figure 5.5: Pipeline instantiation with window size 4. Larger window enables better global optimization by considering four consecutive vertices simultaneously, with increased computational requirements.

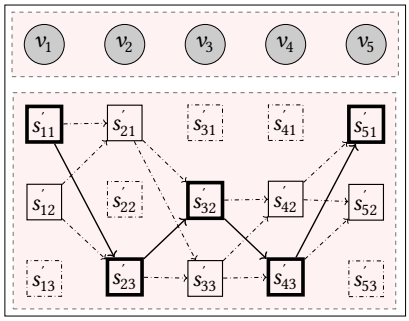


Figure 5.6: Pipeline instantiation with window size 5. Global optimization considers all vertices simultaneously, providing optimal solutions at the cost of maximum computational complexity.

6

Data Governance Framework: Implementation Choices and Experimental Evaluation

Having established the theoretical foundations of the data governance framework and its optimization algorithms in previous chapters, this chapter focuses on the practical implementation and empirical validation of the proposed approach. We present a configurable tool that operationalizes the sliding window heuristic for quality-aware service selection in data processing pipelines, enabling systematic evaluation across diverse scenarios. The chapter demonstrates how the framework's abstract concepts translate into concrete implementation choices and validates their effectiveness through comprehensive experimental evaluation.

The presentation is organized into three main components. First, we describe the architecture and configuration options of the data quality assessment tool, which provides the computational infrastructure for instantiating and benchmarking pipeline instances. Second, we detail the implementation of quality metrics that quantify different dimensions of data completeness and information preservation throughout pipeline execution. Finally, we present experimental results across multiple datasets and filtering strategies, demonstrating that the sliding window heuristic achieves near-optimal performance while maintaining computational feasibility for real-world deployments.

6.1 Data Quality Assessment Tool

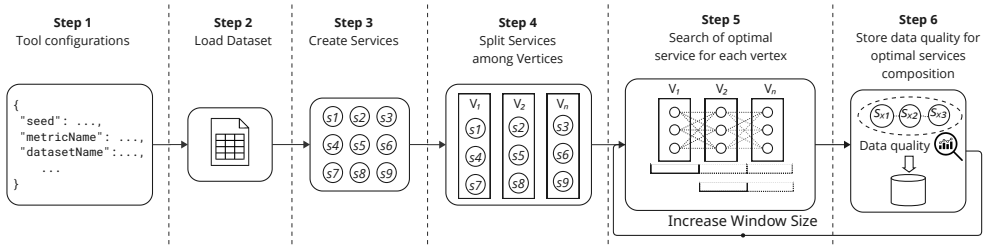


Figure 6.1: Tool: Execution flow

We developed a framework as a configurable tool that builds and benchmarks pipeline instances according to our sliding window approach. The tool is publicly available at `https://github.com/SESARLab/data-quality-simulator`. In the remainder of this section, we provide an overview of the tool configurations and process.

6.1.1 Tool Configurations

The tool offers a range of configurations that customize the pipeline components, supporting a wide coverage of possible evaluation scenarios and guaranteeing full reproducibility of all results, as follows.

Dataset – The dataset used for the data quality evaluation.

Seed – An integer value that ensures reproducibility and pseudo-randomization in simulations.

MaxWindowSize – The sliding window size in a simulation ranges from 1 to *MaxWindowSize*.

Metrics – The quality metrics that capture different dimensions or nuances of the data. Ad hoc quality metrics can be added according to the specific scenario with no impact on the tool operation. The tool is pre-instantiated with three metrics evaluating data completeness in terms of the data loss introduced throughout the pipeline execution: *i)*

a quantitative metric (MJ) based on the Jaccard coefficient [81], which assesses the similarity between two datasets; *ii)* a qualitative metric (MJSD) based on the Jensen-Shannon Divergence [43], which evaluates the similarity between the probability distributions of two datasets; *iii)* an entropy-based metric (ME) using Shannon Entropy to measure information loss during the pipeline execution.

The tool also supports configurable data protection based on policies. The policies operate on data categories according to the following options:

FilteringType – It specifies the strategy for data filtering and offers three options: *row,* where policies operate on a subset of dataset rows; *column,* where policies operate on a subset of the dataset columns; *mixed,* where policies operate on both rows and columns.

RowLowerBound* and *RowUpperBound – They determine the number of rows to anonymize when the *FilteringType* is *row* or *mixed*. Before a data protection policy is applied, our tool chooses the rows to filter out. To this aim, it: *i)* randomly chooses a decimal number D in the range $[RowLowerBound, RowUpperBound]$, with $0<RowLowerBound\leq RowUpperBound\leq 1$, representing the percentage of *plain* data categories; *ii)* randomly picks a column C; *iii)* with $Cat(C)$ being the total number of categories in C, calculates the number $Cat_a(C)$ of categories to protect using the formula $Cat(C)^*(1-D)))$; *iv)* randomly selects the $Cat_a(C)$ categories to protect and stores the positions of the values belonging to these categories; *v)* protect the rows in the stored positions.

As an example, let us consider the dataset presented in Table 6.1. Assuming that $FilteringType=\text{"}row\text{"}$, $C=\text{"}Cat\_col\text{"}$, $RowLowerBound$=0.6, $RowUpperBound$=0.8, and D=0.75, the percentage of categories to anonymize is 25%. Since there are four categories in C, from *cat1* to *cat4*, $Cat_a(C) = 1$. Assuming the selected category is *cat3*, rows with *Id* 2 and 3 are filtered out. When the *FilteringType* is set to *column*, these options determine the number of values to anonymize within each selected column.

Table 6.1: Sample dataset for row filtering

Id	Cat_col	Age
0	cat1	10
1	cat2	20
2	cat3	30
3	cat3	30
4	cat4	40
5	cat4	50

ColumnLowerBound* and *ColumnUpperBound – They determine the number of columns to anonymize when the *FilteringType* is *column* or *mixed*. They drive the column filtering process in the same way that *RowLowerBound* and *RowUpperBound* do for the row filtering process.

The tool finally supports the configurations of the pipeline template, customizing the cardinality of candidate services and the length of the service pipeline.

6.1.2 Tool Process

The tool instantiates a pipeline template in a pipeline instance, driving the selection, composition, and execution of the corresponding services. While integrating services into a pipeline instance, the tool evaluates their impact on quality metrics. This benchmarking process enables users to compare the performance of multiple service compositions, identifying the optimal (local or global) composition of services that maximizes a quality metric.

Following the configurations in Section 6.1.1, the tool enables users to tailor simulations to diverse scenarios. These options include the number of vertices, services, policies, datasets, and metrics, shaping the specific pipeline templates and instances. By simulating exhaustive and heuristic methods,

the tool highlights the trade-offs between computational feasibility and quality and provides actionable insights into real-world pipeline optimization.

Figure 6.1 shows the high-level process of our tool. The tool takes as input *i)* the target dataset and *ii)* all configurations in Section 6.1.1, and returns as output *i)* the pipeline maximizing the data quality and *ii)* all quality metrics calculated on valid pipeline instances, that is, those instances addressing data protection requirements in policies. The process consists of pipeline template definition (Steps 1–4) and pipeline instance generation (Steps 5 and 6).

Step 1. The tool is configured according to configurations in Section 6.1.1. The user can specify them using CLI arguments or with JSON files.

Step 2. The user specifies the dataset, which is first pre-processed to optimize the memory usage and then loaded into memory.

Step 3. The tool generates $|S|*|V|$ candidate services and corresponding profiles, with $|S|$ the number of services assigned to each of the $|V|$ vertices in the pipeline template.

Step 4. Services are evenly split into $|V|$ groups, where each group is assigned to a vertex.

Step 5. The sliding window algorithm evaluates the impact of each candidate service s on the overall data quality in terms of the considered metric. The algorithm selects one service s_i for each vertex $V_i \in \{V_n, V_{n+1}, ..., V_{n+|w|-1}\}$ that maximizes the metric value, where $|w|$ is the size of the sliding window and $\{V_n, V_{n+1}, ..., V_{n+|w|-1}\}$ the portion of the pipeline template under evaluation according to $|w|$. The service selected for V_n is added to the pipeline instance, the window shifts one vertex to the right, and the service selection process is repeated. The process continues until the window reaches the end of the pipeline, that is, $n + |w| - 1$ is equal to the pipeline length $|V|$. All remaining vertices are finally instantiated, building the complete pipeline instance.

Step 6. The service composition at Step 5 represents the pipeline instance that preserves the highest data quality according to the chosen quality metric. The tool stores the data quality (i.e., the metric value) and the corresponding execution parameters in a database. The algorithm returns to step 5, where

the selection process restarts with a sliding window size increased by one.

The algorithm terminates when the sliding window size reaches $|V|$ or *MaxWindowSize*. In the following, we denote an algorithm execution as *simulation*.

6.1.3 Category-based sampling

Category-based sampling enhances the applicability of data protection policies by grouping column values into predefined categories. This ensures that anonymized values belong to the same category (or categories if multiple ones are anonymized), which reflects real-world scenarios where data protection selectively removes specific information.

The process involves pre-processing the dataset by categorizing column values. The number of categories for a column is defined by some general rules based on the nature of the column itself:

Categorical and ordinal columns encode discrete variables. Categorical columns represent unordered nominal data. Ordinal columns represent ordered categorical data with hierarchical relationships. A policy operates on specific elements (e.g., column lastName with value "Doe"). When the domain is limited to a few values, we initially identify the unique values (e.g., column sex with value "male" or "female"), then a category is created for each of them.

Numerical columns contain numerical values such as integers or floats. A policy operates on specific elements (e.g., $frequency{\neq}0$) or ranges (e.g., $age{\geq}18$). Numerical columns are partitioned into equal-width buckets according to the following procedure:

i) Identify the minimum (val_{min}) and the maximum (val_{max}) values;

ii) Identify the number of unique values U;

iii) Set the total number of buckets B as a pseudo-randomly generated integer in the interval $[1, U]$ for integer values, and in the interval $[1, min(U, \sqrt{N})]$, where N is the number of values in the column, for float values;

iv) Create B equal-width buckets, each containing from 0 to N−1 elements: $[V_{min}, V_{b1}[, [V_{b1}, V_{b2}[, ..., [V_{bn}, V_{max}]$.

This imbalanced bucketing aligns well with real-life protection strategies. The policy operates without knowledge of the buckets' content, just like a real-world policy lacks awareness of the values it anonymizes. As an example, let us suppose column *age* contains integers from 1 to 100, and B=10. The created buckets are $[1, 10], [11, 20], ..., [91, 100]$. It is important to emphasize that a bucket may contain 0 elements; for instance, there might be no individuals aged between 81 and 90 in the dataset. Conversely, a bucket could contain nearly all the elements except one, which might occur when there is an outlier and most values are condensed in a narrow interval.

6.1.4 Metrics

Our simulations use the qualitative, quantitative, and entropy-based metrics described in Section 6.1.1. The qualitative metric based on Jensen-Shannon divergence captures the global statistical similarity between the original and anonymized datasets. The quantitative metric based on the Jaccard coefficient captures the similarity between the original and anonymized datasets based on shared elements without considering element frequency or count. They are presented in Algorithm 1 and Algorithm 2, and described in detail in [79].

Algorithm 1: Qualitative metric implementation

INPUT:
df1: Original dataset
df2: Processed dataset
OUTPUT:
qualitativeScore: Qualitative similarity score

```
ds ← [];
ws ← [];
foreach col in df1.columns do
    df1_value_counts ← df1[col].value_counts(normalize=True,
      dropna=False);
    df2_value_counts ← df2[col].value_counts(normalize=True,
      dropna=False);
    combined_index ←
      df1_value_counts.index.union(df2_value_counts.index,
      sort=True);
    p ← df1_value_counts.reindex(combined_index, fill_value=0.0);
    q ← df2_value_counts.reindex(combined_index, fill_value=0.0);
    ds.append(distance.jensenshannon(p, q));
    ws.append(df1[col].nunique() / df1[col].count());
end
weight_sum ← sum(ws);
ws ← [w/weight_sum for w in ws];
return 1 - np.average(ds, weights=ws);
```

Algorithm 2: Quantitative metric implementation

INPUT:
df1: Original dataset
df2: Processed dataset
OUTPUT:
quantitativeScore: Quantitative similarity score

```
df1_non_none_cells ← df1.count().sum();
df2_non_none_cells ← df2.count().sum();
return df2_non_none_cells / df1_non_none_cells;
```

The entropy-based metric is built on the entropy of a dataset. In particular, in the context of probability distributions, entropy measures the amount of uncertainty in the outcomes of a distribution. The metric first computes the Shannon entropy [73] of each column as $H(X)=-\sum_i p_i \log_2 p_i$. Then, the entropy of all columns is averaged to calculate the overall dataset entropy.

By adding an entropy-based metric, we can gain insights into how individual attributes and internal data variability are affected by data protection requirements. Entropy measures the information loss and attribute-level diversity in the original and anonymized datasets.

Each metric provides a different view of how data protection requirements impact the data quality of each pipeline instance in terms of data completeness. Each metric measures the data quality Q of a pipeline (see Algorithm 3) as a metric value in [0, 1], where 0 denotes full data loss and 1 denotes full data preservation. The data quality Q of the pipeline instance is calculated by comparing the dataset given as input to the pipeline and the dataset returned as output by the pipeline according to one or more of the available metrics.

Algorithm 3: Metric evaluation in the sliding window algorithm

```
   INPUT:
   window: Current window
   metricName: Quality metric name
   originalDataset: Input dataset
   OUTPUT:
   bestComposition: Optimal service composition
1  servicesCompositions ← window.cartesianProduct();
2  foreach sc in servicesCompositions do
3  |   pipeline ← Pipeline(services=sc, metricName=metricName);
4  |   execution ← pipeline.run(originalDataset);
5  |   if execution.metric > bestComposition.metric then
6  |   |   bestComposition ← sc;
7  |   end
8  end
9  return bestComposition;
```

6.1.5 Anonymized Values Management

After a policy is applied, the dataset has the same number of rows and columns, with a part of the cell values assigned with the *empty* placeholder. This approach supports multiple anonymizations of the same cell. In other words, we consider the scenario where the policies of two or more consecutive services apply to an (partially) overlapped subset of the dataset.

Protecting through the *empty* placeholder substantially affects the metrics' implementation too. For the qualitative metric, changes in the probability distribution of *empty* placeholders are treated as changes in any other column distribution. For the quantitative metric, we track the number of cells with value $\neq$*empty*. This represents the amount of plain information in the dataset. The entropy-based metric accounts for the probability of *empty* placeholders in the same way as it does for the probability of the other values.

6.2 Experimental Evaluation

This section presents the experimental evaluation of our tool, analyzing data quality variations under several different conditions. The tool examines the impact of various datasets and configurations on data quality.

Each experiment aggregates results from 20 simulations with different seed initializations. The graphs present the window size on the x-axis and the corresponding data quality Q or normalized data quality Q/Q^* on the y-axis, where Q is the quality retrieved by our sliding window heuristic and Q^* is the optimum quality retrieved by the exhaustive approach. Normalized data quality Q/Q^* clarifies the overall performance of our framework and, when clear from the context, we will refer to both Q and Q/Q^* as data quality.

The performance in terms of execution time has been extensively measured in our previous work in [79] and not reported here for conciseness. All experiments have been conducted on a machine equipped with an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz CPU and 32GB of RAM.

Table 6.2: Datasets description

Trait	Inmates	IBM HR Analytics	Red Wine Quality
Average of columns' entropy	5.35	3.13	5.61
Variance of columns' entropy	13.09	8.56	2.01
Row count	10000	1470	1599
Column count	12	35	12
Integer column count	2	12	2
Float column count	0	0	9
Bool column count	0	3	0
Timestamp column count	2	0	0
String column count	4	0	0
Categorical column count	4	20	1

6.2.1 Datasets

We conducted experiments across multiple datasets selected to maximize their diversity in terms of entropy measures, row and column counts, and distribution of data types. Table 6.2 presents the key characteristics of our datasets described in the following.

Inmates. It contains nightly updates of individuals incarcerated in Department of Corrections facilities. Data collection began on August 22, 2017, with daily updates reflecting the current inmate population.

IBM HR Analytics Employee Attrition & Performance. A popular fictional dataset created by IBM data analysts containing information on workers and work-life balance.

Red Wine Quality. Publicly available on Kaggle, it contains chemical and

sensory information from over 1500 red wines, designed for classification and regression modeling. The dataset features class imbalance with predominantly normal wines rather than excellent or poor ones.

The following sections present the experimental evaluations executed on the above datasets.

6.2.2 Experimental Evaluation 1: Metrics and Datasets

Experimental evaluation 1 assesses data quality variations with different combinations of metrics and datasets. The simulation is configured with seven nodes, five services, *RowLowerBound*=0.2, *RowUpperBound*=1, *ColumnLowerBound*=0.6, and *ColumnUpperBound*=0.9.

Figure 6.3, Figure 6.4 and Figure 6.5 present the results varying the target metric. Each row contains two boxplots. In the first boxplot, the orange mark inside the box represents the median value, while the blue line connects the average values. The second boxplot represents the difference in data quality between window $|w|$ and window $|w|-1$ within the same simulation. It highlights how increasing the window size improves the data quality stability. Additionally, all blue lines in Figure 6.2a, Figure 6.2b, and Figure 6.2c are aggregated in Figure 6.2 grouped by the metric type.

Figure 6.2 shows that the datasets and their distributions strongly affect the data quality. However, the overall trend remains consistent across all pairs of datasets and metrics. The results show that the window size significantly impacts all metrics: when the window size increases, the data quality increases, with a steeper trend for smaller window sizes. A side effect occurs in the qualitative metric for dataset Inmates, where the data quality at window size 6 is lower than at window size 5. We discuss this side effect in Section 6.2.5.

To validate whether our findings apply to longer pipelines, we conducted a single experiment with nine nodes (Figure 6.6). Due to the computational complexity of simulating longer pipelines with numerous service compositions, which is modeled by the expression $\sum_{s=MinS}^{MaxS} \sum_{n=MinN}^{MaxN} \sum_{|w|=1}^{min(n,Max|w|)} s^{|w|} * (n - |w| + 1) + n$, we limited our validation to one experiment only using dataset Inmates and the qualitative metric.

The above expression shows that increasing the number of services or nodes causes an exponential growth in the number of tool operations. The experiment was configured with five services, nine nodes, *RowLowerBound*=0.2, *RowUpperBound*=1, *ColumnLowerBound*=0.6, and *ColumnUpperBound*=0.9. Figure 6.6 confirms our previous observation: when the window size increases, the data quality increases, unless a possible side effect discussed in Section 6.2.5.

Across all experiments, once half of the window size is reached, the increase in the data quality flattens (blue line in Figure 6.6). We therefore claim that selecting a window size equal to half of the window size achieves a good balance between execution time and data quality.

6.2.3 Experimental Evaluation 2: Fixed Row-Column Filtering

Experimental evaluation 2 assesses data quality variations with different combinations of row and column filtering. The simulation is configured with seven nodes and five services. We set *RowLowerBound* and *RowUpperBound* to value V_{row} and *ColumnLowerBound* and *ColumnUpperBound* to V_{column}. We tested different combinations of V_{row} and V_{column}, with both parameters taking values in $\{0.2, 0.5, 0.8\}$. Figure 6.7a shows the combinations of V_{row} and V_{column}, each labeled with a letter from A to I. This choice reduces randomness in service implementation and evaluates how V_{row} and V_{column} affect data quality. We randomly selected the categories used to partition the column domains and the categories to remove during filtering.

We repeated the experiment for every dataset and results are presented in Figure 6.7b, Figure 6.7c, and 6.7d. Each line represents the data quality with a specific configuration of V_{row} and V_{column}. Our results show that configurations with high V_{row} (≥ 0.5), that is, a low percentage of filtered categories, produce high data quality. This effect is amplified when V_{column} is high (≥ 0.5) and reaches its highest impact on the data quality when both V_{row} and V_{column} are at the maximum value 0.8. In summary, strategy row filtering has a significantly greater impact on the data quality compared to column filtering.

6.2.4 Experimental Evaluation 3: Number of Candidate Services

Experimental evaluation 3 assesses how data quality varies with the number of candidate services on the dataset Inmates using the qualitative metric. The simulation is configured with six nodes, a number S of candidate services per node ranging from 2 to 10, *RowLowerBound*=0.2, *RowUpperBound*=1, *ColumnLowerBound*=0.6, and *ColumnUpperBound*=0.9.

Figure 6.8 shows that simulations with more candidate services per node yield higher data quality than those with fewer candidate services. This occurs because larger service pools increase the chance of finding near-optimal pipelines, whereas a small service pool significantly limits the number of available service compositions. Besides, the rate of data quality improvement decreases as the service pool size increases. With a small service pool, the chance of finding a better service composition by introducing an extra service is high because few service compositions exist. On the other hand, as the service pool increases in dimension, the marginal benefit of each additional service diminishes. This is due to a *saturation effect*: once a set of near-optimal compositions is available, the probability that a new service yields a better composition is low.

6.2.5 Side Effect of the Sliding Window Algorithm

During the experimental evaluation, we observed a side effect in specific simulation instances: the data quality retrieved at window size $|w|$ was higher than the one at window size $|w|+1$, showing a decrease in data quality with higher window size. This observation contradicts the theoretical expectation that data quality should demonstrate nondecreasing monotonic behavior as the window size increases, given that larger windows provide a broader overview of possible service compositions. This phenomenon is due to the operational characteristics of our heuristic implementation, as discussed below.

To illustrate this side effect, we provide a simple example based on the pipeline template shown in Table 6.3a. The pipeline includes three vertices and four services: the first two services are candidates for the initial vertex, while the remaining services are each associated with one of the remaining

Table 6.3: Side effect on data quality.

Vertex	Services
v_1	s_{11}, s_{12}
v_2	s_{21}
v_3	s_{31}

(a) Pipeline Template

Service Chain	w	Q
s_{11}	w=1	0.8
s_{12}	w=1	0.7
$s_{11} \rightarrow s_{21}$	w=2	0.5
$s_{12} \rightarrow s_{21}$	w=2	0.6
$s_{11} \rightarrow s_{21} \rightarrow s_{31}$	w=3	0.4
$s_{12} \rightarrow s_{21} \rightarrow s_{31}$	w=3	0.3

(b) Service compositions

vertices. Table 6.3b lists the service compositions and corresponding data quality computed on the corresponding composition varying the window size w. In the following, we denote as $Q(s_{1,i} \rightarrow ... \rightarrow s_{|V|,j})$ the quality of the service composition $s_{1,i} \rightarrow ... \rightarrow s_{|V|,j}$.

With window size $|w|$=1, the algorithm retrieves $Q(s_{11})$=0.8 and $Q(s_{12})$=0.7. The service selected for the first node of the pipeline is s_{11} because $Q(s_{11})>Q(s_{12})$. At the end of the algorithm, the resulting data quality is $Q(s_{11} \rightarrow s_{21} \rightarrow s_{31})$=0.4.

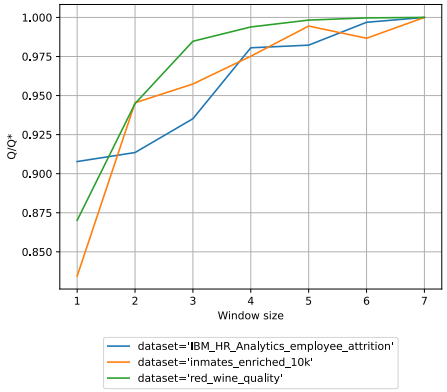
With window size $|w|$=2, the algorithm retrieves $Q(s_{11} \rightarrow s_{21})$=0.5 and $Q(s_{12} \rightarrow s_{21})$=0.6. The service selected for the first node of the pipeline is s_{12} because $Q(s_{11} \rightarrow s_{21})<Q(s_{12} \rightarrow s_{21})$. At the end of the algorithm, the resulting data quality is $Q(s_{12} \rightarrow s_{21} \rightarrow s_{31})$=0.3, which is lower than the metric value computed with $|w|$=1, leading to a decline in data quality.

With window size $|w|$=3, the algorithm evaluates $Q(s_{11} \rightarrow s_{21} \rightarrow s_{31})$=0.4 and $Q(s_{12} \rightarrow s_{21} \rightarrow s_{31})$=0.3. The service selected for the first node of the pipeline is $s11$ because $Q(s_{11} \rightarrow s_{21} \rightarrow s_{31})>Q(s_{12} \rightarrow s_{21} \rightarrow s_{31})$. At the end of the algorithm, the resulting data quality is $Q(s_{11} \rightarrow s_{21} \rightarrow s31)$=0.4. We note that when the window size equals the pipeline length, the algorithm always finds the optimal pipeline instance.

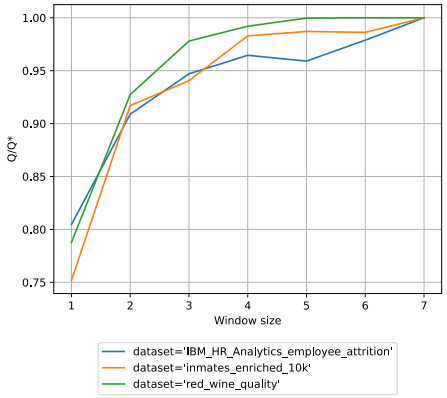
From the above example, we note that the data quality retrieved with window size 2 is lower than that retrieved with window size 1. This fluctuation is due to a dependency between s_{12} and s_{21}, which results in the instantiation of service s_{12} with window size 2. This choice affects the whole pipeline instantiation resulting in a suboptimal choice with window size 2. It is important to note that the side effect in this section does not compromise the overall quality pattern when averaging across numerous simulations.

6.3 Chapter Summary

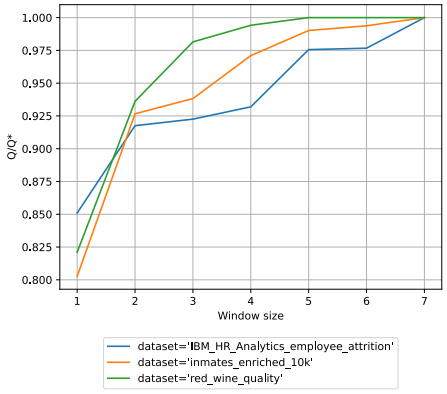
This chapter presented the practical implementation and experimental evaluation of the data governance framework through a configurable tool that validates the sliding window optimization approach. The tool enables reproducible benchmarking of pipeline instances across diverse scenarios while supporting multiple quality metrics and data protection configurations. Experimental results demonstrate that the sliding window heuristic achieves near-optimal data quality performance across different datasets and filtering strategies, confirming the framework's feasibility in balancing computational efficiency with analytical utility in real-world service-based data processing environments.



(a) Qualitative metric



(b) Quantitative metric



(c) Entropy-based metric

Figure 6.2: Aggregated results for metric types using normalized data quality Q/Q^*.

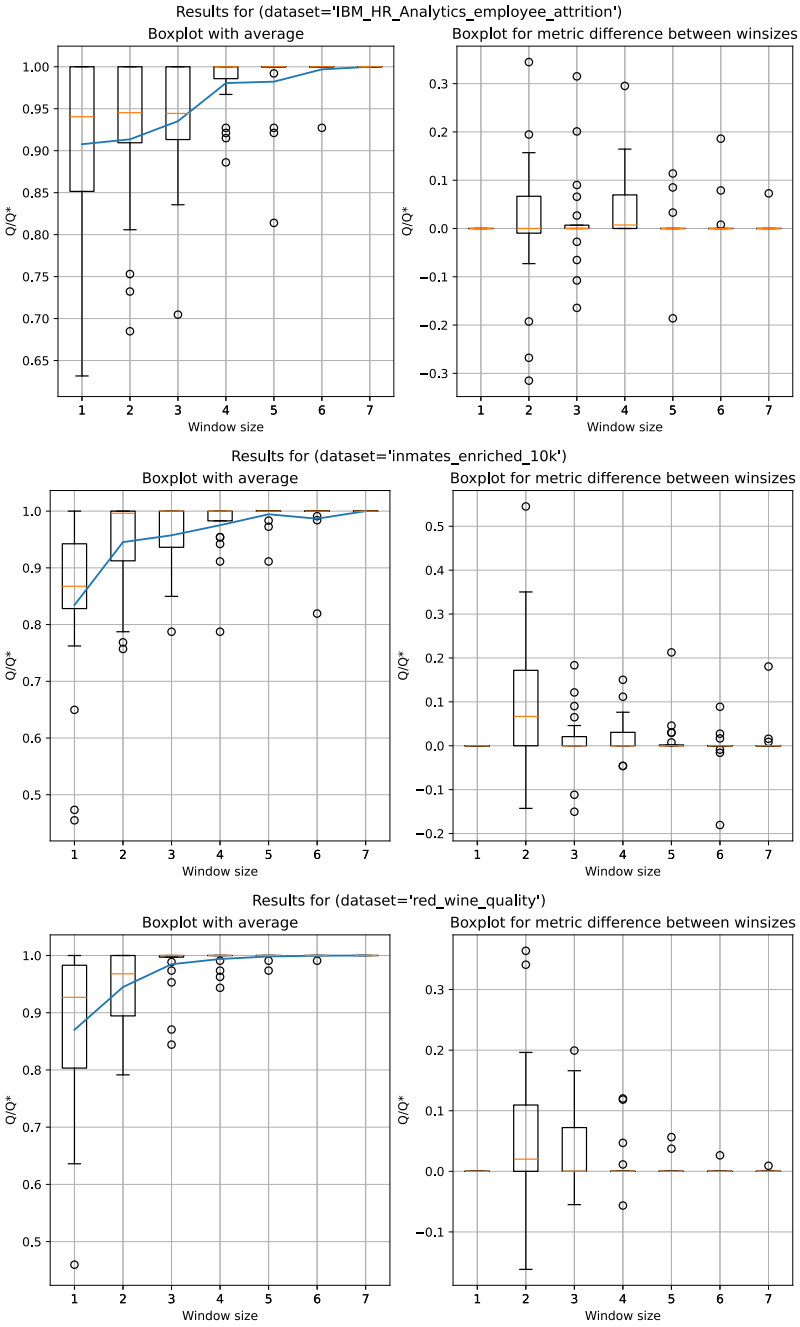


Figure 6.3: Experimental results for qualitative metric using normalized data quality Q/Q^*.

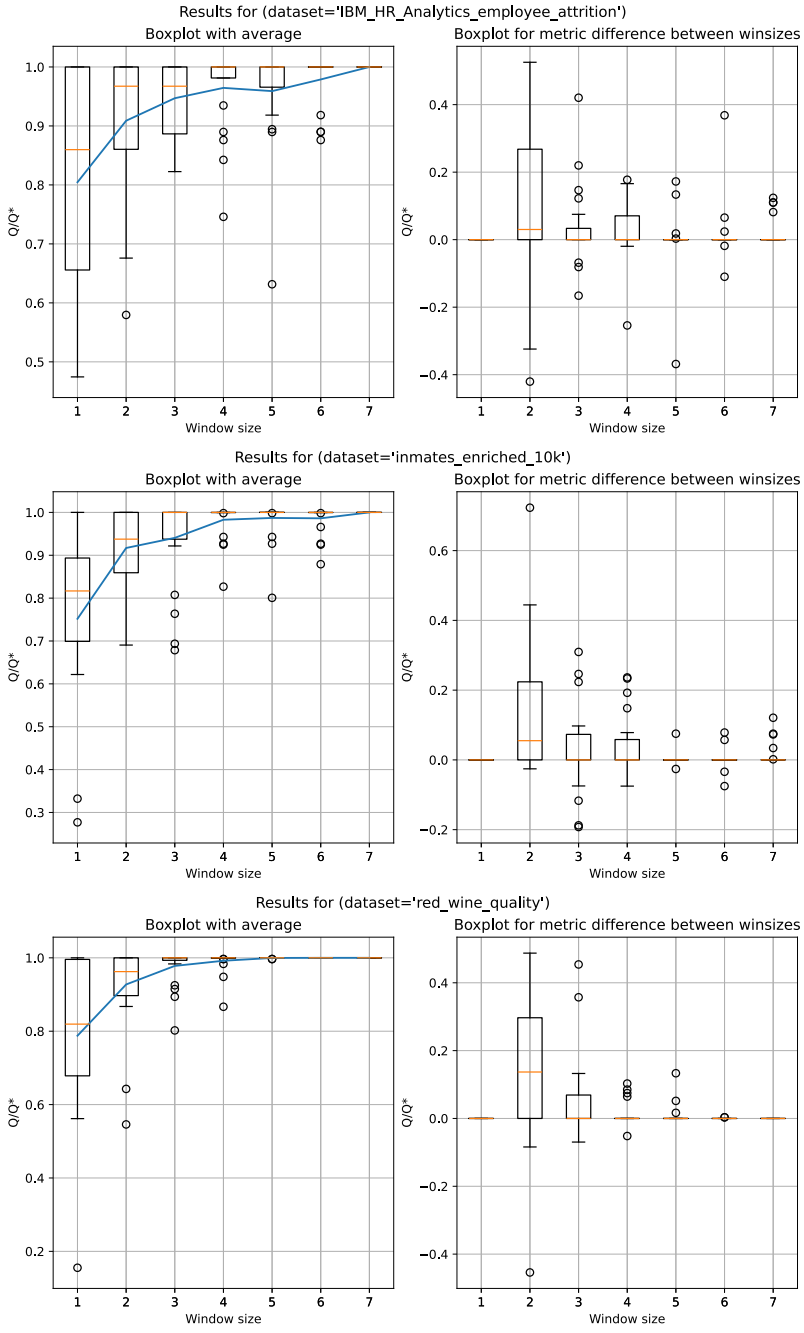


Figure 6.4: Experimental results for quantitative metric using normalized data quality Q/Q^*.

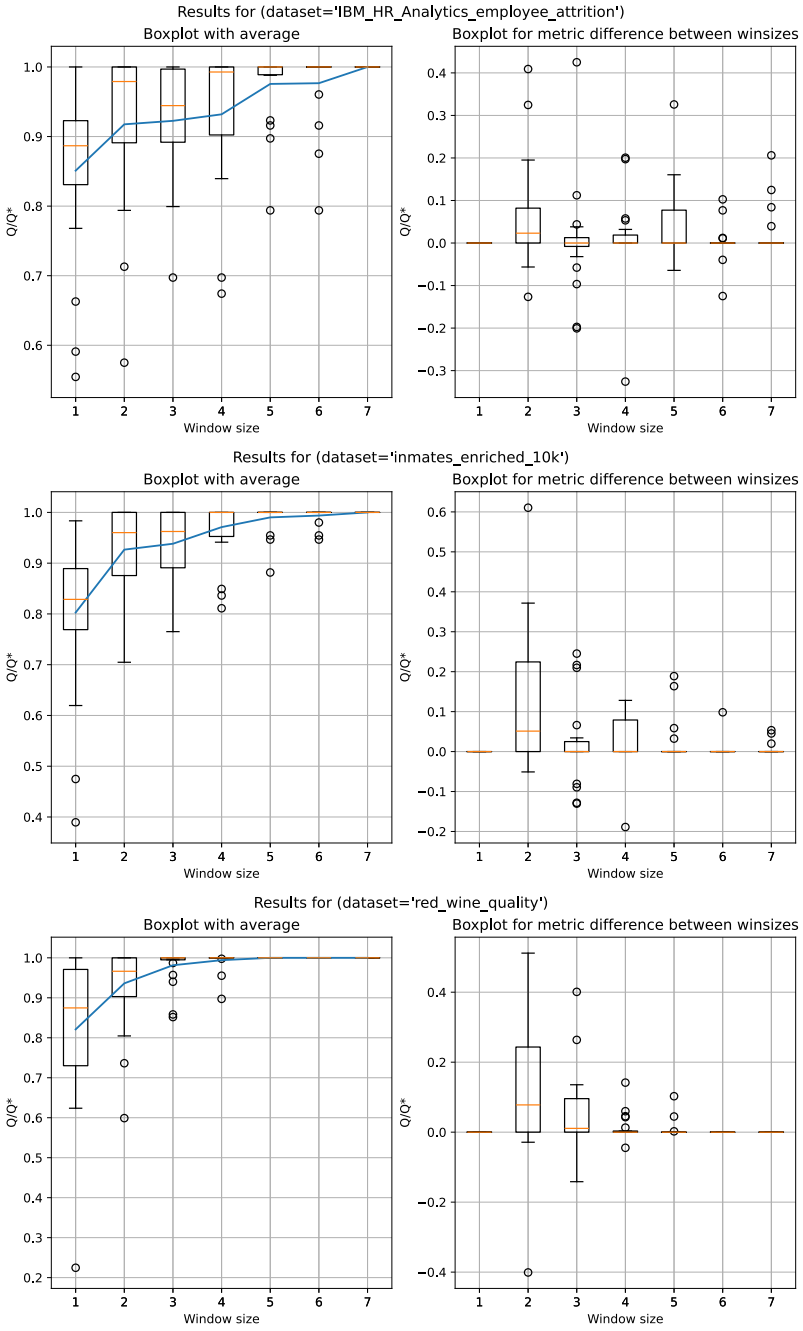


Figure 6.5: Experimental results for entropy-ratio metric using normalized data quality Q/Q^*.

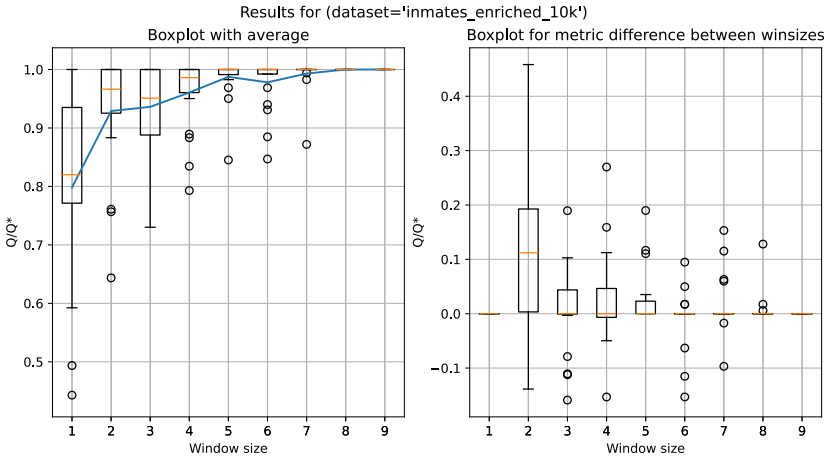
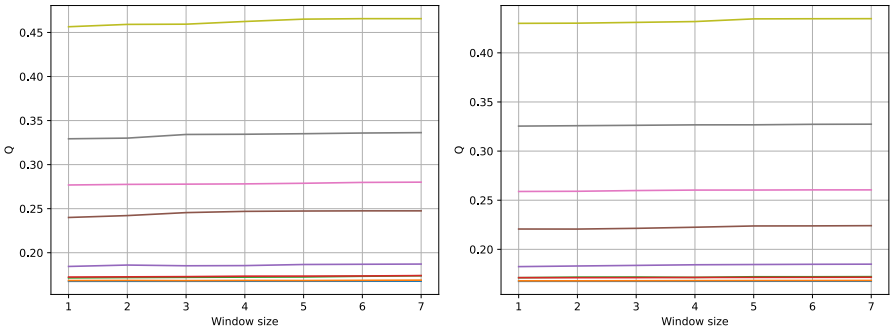


Figure 6.6: Experimental results with window size=9 using normalized data quality Q/Q^*.

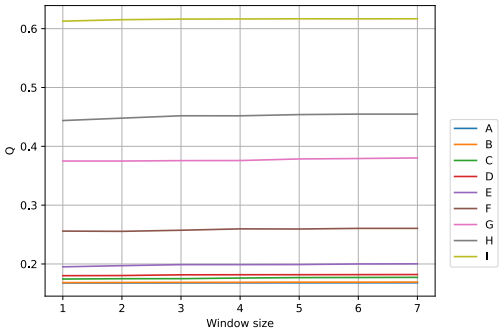
Letter	V_{row}	V_{column}
A	0.2	0.2
B	0.2	0.5
C	0.2	0.8
D	0.5	0.2
E	0.5	0.5
F	0.5	0.8
G	0.8	0.2
H	0.8	0.5
I	0.8	0.8

(a) Mapping of V_{row} and V_{column} configuration to letter A-I



(b) Inmates dataset

(c) IBM dataset



(d) Red-Wine dataset

Figure 6.7: Experimental results varying the dataset, and row and column filtering

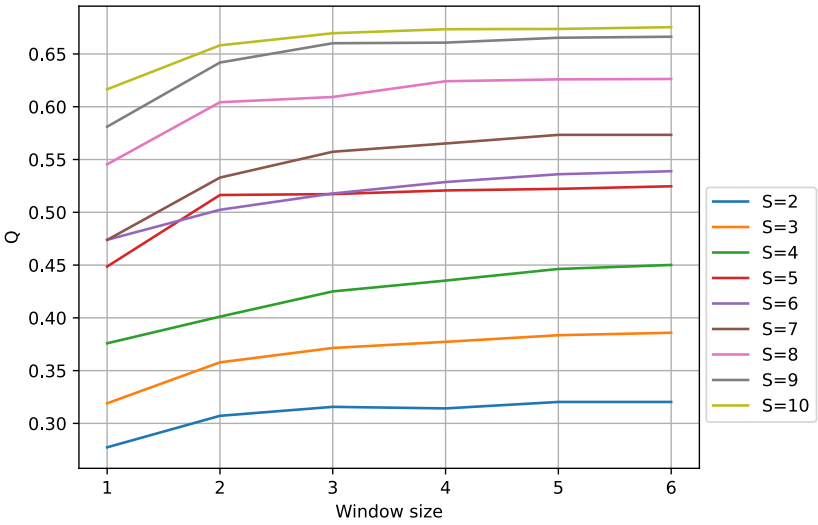


Figure 6.8: Experimental results varying the number S of candidate services in [2,10]

1186. Data Governance Framework: Implementation Choices and Experimental Evaluation

7

Industrial Integration

7.1 Introduction

This chapter reports the industrial validation of the proposed methodology using ALIDA (Advanced Laboratory for Interactive Data Analytics), a microservice-based platform developed by Engineering Ingegneria Informatica for composing, deploying, and executing Big Data Analytics workflows. The goal is to assess the feasibility of maximizing data quality while upholding data privacy in a real orchestration environment. The validation approach leverages a dedicated evaluation service that executes parametrised anonymization variants with controlled quality and protection behaviours. Data quality is monitored by comparing input and output datasets to trace transformations throughout the pipeline. An automation script interacts with the ALIDA API to systematically explore different configurations, launch pipelines, and collect results. This systematic exploration identifies trade-offs between utility and protection, deriving configurations that satisfy privacy constraints while preserving quality.

The chapter is organized as follows: Section 7.2 introduces the ALIDA platform, describing its user interface components and underlying architecture. Section 7.3 presents the extended process that incorporates the proposed optimization methodology within ALIDA's workflow execution. The integration details, including the optimization algorithm Finally, **??** presents the experimental results and validation outcomes.

7.2 ALIDA

ALIDA provides a model-based BDA-as-a-Service environment in which services are registered via APIs or graphical interfaces and assembled into batch or streaming pipelines using a visual designer. The standard workflow involves service registration in a shared catalogue, pipeline composition through a visual interface, configuration of execution parameters, deployment and monitoring of results through integrated dashboards. Services can be parameterized and reused across different pipelines, enabling modular workflow construction. The ALIDA platform organizes resources

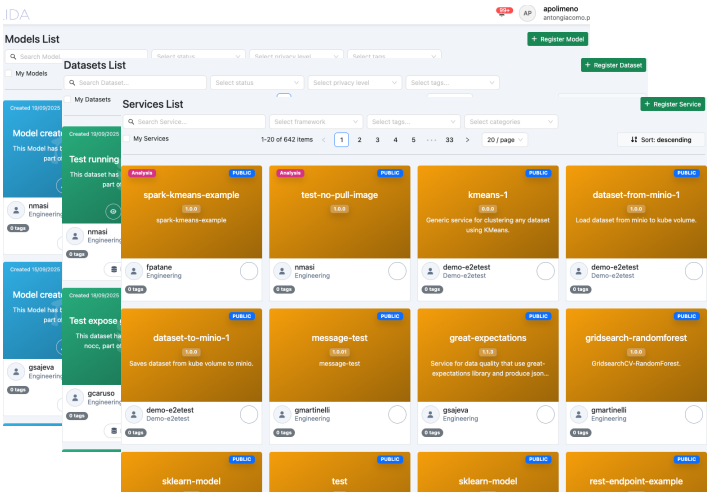


Figure 7.1: ALIDA resource management interfaces showing examples of dedicated list pages for models, datasets, and services. Each resource type has its own management interface that provides creation, browsing, and configuration capabilities for platform components.

through dedicated management interfaces for each resource type: data sources, datasets, models, services, and Big Data Application (BDA) workflows. Each resource category has its own list page that provides management capabilities including creation, browsing, filtering, and configuration of components. Figure 7.1 illustrates examples of these resource management

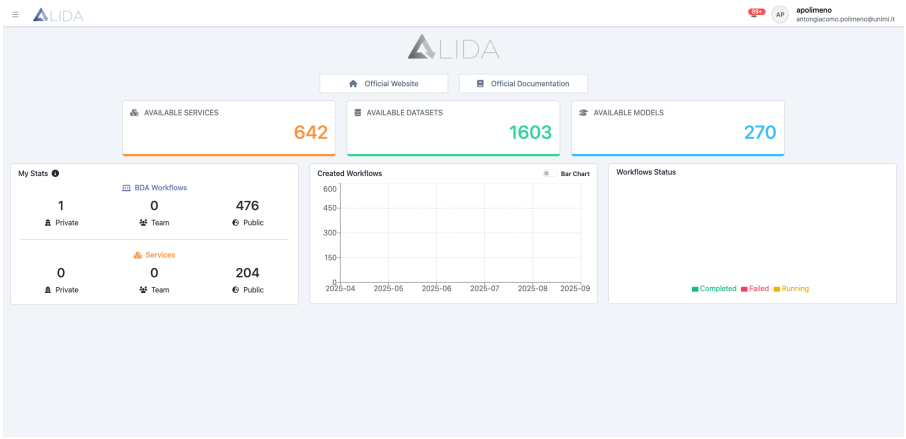


Figure 7.2: ALIDA main dashboard showing system statistics, workflow status, and platform utilization metrics.

interfaces, showing the models list, datasets list, and services list pages that enable users to organize and access platform components systematically.

These dedicated list pages enable resource lifecycle management: users can create new resources directly through form-based interfaces, browse existing items with filtering and search capabilities, access detailed metadata and version information, and configure component parameters for pipeline integration. The separation of resource types ensures organized management while maintaining clear dependencies and relationships between different platform components.

The main dashboard, illustrated in Figure 7.2, presents system status and statistics for Big Data Analytics workflows and items within the ALIDA platform. It provides an overview of workflow execution status, system resource utilization, and performance metrics.

The BDA list page, shown in Figure 7.3, provides a view of public and personal Big Data Analytics workflows within the platform. Users can browse, filter, and search through available workflows, accessing summary information including execution status, creation date, and performance metrics. The interface is a navigation hub for workflow management and enables users to locate and access analytical processes.

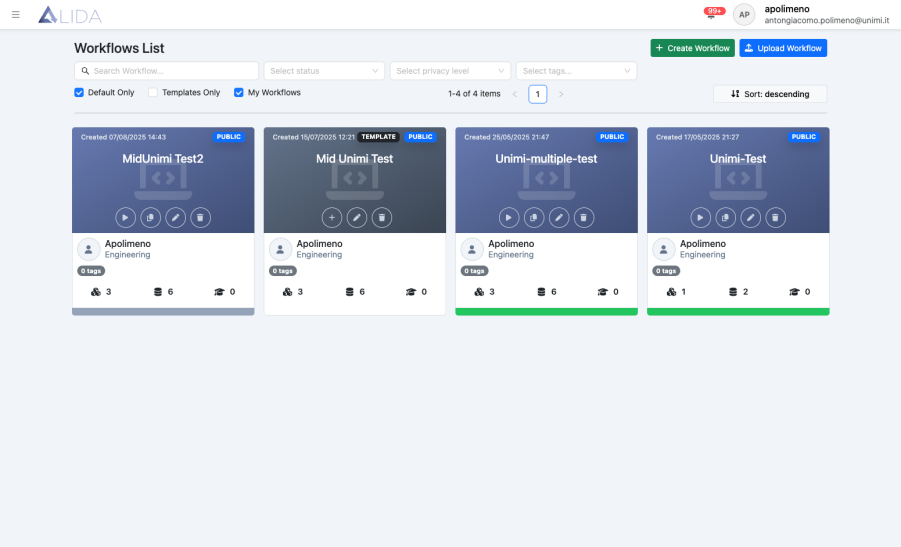


Figure 7.3: ALIDA BDA list interface displaying available Big Data Analytics workflows with filtering and search capabilities. Users can browse and access workflow summaries and execution status.

Individual BDA workflow details and their outputs are presented through dedicated detail pages, as shown in Figures 7.4 , 7.5, 7.6. These interfaces provide, among others, workflow management capabilities: users can modify workflows by adding or removing services, datasets, and models, and connecting them to create custom processing pipelines. Moreover users can start, clone or delete obsolete workflows and launch new instances. Beyond execution control, users can access experimental configurations, review execution history, and analyze past runs.

From an infrastructural point of view, ALIDA is a microservice-based platform [80]. Core components include Argo Workflow for orchestration, Kubernetes for deployment, Apache Kafka for messaging, and Minio for object storage. GPU support and MLOps features extend the platform to machine learning workflows, enabling model serving with lifecycle management (versioning, automated deployment). The architecture, summarized in

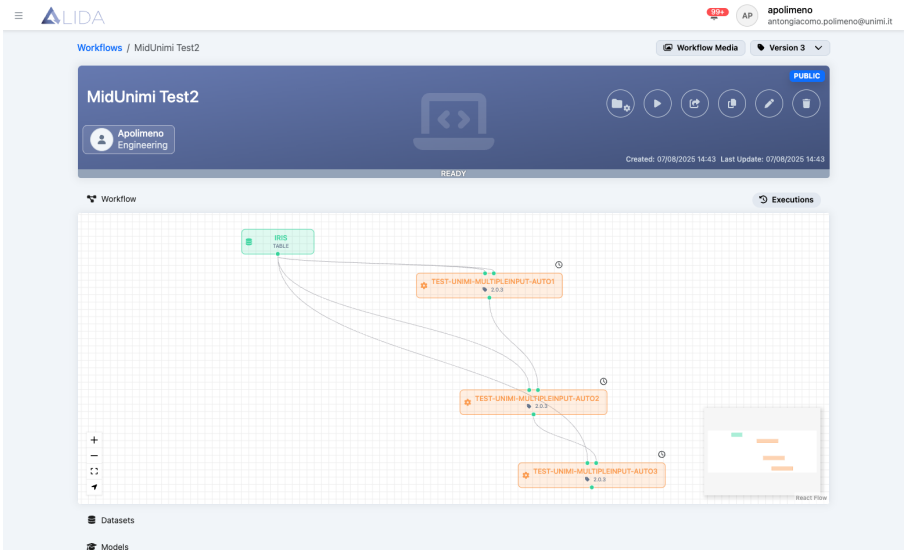


Figure 7.4: ALIDA BDA detail interface showing workflow information including configuration parameters, execution status, and processing details for individual analytics workflows.

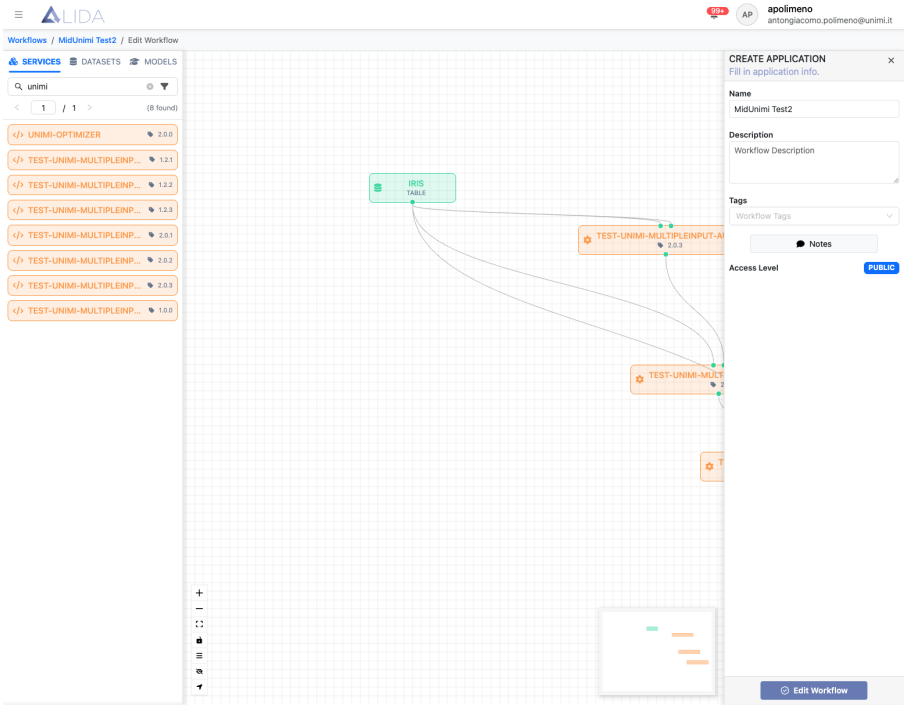


Figure 7.5: ALIDA BDA detail interface in edit mode, allowing users to modify workflow configurations, add or remove services, datasets, and models, and connect components to create custom processing pipelines.

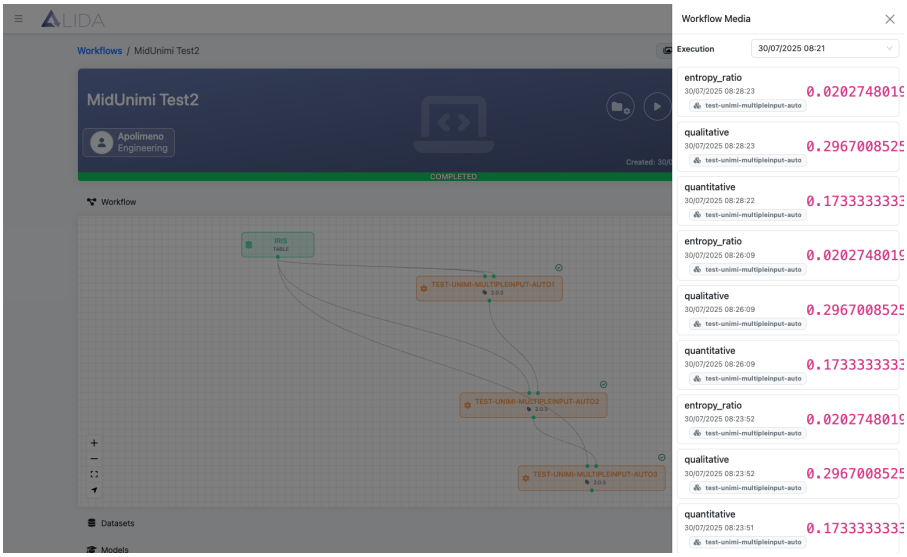


Figure 7.6: ALIDA BDA metrics interface displaying detailed output metrics and results from workflow execution, providing analytical insights and performance measurements for completed analytics processes.

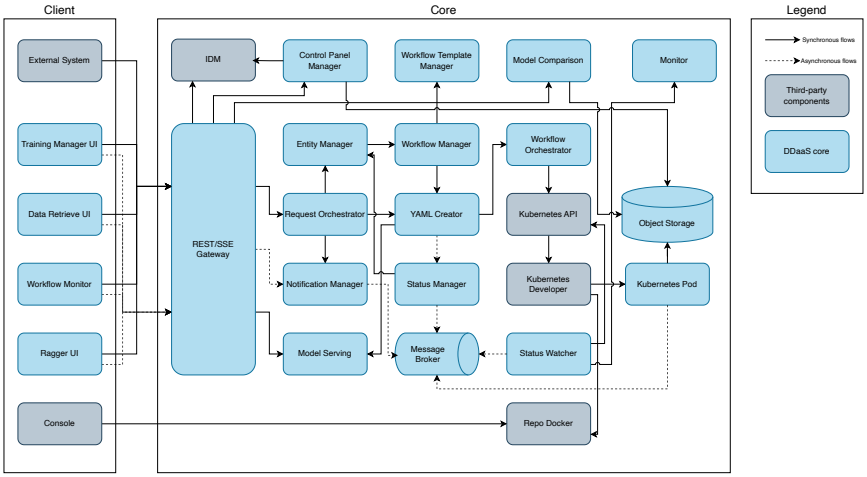


Figure 7.7: High-level architecture of the ALIDA platform, illustrating the interaction between client components, core services, orchestration infrastructure, and storage systems.

Figure 7.7, is layered: clients access the system through a REST/SSE gateway; identity services enforce authentication and authorization; managers and orchestrators translate templates and YAML specifications into Kubernetes instructions; the cluster executes Pods using container images and object storage for artifacts; status and monitoring services track execution and provide feedback; a message broker distributes asynchronous events; and results are returned through the gateway. This arrangement maintains a continuous flow from request to execution with orchestration and persistent storage.

7.3 Methodology Integration and Optimization Framework

Building upon ALIDA's workflow capabilities, this section presents the integration of the proposed data quality and privacy optimization methodology within the platform's execution environment. While ALIDA provides work-

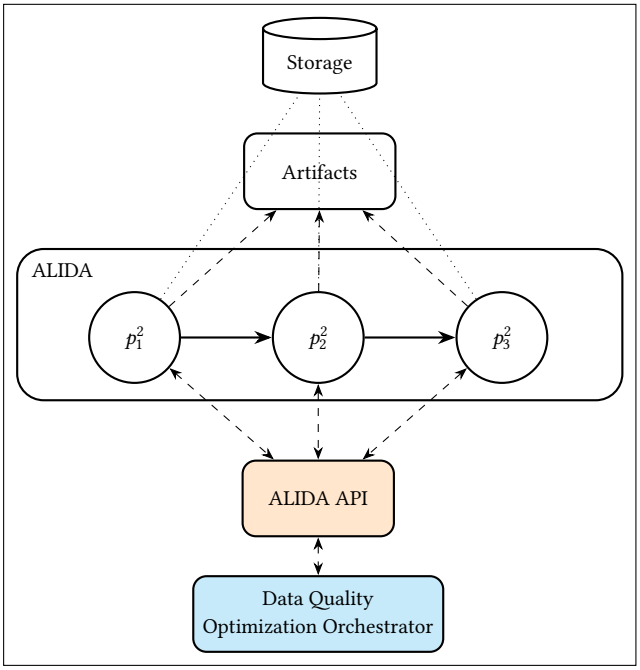


Figure 7.8: Integration of the proposed data quality and protection optimization framework within the ALIDA platform architecture. The framework operates as an external service that interacts with ALIDA's orchestration layer via its API, enabling remote control of pipeline execution and evaluation.

flow management and execution infrastructure, the manual configuration of privacy-preserving data processing pipelines remains challenging, particularly when balancing competing objectives of data utility and protection.

To address this limitation, an automated optimization orchestrator has been developed that extends ALIDA's functionality by systematically exploring configuration spaces to identify optimal trade-offs between data quality and privacy constraints. The extended architecture, illustrated in Figure 7.8, operates through ALIDA's API, enabling programmatic parameterization and workflow control of execution while leveraging the platform's native scaling, monitoring and metrics collection capabilities.

Algorithm 4: Multi-Objective Fraction Optimization for Service Groups

INPUT: List of services S; List of sample fractions F; Objective: minimize qualitative, quantitative, entropy metrics.
OUTPUT: Best service grouping and fraction, for all objectives.

Load DATASET;
Define experiment, logging, and optimization parameters;
Set up MLflow tracking and experiment;
Set optimization directions: [min, min, min] (for qualitative, quantitative, entropy);
foreach *service grouping g of* $|S|$ *services* **do**
    foreach *fraction tuple f in* $F^{|S|}$ **do**
        Add (g, f) to candidate set;
    end
end
foreach *trial* **do**
    Select (g, f) from candidate set;
    Build parameter set for BDA using (g, f);
    Evaluate BDA run with current params → metric matrix M;
    if *no valid output* **then**
        Assign infinite objectives (trial fails);
    end
    else
        $Q \leftarrow$ sum of qualitative values in M;
        $N \leftarrow$ sum of quantitative values in M;
        $E \leftarrow$ sum of entropy values in M;
        Return (Q, N, E) as objective triple;
    end
end
Run multi-objective optimization (e.g., Optuna) using objective function;
foreach *optimal trial found* **do**
    Log service parameters and metrics to experiment tracker;
    Save or export summary of results;
end
return *Set of best configurations for all objectives*

A dedicated data quality assessment service performs comparative analysis between input and output datasets, calculating differentials in quality metrics to measure preservation or degradation during processing. This service requires an additional input parameter containing the original dataset to ensure continuous quality monitoring throughout data transformation and store the resulting metrics leveraging ALIDA's native metric output system.

The pseudocode outlined in algorithm **??** formalizes the logic of experimental optimization algorithm. The process begins by initializing global experiment parameters and service lists (lines 1-4), linking the experiment to a tracking platform to store and track quality metrics. The algorithm generate all possible combinations of sampling fractions assigned to each service group (lines 5-9). This exhaustive product of possible assignments forms the candidate configuration set, ensuring that the optimization is not limited to local minima and can survey the entire feasible search space compliant with policy constraints inherent to the pipeline. For each candidate configuration, the algorithm constructs parameters that are passed to each service, encapsulating the service label and the assigned sample fraction (lines 11-12). The configuration is then evaluated through a BDA run that produces a metric matrix (line 13). Each experiment results in a matrix containing the computed values for three metrics: qualitative score, quantitative score, and entropy for each group. These metrics collectively capture distinct facets of both data utility and information-theoretic unpredictability within the pipeline. The algorithm proceeds by summing across the groups for each metric (lines 17-19), producing aggregated values for the three objectives: the overall qualitative score, the overall quantitative score, and the overall entropy. It then leverages a multi-objective optimization framework (line 23), treating each metric as an independent objective to be minimized, thereby jointly balancing competing system goals. If a candidate configuration fails to produce results due to policy violations or data insufficiency, it is assigned infinite objective values (line 15), effectively pruning unviable solutions from the search process.

Through iterative trials, the optimizer actively explores the configuration space, guided by the outcomes of previous evaluations. Upon completion, the algorithm returns the set of best-performing configurations-those which achieve the most favorable trade-offs among the three objectives.

Rather than employing the simpler sliding window heuristics, the selection of optimal configurations utilizes an advanced Tree-structured Parzen Estimator (TPE) algorithm. TPE is a hyperparameter optimization algorithm that operates within the Bayesian optimization framework by iteratively selecting hyperparameter configurations to minimize an unknown objective function. The algorithm employs Parzen window estimators, which are kernel density estimators that provide smoothed approximations of probability distributions, to model the density of hyperparameter configurations based on their performance. TPE partitions previously evaluated hyperparameter configurations into two groups using a threshold value y^* that separates good and bad performance outcomes, then constructs density estimates $l(x) = p(x|y < y^*)$ for configurations with superior performance and $g(x) = p(x|y > y^*)$ for configurations with inferior performance. The algorithm selects the next hyperparameter configuration by maximizing the density ratio $l(x)/g(x)$, effectively choosing configurations that are most likely to yield good results while being least likely to produce poor outcomes. The traditional independent TPE implementation assumes parameter independence and estimates multivariate densities as products of univariate Parzen window estimators, which fails to capture interdependencies between hyperparameters. The multivariate TPE extension addresses this limitation by directly modeling the joint densities $l(x)$ and $g(x)$ using multivariate Parzen window estimators, thereby capturing parameter dependencies and improving optimization performance through more expressive density modeling, particularly in scenarios where optimal parameter values exhibit correlation or conditional dependencies.

7.4 Framework Integration with ALIDA

The integration of ALIDA was implemented through a systematic approach designed to monitor and optimize data quality throughout the privacy-preserving data processing pipeline. A dedicated service was developed to leverage ALIDA's native metrics output system for data quality assessment.

This service performs comparative analysis between input and output datasets by calculating the differential in data quality metrics, enabling precise measurement of quality preservation or degradation during processing.

To facilitate this comparative analysis, the service requires an additional artificial input parameter containing the original dataset, ensuring continuous quality monitoring capabilities throughout the data transformation process. The service architecture supports configurable operational parameters, including encryption policies and data protection mechanisms, providing flexibility for various privacy requirements.

The optimization process is managed through a dedicated configurator component that systematically evaluates different service configurations to identify the optimal setup for maximizing data quality. This configurator operates through programmatic manipulation of ALIDA via its application programming interface, executing a two-phase optimization workflow.

The configuration phase utilizes a base pipeline template that serves as the foundation for the research methodology, incorporating all necessary service types required for the privacy-preserving data processing workflow. Following template establishment, the optimization script generates multiple configuration instances from this base template and manages their parallel execution, monitoring completion status and retrieving results for comparative evaluation.

Prior to pipeline execution, the script computes all possible service combinations within the defined parameter space, creating a search domain for configuration optimization. Rather than employing basic sliding window heuristics, the selection of optimal configurations utilizes an advanced tree-parser algorithm, which provides more sophisticated optimization capabilities for navigating the complex configuration space and identifying solutions that maximize data utility while maintaining privacy constraints.

7.5 Multi-Objective Optimization Algorithm

The pseudocode outlined in Line 5 formalizes the logic underpinning our experimental optimization algorithm within the data quality and protection framework. The process begins by initializing global experiment parameters and service lists (lines 1-4), linking the experiment to a tracking platform to ensure full reproducibility.

The core of the algorithm is the systematic generation of all possible combinations of sampling fractions assigned to each service group (lines 5-9).

Algorithm 5: Multi-Objective Fraction Optimization for Service Groups

INPUT: List of services S; List of sample fractions F; Objective: minimize qualitative, quantitative, entropy metrics.
OUTPUT: Best service grouping and fraction, for all objectives.

```
Load DATASET;
Define experiment, logging, and optimization parameters;
Set up MLflow tracking and experiment;
Set optimization directions: [min, min, min] (for qualitative, quantitative, entropy);
foreach service grouping g of |S| services do
    foreach fraction tuple f in F^|S| do
        Add (g, f) to candidate set;
    end
end
foreach trial do
    Select (g, f) from candidate set;
    Build parameter set for BDA using (g, f);
    Evaluate BDA run with current params → metric matrix M;
    if no valid output then
        Assign infinite objectives (trial fails);
    end
    else
        Q ← sum of qualitative values in M;
        N ← sum of quantitative values in M;
        E ← sum of entropy values in M;
        Return (Q, N, E) as objective triple;
    end
end
Run multi-objective optimization (e.g., Optuna) using objective function;
foreach optimal trial found do
    Log service parameters and metrics to experiment tracker;
    Save or export summary of results;
end
return Set of best configurations for all objectives
```

This exhaustive product of possible assignments forms the candidate configuration set, ensuring that the optimization is not limited to local minima and can survey the entire feasible search space compliant with policy constraints inherent to the pipeline.

For each candidate configuration, the algorithm constructs parameters that are passed to the batch data analysis application (BDA), encapsulating the service label and the assigned sample fraction (lines 11-12). The configuration is then evaluated through a BDA run that produces a metric matrix (line 13).

Each experiment results in a matrix containing the computed values for three metrics: qualitative score, quantitative score, and entropy for each group. These metrics collectively capture distinct facets of both data utility and information-theoretic unpredictability within the pipeline. The algorithm proceeds by summing across the groups for each metric (lines 17-19), producing aggregated values for the three objectives: the overall qualitative score, the overall quantitative score, and the overall entropy.

It then leverages a multi-objective optimization framework (line 23), treating each metric as an independent objective to be minimized, thereby jointly balancing competing system goals. If a candidate configuration fails to produce results due to policy violations or data insufficiency, it is assigned infinite objective values (line 15), effectively pruning unviable solutions from the search process.

Through iterative trials, the optimizer actively explores the configuration space, guided by the outcomes of previous evaluations. Upon completion, the algorithm returns the set of best-performing configurations—those which achieve the most favorable trade-offs among the three objectives—enabling further analysis, benchmarking, or deployment.

Tree-structured Parzen Estimator Implementation
TPE is a hyperparameter optimization algorithm that operates within the Bayesian optimization framework by iteratively selecting hyperparameter configurations to minimize an unknown objective function. The algorithm employs Parzen window estimators, which are kernel density estimators that provide smoothed approximations of probability distributions, to model the density of hyperparameter configurations based on their performance.
TPE partitions previously evaluated hyperparameter configurations into two groups using a threshold value y^* that separates good and bad performance outcomes, then constructs density estimates $l(x) = p(x|y < y^*)$ for configurations with superior performance and $g(x) = p(x|y > y^*)$ for configurations with inferior performance. The algorithm selects the next hyperparameter configuration by maximizing the density ratio $l(x)/g(x)$, effectively choosing configurations that are most likely to yield good results while being least likely to produce poor outcomes.
The traditional independent TPE implementation assumes parameter independence and estimates multivariate densities as products of univariate Parzen window estimators, which fails to capture interdependencies between hyperparameters. The multivariate TPE extension addresses this limitation by directly modeling the joint densities $l(x)$ and $g(x)$ using multivariate Parzen window estimators, thereby capturing parameter dependencies and improving optimization performance through more expressive density modeling, particularly in scenarios where optimal parameter values exhibit correlation or conditional dependencies.

7.6 Experimental Evaluation

The experimental evaluation of the new multi-objective optimization heuristics compared to the sliding window approach is currently in progress. The evaluation requires the reimplementation of the sliding window heuristic to enable direct comparisons with the TPE-based optimization.

Implementation work has been completed for the TPE integration and

multi-objective optimization framework. The full experimental results will be available upon completion of the framework reimplementation and comprehensive evaluation across multiple industrial scenarios.

7.7 Chapter Summary

This chapter presented the industrial integration of the data governance framework through collaboration with industry partners and the development of advanced multi-objective optimization algorithms. The work extends the original sliding window heuristic with Bayesian optimization techniques that can simultaneously optimize multiple quality dimensions while ensuring data protection compliance. The Tree-structured Parzen Estimator implementation enables more sophisticated service selection by capturing parameter dependencies and improving optimization performance in complex industrial scenarios with interdependent service configurations.

8

Conclusions

In this thesis, we have investigated data governance frameworks in the context of modern distributed data processing systems, presenting a novel policy-driven approach that addresses the requirements and challenges of balancing data quality and privacy protection in service-based pipelines. After a brief introduction (Chapter 1) and related work analysis (Chapter 2), Chapters 3–5 presented a multi-dimensional, dynamic optimization framework at the core of our approach. The framework has been validated through experimental evaluation (Chapter 6) and integrated with industrial platforms (Chapter 7). In this chapter, we summarize the contributions of this thesis (Section 8.1) and outline challenges left to future work (Section 8.2).

8.1 Summary of the Contributions

The contribution of this thesis is the definition of a data governance framework for modern distributed data processing applications. The framework consists of a policy-driven service selection and composition approach that dynamically optimizes quality-privacy trade-offs, and is fully integrated within the data pipeline lifecycle, from template design and service discovery to instance optimization and execution monitoring.

Policy-Driven Service Selection Framework. We defined a policy-driven framework that enables systematic specification and enforcement of quality and privacy requirements in distributed data processing pipelines. Our approach extends traditional attribute-based access control (ABAC) models to incorporate data quality considerations alongside privacy protec-

tion requirements. We developed a service selection process that simultaneously optimizes multiple, often conflicting objectives including data quality preservation and privacy protection strength. The framework includes systematic approaches for managing the fundamental tension between data utility maximization and privacy protection, with metrics that quantify both quality degradation and privacy protection strength for objective evaluation of different service combinations.

Industrial Integration and Validation. We provided practical approaches for integration with existing data processing platforms and industrial systems. We integrated our framework with ALIDA, an industrial data analytics platform developed by Engineering Ingegneria Informatica, demonstrating practical deployment of quality-privacy optimization in real-world scenarios. We developed experimental evaluation approaches that assess the effectiveness of our quality-privacy optimization across multiple dimensions and provided analysis of the computational and operational overhead introduced by our governance framework, demonstrating that quality-privacy optimization can be achieved with acceptable performance characteristics for practical deployment in distributed data processing environments.

8.2 Future Work

The research described in this thesis can be extended along several directions.

Increasing automation while reducing complexity assumptions. Any data governance framework, including the one presented in this thesis, builds on assumptions about the complexity and predictability of service behaviors and quality-privacy relationships. Future work includes the definition of techniques that reduce these assumptions while preserving the effectiveness of quality-privacy optimization. This might require new approaches to automatically learn service characteristics and quality-privacy trade-offs from observed behavior, reducing the need for manual policy specification. Additionally, federated learning techniques might be implemented to enable collaborative governance across organizational boundaries without sharing sensitive policy information.

Governance of composite distributed pipelines. Governance of

complex distributed pipelines refers to the ability to optimize quality-privacy trade-offs across pipeline compositions spanning multiple organizations and platforms. This challenge extends beyond single-platform optimization to address inter-organizational data sharing agreements and federated governance policies. Work to be investigated includes: i) how to harmonize quality metrics and privacy requirements across different organizational contexts and regulatory jurisdictions; ii) how to compose optimization decisions in a multi-stakeholder setting where different participants may have conflicting objectives and constraints.

Governance-driven system lifecycle management. The need to maintain quality-privacy balance throughout system evolution has led to approaches for dynamic service adaptation built on continuous optimization. An interesting research direction consists of a fully governance-driven approach for distributed system management. This includes: i) novel predictive approaches that anticipate changes in data characteristics, service availability, or regulatory requirements that might affect quality-privacy optimization; ii) proactive adaptation strategies that prevent quality-privacy violations rather than reactive optimization that responds to observed changes.

Advanced machine learning governance. The integration of ML workloads represents an important step towards data governance in AI-driven systems. Several extensions remain to be explored. First, additional approaches for ML-specific quality assessment should be considered, including fairness evaluation techniques that can provide stronger guarantees comparable to formal verification methods, and interpretability analysis that enables inspection of ML model decision-making processes. Second, ML models are increasingly treated as reusable artifacts that are initially trained by one organization and then fine-tuned by others for specific applications. This scenario points to a marketplace of ML services whose cornerstone must be governance-aware deployment. It introduces the need for: i) standardized quality-privacy specifications that are both rigorous and understandable for diverse stakeholders; ii) governance processes that evaluate ML services across different deployment contexts and adaptation phases. Third, the relationship between distributed data processing and ML model composition should be considered, especially when ML is used both for data analytics and pipeline orchestration. Each ML service may depend on complex data

processing pipelines that include other ML components, requiring novel governance approaches for composite ML-driven systems.

8.3 Closing Remarks

The research presented in this thesis addresses fundamental challenges in modern distributed data processing systems where the tension between data utility maximization and privacy protection requires sophisticated governance approaches. The policy-driven framework we developed provides a systematic foundation for balancing these competing objectives while maintaining the flexibility needed for diverse analytical scenarios and stakeholder requirements.

The practical validation through industrial integration demonstrates that quality-privacy optimization can be achieved with acceptable computational overhead and operational complexity, making the approach viable for real-world deployment. The successful adaptation to machine learning workloads further establishes the framework's relevance for contemporary data-driven applications where ML models are central to analytical value creation.

Looking forward, the increasing importance of collaborative data analytics across organizational boundaries, combined with evolving privacy regulations and quality expectations, underscores the continued relevance of research in this area. The foundation established in this thesis provides a starting point for addressing more complex governance challenges in federated learning, cross-border data sharing, and AI-driven system orchestration.

The ultimate goal of enabling trustworthy, collaborative data processing while protecting individual privacy and maintaining analytical utility remains an active area of research and development, with significant implications for how organizations leverage data assets in an increasingly connected and regulated digital environment.

Bibliography

[1] M. Abrahamowicz. “Bias Due to Aggregation of Individual Covariates in the Cox Regression Model”.
In: *American Journal of Epidemiology* 160.7 (Oct. 2004), pp. 696–706.
ISSN: 0002-9262. DOI: 10.1093/aje/kwh266.

[2] T. Adams, C. Birkenbihl, K. Otte, H. G. Ng, J. A. Rieling, A.-F. Näher, U. Sax, F. Prasser, and H. Fröhlich. “On the Fidelity versus Privacy and Utility Trade-off of Synthetic Patient Data”.
In: *iScience* 28.5 (May 2025), p. 112382. ISSN: 25890042.
DOI: 10.1016/j.isci.2025.112382.

[3] J. Ahlbrandt, D. Brammen, R. Majeed, R. Lefering, S. Semler, S. Thun, F. Walcher, and R. R ohrig.
“Balancing the need for big data and patient data privacy–an IT infrastructure for a decentralized emergency care research database”.
In: *Studies in health technology and informatics* 205 (2014), pp. 750–754.
DOI: 10.3233/978-1-61499-432-9-750.

[4] M. M. B. Aissa, L. Sfaxi, and R. Robbana. “DECIDE: A New Decisional Big Data Methodology for a Better Data Governance”. In: *Proc. of EMCIS*. Vol. 402. Springer. Dubai, EAU, Nov. 2020, pp. 63–78.
DOI: 10.1007/978-3-030-63396-7_5.

[5] D. A. Albertini, B. Carminati, and E. Ferrari.
“An Extended Access Control Mechanism Exploiting Data Dependencies”.
In: *International Journal of Information Security* 16.1 (Feb. 2017), pp. 75–89.
ISSN: 1615-5262, 1615-5270. DOI: 10.1007/s10207-016-0322-4.

[6] Amazon Web Services. *AWS Macie*.
Available at: https://aws.amazon.com/macie/. 2023.
(Visited on 2023-10-31).

[7] M. Anisetti, C. A. Ardagna, C. Braghin, E. Damiani, A. Polimeno, and A. Balestrucci. “Dynamic and Scalable Enforcement of Access Control Policies for Big Data”. In: *Proceedings of the 13th International Conference on Management of Digital EcoSystems*.

Virtual Event Tunisia: ACM, Nov. 2021, pp. 71–78. ISBN: 978-1-4503-8314-1. DOI: 10.1145/3444757.3485107.

[8] M. Anisetti, C. A. Ardagna, C. Braghin, E. Damiani, A. Polimeno, and A. Balestrucci. "Dynamic and Scalable Enforcement of Access Control Policies for Big Data". In: *Proc. of MEDES*. Hammamet, Tunisia, Nov. 2021.

[9] M. Anisetti, N. Bena, F. Berto, and G. Jeon. "A DevSecOps-based Assurance Process for Big Data Analytics". In: *Proc. of IEEE ICWS 2022*. Barcelona, Spain, July 2022. DOI: 10.1109/ICWS55610.2022.00017.

[10] *Apache Atlas: Data Governance and Metadata Framework for Hadoop*. https://atlas.apache.org/. Apache Software Foundation; version 1.2.0 (or the version you used). 2019.

[11] Apache Software Foundation. *Apache Ranger: Framework to enable, monitor, and manage comprehensive data security across the Hadoop platform*. https://ranger.apache.org/. Accessed: 2024-10-30.

[12] Apache Software Foundation. *Apache Sentry*. Available at: https://sentry.apache.org/. 2023. (Visited on 2023-10-31).

[13] O. Arazy and R. Kopak. "On the Measurability of Information Quality". In: *Journal of the American Society for Information Science and Technology* 62.1 (Jan. 2011), pp. 89–99. ISSN: 15322882. DOI: 10.1002/asi.21447. (Visited on 2025-08-27).

[14] C. A. Ardagna, N. Bena, C. Hebert, M. Krotsiani, C. Kloukinas, and G. Spanoudakis. "Big Data Assurance: An Approach Based on Service- Level Agreements". In: *Big Data* 11 (3 2023). DOI: 10.1089/big.2021.0369.

[15] Z. Aslanyan and P. Vasilikos. *Privacy-Preserving Machine Learning - A Practical Guide*. Tech. rep. The Alexandra Institute, Oct. 2020. URL: https://www.alexandra.dk/wp-content/uploads/2020/10/Alexandra-Instituttet-whitepaper-Privacy-Preserving-Machine-Learning-A-Practical-Guide.pdf.

[16] M. Atzori, A. Ciaramella, C. Diamantini, B. Martino, S. Distefano, T. Facchinetti, F. Montecchiani, A. Nocera, G. Ruffo, R. Trasarti, et al. "Dataspaces: Concepts, architectures and initiatives". In: *CEUR workshop proceedings*. Vol. 3606. CEUR-WS. 2024.

[17] A. Al-Badi, A. Tarhini, and A. I. Khan.
"Exploring Big Data Governance Frameworks".
In: *Procedia Computer Science* 141 (2018), pp. 271–277.
DOI: 10.1016/j.procs.2018.10.181.

[18] R. Bayardo and R. Agrawal.
"Data Privacy through Optimal K-Anonymization".
In: *21st International Conference on Data Engineering (ICDE'05)*.
Tokyo, Japan: IEEE, 2005, pp. 217–228. ISBN: 978-0-7695-2285-2.
DOI: 10.1109/ICDE.2005.42.

[19] J. Brickell and V. Shmatikov. "The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing".
In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
Las Vegas Nevada USA: ACM, Aug. 2008, pp. 70–78.
ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401904.

[20] L. Cai and Y. Zhu. "The Challenges of Data Quality and Data Quality Assessment in the Big Data Era".
In: *Data Science Journal* 14.0 (May 2015), p. 2. ISSN: 1683-1470.
DOI: 10.5334/dsj-2015-002. (Visited on 2024-07-16).

[21] J. Cao, B. Carminati, E. Ferrari, and K.-L. Tan.
"ACStream: Enforcing Access Control over Data Streams".
In: *2009 IEEE 25th International Conference on Data Engineering*.
Shanghai, China: IEEE, Mar. 2009, pp. 1495–1498. ISBN: 978-1-4244-3422-0.
DOI: 10.1109/ICDE.2009.25.

[22] B. Carminati, E. Ferrari, J. Cao, and K. L. Tan.
"A Framework to Enforce Access Control over Data Streams". In: *ACM Transactions on Information and System Security* 13.3 (July 2010), pp. 1–31.
ISSN: 1094-9224, 1557-7406. DOI: 10.1145/1805974.1805984.

[23] B. Carminati, E. Ferrari, and A. Perego.
"Enforcing Access Control in Web-based Social Networks". In: *ACM Transactions on Information and System Security* 13.1 (Oct. 2009), pp. 1–38.
ISSN: 1094-9224, 1557-7406. DOI: 10.1145/1609956.1609962.

[24] B. Carminati, E. Ferrari, and K. L. Tan.
"Enforcing Access Control over Data Streams". In: *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*.
Sophia Antipolis France: ACM, June 2007, pp. 21–30.
ISBN: 978-1-59593-745-2. DOI: 10.1145/1266840.1266845.

[25] M. Ceci, R. Corizzo, N. Japkowicz, P. Mignone, and G. Pio. "ECHAD: Embedding-Based Change Detection From Multivariate Time Series in Smart Grids". In: *IEEE Access* (2020). DOI: 10.1109/ACCESS.2020.3019095.

[26] J. Chabin, C. D. A. Ciferri, M. Halfeld-Ferrari, C. S. Hara, and R. R. M. Penteado. "Role-Based Access Control on Graph Databases". In: *SOFSEM 2021: Theory and Practice of Computer Science*. Ed. by T. Bureš, R. Dondi, J. Gamper, G. Guerrini, T. Jurdziński, C. Pahl, F. Sikora, and P. W. Wong. Vol. 12607. Cham: Springer International Publishing, 2021, pp. 519–534. ISBN: 978-3-030-67730-5 978-3-030-67731-2. DOI: 10.1007/978-3-030-67731-2_38.

[27] C. Chen, M. A. Elsayed, and M. Zulkernine. "HBD-Authority: Streaming Access Control Model for Hadoop". In: *2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys)*. Nadi, Fiji: IEEE, Dec. 2020, pp. 16–25. ISBN: 978-1-7281-7651-2. DOI: 10.1109/DependSys51298.2020.00012.

[28] V. Ciriani, S. De Capitani Di Vimercati, S. Foresti, and P. Samarati. "κ-Anonymity". In: *Secure Data Management in Decentralized Systems*. Ed. by T. Yu and S. Jajodia. Vol. 33. Boston, MA: Springer US, 2007, pp. 323–353. ISBN: 978-0-387-27694-6 978-0-387-27696-0. DOI: 10.1007/978-0-387-27696-0_10.

[29] P. Colombo and E. Ferrari. "Access control technologies for Big Data management systems: literature review and future trends". In: *Cybersecurity* 2.1 (Jan. 2019), p. 3. DOI: https://doi.org/10.1186/s42400-018-0020-9.

[30] R. Corizzo, M. Ceci, G. Pio, P. Mignone, and N. Japkowicz. "Spatially-Aware Autoencoders for Detecting Contextual Anomalies in Geo-Distributed Data". In: *Proc. of DS*. Halifax, NS, Canada, Oct. 2021. ISBN: 978-3-030-88942-5. DOI: 10.1007/978-3-030-88942-5_36.

[31] S. Creese, P. Hopkins, S. Pearson, and Y. Shen. "Data Protection-Aware Design for Cloud Services". In: *Cloud Computing*. Ed. by M. G. Jaatun, G. Zhao, and C. Rong. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 119–130. ISBN: 978-3-642-10665-1. DOI: 10.1007/978-3-642-10665-1_11.

[32] D. De Pascale, G. Cascavilla, D. A. Tamburri, and W.-J. Van Den Heuvel. "Real-World K-Anonymity Applications: The KGen Approach and Its Evaluation in Fraudulent Transactions".
In: *Information Systems* 115 (May 2023), p. 102193. ISSN: 03064379.
DOI: 10.1016/j.is.2023.102193.

[33] C. Dwork. "Differential Privacy: A Survey of Results".
In: *Theory and Applications of Models of Computation*.
Ed. by M. Agrawal, D. Du, Z. Duan, and A. Li. Vol. 4978.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–19.
ISBN: 978-3-540-79227-7 978-3-540-79228-4.
DOI: 10.1007/978-3-540-79228-4_1.

[34] EU.
The General Data Protection Regulation (GDPR) - Regulation (EU) 2016/679.
May 25, 2018. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504&qid=1532348683434.

[35] *European Data Governance Act*. en.
URL: https://digital-strategy.ec.europa.eu/en/policies/data-governance-act (visited on 2022-08).

[36] European Parliament and Council of the European Union.
Regulation (EU) 2022/2065 of the European Parliament and of the Council of 19 October 2022 on a Single Market for Digital Services (Digital Services Act).
Official Journal of the European Union, L 277/1–102, 27 October 2022.
Entry into force 17 February 2024; some provisions effective as of 16 November 2022. 2022.

[37] European Parliament and Council of the European Union.
Regulation (EU) 2025/xxx of the European Parliament and of the Council on the European Health Data Space (EHDS).
Official Journal of the European Union, L xx/xx, 5 March 2025.
Entry into force 26 March 2025; implementing acts by March 2027; full application phases from 2029–2031. 2025.
URL: https://health.ec.europa.eu/ehealth-digital-health-and-care/european-health-data-space_en.

[38] W. Fan. "Data Quality: From Theory to Practice".
In: *ACM SIGMOD Record* 44.3 (Dec. 2015), pp. 7–18. ISSN: 0163-5808.
DOI: 10.1145/2854006.2854008. (Visited on 2025-08-27).

[39] N. Farhadighalati, L. A. Estrada-Jimenez, S. Nikghadam-Hojjati, and J. Barata. “A Systematic Review of Access Control Models: Background, Existing Research, and Challenges”.
In: *IEEE Access* 13 (2025), pp. 17777–17806. ISSN: 2169-3536.
DOI: 10.1109/ACCESS.2025.3533145.

[40] Federal Government of Brazil.
Lei Geral de Proteção de Dados Pessoais (LGPD), Law No. 13,709.
Official Gazette of the Federative Republic of Brazil, 14 August 2018.
Effective 18 September 2020. 2018.
URL: https://www.gov.br/pt-br/noticias/saude-e-vigilancia-sanitaria/2020/08/lgpd-entra-em-vigor.

[41] N. H. Fefferman, E. A. O’Neil, and E. N. Naumova. “Confidentiality and Confidence: Is Data Aggregation a Means to Achieve Both?”
In: *Journal of Public Health Policy* 26.4 (Dec. 2005), pp. 430–449.
ISSN: 0197-5897, 1745-655X. DOI: 10.1057/palgrave.jphp.3200029.

[42] S. Fletcher and M. Z. Islam.
“Quality Evaluation of an Anonymized Dataset”.
In: *2014 22nd International Conference on Pattern Recognition.*
Stockholm, Sweden: IEEE, Aug. 2014, pp. 3594–3599.
ISBN: 978-1-4799-5209-0. DOI: 10.1109/ICPR.2014.618.
(Visited on 2025-08-26).

[43] B. Fuglede and F. Topsoe.
“Jensen-Shannon divergence and Hilbert space embedding”.
In: *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* IEEE. DOI: 10.1109/isit.2004.1365067.
URL: http://dx.doi.org/10.1109/ISIT.2004.1365067.

[44] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu.
“Privacy-preserving data publishing: A survey of recent developments”.
In: *ACM Comput. Surv.* 42.4 (June 2010). ISSN: 0360-0300.
DOI: 10.1145/1749603.1749605.
URL: https://doi.org/10.1145/1749603.1749605.

[45] B. C. Fung, K. Wang, A. W.-C. Fu, and P. S. Yu. *Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques.* 1st.
New York: Chapman & Hall/CRC, 2010. ISBN: 1420091484.
DOI: 10.1201/9781420091502.

[46] P. Geetha, C. Naikodi, and S. L. N. Setty.
"Design of Big Data Privacy Framework—A Balancing Act".
In: *Advances in Data Sciences, Security and Applications*.
Ed. by V. Jain, G. Chaudhary, M. C. Taplamacioglu, and M. S. Agarwal.
Singapore: Springer Singapore, 2020, pp. 253–265. ISBN: 978-981-15-0372-6.
DOI: 10.1007/978-981-15-0372-6_19.

[47] Y. Gong, G. Liu, Y. Xue, R. Li, and L. Meng.
"A survey on dataset quality in machine learning".
In: *Information and Software Technology* 162 (2023), p. 107268.
ISSN: 0950-5849.
DOI: https://doi.org/10.1016/j.infsof.2023.107268. URL: https://www.sciencedirect.com/science/article/pii/S0950584923001222.

[48] Google Cloud. *Google Cloud Data Loss Prevention (DLP)*.
Available at: https://cloud.google.com/dlp. 2023.
(Visited on 2023-10-31).

[49] N. Gruschka, V. Mavroeidis, K. Vishi, and M. Jensen. "Privacy Issues and Data Protection in Big Data: A Case Study Analysis under GDPR".
In: *2018 IEEE International Conference on Big Data (Big Data)*.
Seattle, WA, USA: IEEE, Dec. 2018, pp. 5027–5033. ISBN: 978-1-5386-5035-6.
DOI: 10.1109/BigData.2018.8622621.

[50] V. N. Gudivada, A. W. Apon, and J. Ding.
"Data Quality Considerations for Big Data and Machine Learning: Going Beyond Data Cleaning and Transformations". In: 2017.
URL: https://api.semanticscholar.org/CorpusID:221131081.

[51] M. Gupta, F. Patwa, and R. Sandhu. "An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem". In:
Proceedings of the Third ACM Workshop on Attribute-Based Access Control.
Tempe AZ USA: ACM, Mar. 2018, pp. 13–24. ISBN: 978-1-4503-5633-6.
DOI: 10.1145/3180457.3180463.

[52] M. Gupta, F. Patwa, and R. Sandhu.
"POSTER: Access Control Model for the Hadoop Ecosystem".
In: *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*. Indianapolis Indiana USA: ACM, June 2017, pp. 125–127.
ISBN: 978-1-4503-4702-0. DOI: 10.1145/3078861.3084164.

[53] Haihua Chen.
"Data Quality Evaluation and Improvement for Machine Learning".
In: (2022). DOI: 10.13140/RG.2.2.15870.87361.

[54] M. U. Hassan, M. H. Rehmani, and J. Chen.
"Differential Privacy Techniques for Cyber Physical Systems: A Survey".
In: *IEEE Communications Surveys & Tutorials* 22.1 (2020), pp. 746–789.
ISSN: 1553-877X, 2373-745X. DOI: 10.1109/COMST.2019.2944748.

[55] V. Hotz, C. Bollinger, T. Komarova, C. Manski, R. Moffitt, D. Nekipelov, A. Sojourner, and B. Spencer.
"Balancing data privacy and usability in the federal statistical system".
In: *Proceedings of the National Academy of Sciences* 119 (Aug. 2022).
DOI: 10.1073/pnas.2104906119.

[56] L. Huang, Y. Zhu, X. Wang, and F. Khurshid.
"An Attribute-Based Fine-Grained Access Control Mechanism for HBase".
In: *Database and Expert Systems Applications*. Ed. by S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa, and I. Khalil. Vol. 11706.
Cham: Springer International Publishing, 2019, pp. 44–59.
ISBN: 978-3-030-27614-0 978-3-030-27615-7.
DOI: 10.1007/978-3-030-27615-7_4.

[57] IBM. *IBM Security Guardium Data Protection.*
Available at: https://www.ibm.com/products/guardium. 2023.
(Visited on 2023-10-31).

[58] P. Ilia, B. Carminati, E. Ferrari, P. Fragopoulou, and S. Ioannidis.
"SAMPAC: Socially-Aware Collaborative Multi-Party Access Control".
In: *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. Scottsdale Arizona USA: ACM, Mar. 2017, pp. 71–82.
ISBN: 978-1-4503-4523-1. DOI: 10.1145/3029806.3029834.

[59] E. Im, H. Kim, H. Lee, X. Jiang, and J. H. Kim. "Exploring the Tradeoff between Data Privacy and Utility with a Clinical Data Analysis Use Case".
In: *BMC Medical Informatics and Decision Making* 24.1 (May 2024), p. 147.
ISSN: 1472-6947. DOI: 10.1186/s12911-024-02545-9.

[60] International Data Spaces Association.
International Data Spaces Association: Reference Architecture Model.
https://internationaldataspaces.org/. Accessed: 2025-08-19. 2023.

[61] V. S. Iyengar. "Transforming Data to Satisfy Privacy Constraints".
In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
Edmonton Alberta Canada: ACM, July 2002, pp. 279–288.
ISBN: 978-1-58113-567-1. DOI: 10.1145/775047.775089.

[62] M. Jensen. “Challenges of Privacy Protection in Big Data Analytics”.
In: *2013 IEEE International Congress on Big Data*.
Santa Clara, CA, USA: IEEE, June 2013, pp. 235–238.
ISBN: 978-0-7695-5006-0. DOI: 10.1109/BigData.Congress.2013.39.

[63] M. Jensen, J. Schwenk, N. Gruschka, and L. L. Iacono.
“On Technical Security Issues in Cloud Computing”.
In: *2009 IEEE International Conference on Cloud Computing*.
Bangalore, India: IEEE, 2009, pp. 109–116. ISBN: 978-1-4244-5199-9.
DOI: 10.1109/CLOUD.2009.60.

[64] A. Kalam, R. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte,
A. Miege, C. Saurel, and G. Trouessin.
“Organization Based Access Control”.
In: *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*.
Lake Como, Italy: IEEE Comput. Soc, 2003, pp. 120–131.
ISBN: 978-0-7695-1933-3. DOI: 10.1109/POLICY.2003.1206966.

[65] Y. Kanza, A. O. Mendelzon, R. J. Miller, and Z. Zhang.
“Authorization-Transparent Access Control for XML Under the
Non-Truman Model”. In: *Advances in Database Technology - EDBT 2006*.
Ed. by D. Hutchison et al. Vol. 3896.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 222–239.
ISBN: 978-3-540-32960-2 978-3-540-32961-9. DOI: 10.1007/11687238_16.

[66] H. Kellerer, U. Pferschy, and D. Pisinger.
“Multidimensional Knapsack Problems”. In: *Knapsack Problems*.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 235–283.
ISBN: 978-3-540-24777-7. DOI: 10.1007/978-3-540-24777-7_9.

[67] N. Li, T. Li, and S. Venkatasubramanian.
“T-Closeness: Privacy Beyond k-Anonymity and l-Diversity”.
In: *2007 IEEE 23rd International Conference on Data Engineering*.
Istanbul: IEEE, Apr. 2007, pp. 106–115. ISBN: 978-1-4244-0802-3.
DOI: 10.1109/ICDE.2007.367856.

[68] W. Lindner and J. Meier.
“Towards a Secure Data Stream Management System”.
In: *Trends in Enterprise Application Architecture*.
Ed. by D. Draheim and G. Weber. Vol. 3888.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 114–128.
ISBN: 978-3-540-32734-9 978-3-540-32735-6. DOI: 10.1007/11681885_10.

[69] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. "*L* -Diversity: Privacy beyond *k* -Anonymity". In: *ACM Transactions on Knowledge Discovery from Data* 1.1 (Mar. 2007), p. 3. ISSN: 1556-4681, 1556-472X. DOI: 10.1145/1217299.1217302.

[70] A. Majeed and S. Lee. "Anonymization Techniques for Privacy Preserving Data Publishing: A Comprehensive Survey". In: *IEEE Access* 9 (2021), pp. 8512–8545. DOI: 10.1109/ACCESS.2020.3045700. URL: https://api.semanticscholar.org/CorpusID:231616865.

[71] Microsoft. *Microsoft Presidio: Open-source tools for differential privacy and PII detection.* https://github.com/microsoft/presidio. Accessed: 2024-10-30.

[72] R. Nagpal, U. Usua, R. Palacios, and A. Gupta. "FairRAG: A Privacy-Preserving Framework for Fair Financial Decision-Making". In: *Applied Sciences* 15.15 (July 2025), p. 8282. ISSN: 2076-3417. DOI: 10.3390/app15158282.

[73] J. Natal, I. Ávila, V. B. Tsukahara, M. Pinheiro, and C. D. Maciel. "Entropy: From thermodynamics to information processing". In: *Entropy* 23.10 (2021).

[74] National Institute of Standards and Technology. *NIST PRIVACY FRAMEWORK: A TOOL FOR IMPROVING PRIVACY THROUGH ENTERPRISE RISK MANAGEMENT, VERSION 1.0.* Tech. rep. NIST CSWP 01162020. Gaithersburg, MD: National Institute of Standards and Technology, Jan. 2020, NIST CSWP 01162020. DOI: 10.6028/NIST.CSWP.01162020.

[75] National People's Congress of China. *Personal Information Protection Law of the People's Republic of China (PIPL).* Effective 1 November 2021. China's first comprehensive data protection law. 2021. URL: http://www.npc.gov.cn/englishnpc/c23934/202108/93adf9d61dc94e9dba7f94c6a7ab30bb.shtml.

[76] L. Ohno-Machado. "Effects of Data Anonymization by Cell Suppression on Descriptive Statistics and Predictive Modeling Performance". In: *Journal of the American Medical Informatics Association* 9.90061 (Nov. 2002), 115S–119. ISSN: 1067-5027, 1527-974X. DOI: 10.1197/jamia.M1241.

[77] *OpenMetadata: Open and Unified Metadata Platform.* https://open-metadata.org/. Open source metadata store for discovery, lineage, observability, governance. 2024.

[78] Parliament of Canada. *Personal Information Protection and Electronic Documents Act (PIPEDA).* S.C. 2000, c. 5. Effective 1 January 2001; governs private-sector data handling in Canada. 2000. URL: https://laws-lois.justice.gc.ca/eng/acts/P-8.6/.

[79] A. Polimeno, C. Braghin, M. Anisetti, and C. A. Ardagna. "Maximizing data quality while ensuring data protection in service-based data pipelines". In: *Journal of Big Data* 12 (2025).

[80] D. Profeta, N. Masi, D. Messina, D. Dalle Carbonare, S. Bonura, and V. Morreale. "A novel micro-service based platform for composition, deployment and execution of BDA applications". In: *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).* IEEE. 2019, pp. 182–185.

[81] M. Rahman, M. R. Hassan, and R. Buyya. "Jaccard Index based availability prediction in enterprise grids". In: *Procedia Computer Science* 1.1 (2010). ICCS 2010, pp. 2707–2716. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2010.04.304. URL: https://www.sciencedirect.com/science/article/pii/S1877050910003054.

[82] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. "Extending Query Rewriting Techniques for Fine-Grained Access Control". In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data.* Paris France: ACM, June 2004, pp. 551–562. ISBN: 978-1-58113-859-7. DOI: 10.1145/1007568.1007631.

[83] M. Saifuzzaman, T. N. Ananna, M. J. M. Chowdhury, M. S. Ferdous, and F. Chowdhury. "A Systematic Literature Review on Wearable Health Data Publishing under Differential Privacy". In: *International Journal of Information Security* 21.4 (Aug. 2022), pp. 847–872. ISSN: 1615-5262, 1615-5270. DOI: 10.1007/s10207-021-00576-1.

[84] P. Samarati. "Protecting Respondents Identities in Microdata Release". In: *IEEE Transactions on Knowledge and Data Engineering* 13.6 (Dec. 2001), pp. 1010–1027. ISSN: 10414347. DOI: 10.1109/69.971193.

[85] P. Samarati and L. Sweeney.
"Protecting Privacy When Disclosing Information: K-Anonymity and Its Enforcement through Generalization and Suppression". In: (1998), 0 Bytes.
DOI: 10.1184/R1/6625469.V1.

[86] R. Sandhu.
"Role Hierarchies and Constraints for Lattice-Based Access Controls".
In: *Computer Security — ESORICS 96*. Ed. by G. Goos, J. Hartmanis, J. Van Leeuwen, E. Bertino, H. Kurth, G. Martella, and E. Montolivo.
Vol. 1146. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 65–79.
ISBN: 978-3-540-61770-9 978-3-540-70675-5.
DOI: 10.1007/3-540-61770-1_28.

[87] D. Schwabe, K. Becker, M. Seyferth, A. Klaß, and T. Schaeffter.
"The METRIC-framework for Assessing Data Quality for Trustworthy AI in Medicine: A Systematic Review".
In: *npj Digital Medicine* 7.1 (Aug. 2024), p. 203. ISSN: 2398-6352.
DOI: 10.1038/s41746-024-01196-4.

[88] A. Sharma, G. Singh, and S. Rehman.
"A Review of Big Data Challenges and Preserving Privacy in Big Data".
In: *Advances in Data and Information Sciences*.
Ed. by M. L. Kolhe, S. Tiwari, M. C. Trivedi, and K. K. Mishra.
Singapore: Springer Singapore, 2020, pp. 57–65. ISBN: 978-981-15-0694-9.
DOI: 10.1007/978-981-15-0694-9_7.

[89] State of California. *California Consumer Privacy Act of 2018 (CCPA)*.
California Civil Code, Sections 1798.100–1798.199.
Effective 1 January 2020; amended by CPRA in 2023. 2018.
URL: https://oag.ca.gov/privacy/ccpa.

[90] L. Sweeney. "ACHIEVING K-ANONYMITY PRIVACY PROTECTION USING GENERALIZATION AND SUPPRESSION".
In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (Oct. 2002), pp. 571–588. ISSN: 0218-4885, 1793-6411.
DOI: 10.1142/S021848850200165X.

[91] I. Taleb, M. A. Serhani, C. Bouhaddioui, and R. Dssouli. "Big Data Quality Framework: A Holistic Approach to Continuous Quality Management".
In: *Journal of Big Data* 8.1 (Dec. 2021), p. 76. ISSN: 2196-1115.
DOI: 10.1186/s40537-021-00468-0.

[92] A. Tembhare, S. Sibi Chakkaravarthy, D. Sangeetha, V. Vaidehi, and M. Venkata Rathnam. "Role-Based Policy to Maintain Privacy of Patient Health Records in Cloud".
In: *The Journal of Supercomputing* 75.9 (Sept. 2019), pp. 5866–5881. ISSN: 0920-8542, 1573-0484. DOI: 10.1007/s11227-019-02887-6.

[93] R. K. Thomas. "Team-Based Access Control (TMAC): A Primitive for Applying Role-Based Access Controls in Collaborative Environments".
In: *Proceedings of the Second ACM Workshop on Role-based Access Control - RBAC '97*. Fairfax, Virginia, United States: ACM Press, 1997, pp. 13–19. ISBN: 978-0-89791-985-2. DOI: 10.1145/266741.266748.

[94] U.S. Congress. *Health Insurance Portability and Accountability Act of 1996*. https://www.govinfo.gov/content/pkg/PLAW-104publ191/html/PLAW-104publ191.htm.
Public Law 104-191, 110 Stat. 1936. 1996.
URL: https://www.govinfo.gov/content/pkg/PLAW-104publ191/html/PLAW-104publ191.htm.

[95] UK Parliament. *Data Protection Act 2018*. https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted.
United Kingdom Public General Act 2018, c.12. 2018.

[96] M. Usman, M. L. Bernardi, and M. Cimitile.
"Introducing a Quality-Driven Approach for Federated Learning".
In: *Sensors* 25.10 (May 2025), p. 3083. ISSN: 1424-8220.
DOI: 10.3390/s25103083.

[97] T. van den Broek and A. F. van Veenstra.
"Governance of big data collaborations: How to balance regulatory compliance and disruptive innovation".
In: *Technological Forecasting and Social Change* 129 (2018), pp. 330–338. ISSN: 0040-1625.
DOI: https://doi.org/10.1016/j.techfore.2017.09.040.
URL: https://www.sciencedirect.com/science/article/pii/S0040162517314695.

[98] J. Wang, Y. Liu, P. Li, Z. Lin, S. Sindakis, and S. Aggarwal.
"Overview of Data Quality: Examining the Dimensions, Antecedents, and Impacts of Data Quality".
In: *Journal of the Knowledge Economy* 15.1 (Feb. 2023), pp. 1159–1178. ISSN: 1868-7873. DOI: 10.1007/s13132-022-01096-6.
URL: http://dx.doi.org/10.1007/s13132-022-01096-6.

[99] R. Y. Wang and D. M. Strong.
"Beyond Accuracy: What Data Quality Means to Data Consumers".
In: *Journal of Management Information Systems* 12.4 (Mar. 1996), pp. 5–33.
ISSN: 0742-1222, 1557-928X. DOI: 10.1080/07421222.1996.11518099.

[100] X. Xiao, K. Yi, and Y. Tao.
"The Hardness and Approximation Algorithms for L-Diversity".
In: *Proceedings of the 13th International Conference on Extending Database Technology*. Lausanne Switzerland: ACM, Mar. 2010, pp. 135–146.
ISBN: 978-1-60558-945-9. DOI: 10.1145/1739041.1739060.

[101] J. Xie, L. Sun, and Y. F. Zhao.
"On the Data Quality and Imbalance in Machine Learning-based Design and Manufacturing—A Systematic Review".
In: *Engineering* 45 (Feb. 2025), pp. 105–131. ISSN: 20958099.
DOI: 10.1016/j.eng.2024.04.024.

[102] X. Xiong, S. Liu, D. Li, Z. Cai, and X. Niu.
"A Comprehensive Survey on Local Differential Privacy". In: *Security and Communication Networks* 2020 (Oct. 2020). Ed. by A. M. Del Rey, pp. 1–29.
ISSN: 1939-0122, 1939-0114. DOI: 10.1155/2020/8829523.

[103] T. Xue, Y. Wen, B. Luo, B. Zhang, Y. Zheng, Y. Hu, Y. Li, G. Li, and D. Meng.
"GuardSpark++: Fine-Grained Purpose-Aware Access Control for Secure Data Sharing and Analysis in Spark".
In: *Annual Computer Security Applications Conference*.
Austin USA: ACM, Dec. 2020, pp. 582–596. ISBN: 978-1-4503-8858-0.
DOI: 10.1145/3427228.3427640.

[104] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer.
"Quality Assessment for Linked Data: A Survey: A Systematic Literature Review and Conceptual Framework".
In: *Semantic Web* 7.1 (Mar. 2015). Ed. by P. Hitzler, pp. 63–93.
ISSN: 22104968, 15700844. DOI: 10.3233/SW-150175.

[105] L. Zhang, Z. You, K. Wang, and Z. Cui.
"Research on Access Control Scheme of System Wide Information Management Based on Attribute Association". In: *Security and Communication Networks* 2022 (May 2022). Ed. by B. Wang, pp. 1–15.
ISSN: 1939-0122, 1939-0114. DOI: 10.1155/2022/6181995.

[106] Y. Zhou, F. Tu, K. Sha, J. Ding, and H. Chen.
A Survey on Data Quality Dimensions and Tools for Machine Learning.
June 2024. arXiv: 2406.19614 [cs].

[107] Y. Zhou, F. Tu, K. Sha, J. Ding, and H. Chen.
A Survey on Data Quality Dimensions and Tools for Machine Learning. 2024.
DOI: 10.48550/ARXIV.2406.19614.
URL: https://arxiv.org/abs/2406.19614.

[108] T. Zhu, G. Li, W. Zhou, and P. S. Yu. *Differential Privacy and Applications*.
Vol. 69. Advances in Information Security.
Cham: Springer International Publishing, 2017.
ISBN: 978-3-319-62002-2 978-3-319-62004-6.
DOI: 10.1007/978-3-319-62004-6.

[109] F. Zouari, C. Ghedira-Guegan, K. Boukadi, and N. Kabachi.
"A Semantic and Service-Based Approach for Adaptive Mutli-Structured Data Curation in Data Lakehouses".
In: *World Wide Web* 26.6 (Nov. 2023), pp. 4001–4023.
ISSN: 1386-145X, 1573-1413. DOI: 10.1007/s11280-023-01218-3.

Appendices

A

Publications

This appendix lists the publications resulting from the research presented in this thesis.

P1: Balancing Protection and Quality in Big Data Analytics Pipelines

Co-authors: Paolo Mignone, Chiara Braghin, Marco Anisetti, Michelangelo Ceci, Donato Malerba, Claudio A. Ardagna

In: Big Data, vol. 13, no. 2, pp. 127-143, 2025, Mary Ann Liebert, Inc.

Abstract: Existing data engine implementations do not properly manage the conflict between the need of protecting and sharing data, which is hampering the spread of big data applications and limiting their impact. These two requirements have often been studied and defined independently, leading to a conceptual and technological misalignment. This article presents the architecture and technical implementation of a data engine addressing this conflict by integrating a new governance solution based on access control within a big data analytics pipeline. Our data engine enriches traditional components for data governance with an access control system that enforces access to data in a big data environment based on data transformations. Data are then used along the pipeline only after sanitization, protecting sensitive attributes before their usage, in an effort to facilitate the balance between protection and quality. The solution was tested in a real-world smart city scenario using the data of the Oslo city transportation system. Specifically, we compared the different predictive models trained with the data views obtained by applying the secure transformations required by different

user roles to the same data set. The results show that the predictive models, built on data manipulated according to access control policies, are still effective.

P2: Maximizing data quality while ensuring data protection in service-based data pipelines

Co-authors: Chiara Braghin, Marco Anisetti, Claudio A. Ardagna

In: Journal of Big Data, vol. 12, no. 1, pp. 62, 2025, Springer International Publishing

Abstract: The growing capacity to handle vast amounts of data, combined with a shift in service delivery models, has improved scalability and efficiency in data analytics, particularly in multi-tenant environments. Data are treated as digital products and processed through orchestrated service-based data pipelines. However, advancements in data analytics do not find a counterpart in data governance techniques, leaving a gap in the effective management of data throughout the pipeline lifecycle. This gap highlights the need for innovative service-based data pipeline management solutions that prioritize balancing data quality and data protection. The framework proposed in this paper optimizes service selection and composition within service-based data pipelines to maximize data quality while ensuring compliance with data protection requirements, expressed as access control policies. Given the NP-hard nature of the problem, a sliding-window heuristic is defined and evaluated against the exhaustive approach and a baseline modeling the state of the art. Our results demonstrate a significant reduction in computational overhead, while maintaining high data quality.

P3: Data Pipelines Assessment: The Role of Data Engine Deployment Models

Co-authors: Claudio A. Ardagna, Valerio Bellandi, Marco Luzzara

In: CEUR Workshop Proceedings, pp. 1-13, 2024

Abstract: In this paper, we explore different deployment models for data engines and elucidate their implications on data pipeline behavior. Specifically, we examine the impact on data sharing, data protection, pipeline uptime and latency, and the feasibility of moving segments of

typical data engines to the edge. Our work demonstrates the consequences of various deployment strategies on non-functional properties of data pipelines, focusing on availability, performance, and privacy. By considering the interplay between data engine deployment and data pipeline requirements, stakeholders can make informed decisions to optimize the efficiency and effectiveness of data-driven systems.

P4: Real-time anonymization of sensitive personal data using a service-based architecture

Co-authors: Fabio Giampaolo, Stefano Izzo, Stefano Siccardi, Valerio Bellandi, Francesco Piccialli

In: 2023 IEEE International Conference on Web Services (ICWS), pp. 701-703, 2023

Abstract: Anonymization is an important aspect of data privacy protection, especially in the context of sensitive personal information collected through sensors. In this paper, we propose a new service-based architecture for anonymizing such data in real-time, ensuring that data is accessible to authorized users while maintaining privacy. Our architecture is based on the annotation of data at ingestion time, where privacy levels are assigned to sets of columns. The anonymization procedure is performed by compressing and encoding the data through an autoencoder model, where the encoder and decoder functions are defined as parametric functions composed of multiple hidden layers.

P5: Knowledge-based legal document retrieval: A case study on italian civil court decisions

Co-authors: Valerio Bellandi, Silvana Castano, Paolo Ceravolo, Ernesto Damiani, Alfio Ferrara, Stefano Montanelli, Sergio Picascia, Davide Riva

In: CEUR Workshop Proceedings, vol. 3256, pp. 1-13, 2022

Abstract: In this paper, we present a knowledge-based approach for legal document retrieval based on the organization of a textual data repository and on document embedding models. Pre-processed and embedded documents are iteratively classified at sentence level through a terminology extraction and concept formation cycle, using a zero-knowledge approach that offers a high degree of flexibility with

regard to the integration of external knowledge and the variability of inputs, suitable to face the scarcity of annotated data and the specificity of terminology that feature the Italian legal domain document corpora.

P6: Security Assurance in Modern IoT Systems

Co-authors: Nicola Bena, Ruslan Bondaruc

In: 2022 IEEE 95th Vehicular Technology Conference (VTC2022-Spring), pp. 1-5, 2022

Abstract: Modern distributed systems consist of a multilayer architecture of IoT, edge, and cloud nodes. Together, they are revolutionizing our lives, bringing intelligence to existing processes (e.g., smart grids) and enabling novel, efficient and effective processes (e.g., remote surgery). This transition however does not come without drawbacks, due to the ever-increasing reliance on devices whose security and safety are, at least, questionable. In this context, research is in its infancy, struggling to adapt successful practices applied, for instance, in cloud systems. Security of modern IoT systems still relies on old-fashioned approaches, mostly static assessments considering only very specific parts of the target system, rather than assessing the system as a whole. In this paper, we put forward the idea of security assurance for IoT, as a higher-level assurance process evaluating the target system at different layers and different moments of its lifecycle, then implemented by a flexible assurance framework. The quality of our approach is evaluated in a real-world smart lighting system.

P7: Dynamic and scalable enforcement of access control policies for big data

Co-authors: Marco Anisetti, Claudio A. Ardagna, Chiara Braghin, Ernesto Damiani, Alessandro Balestrucci

In: Proceedings of the 13th International Conference on Management of Digital EcoSystems, pp. 71-78, 2021

Abstract: The conflict between the need of protecting and sharing data is hampering the spread of big data applications. Security and privacy assurance is required to protect data owners, while data access and sharing are fundamental to implement smart big data solu-

tions. In this context, access control systems can assume a central role in balancing data protection and data sharing. However, existing access control solutions are not general and scalable enough to address the software and technological complexity of big data ecosystems, being unable to support such a dynamic and collaborative environment. In this paper, we propose an access control system that enforces access to data in a distributed, multi-party big data environment. It is based on data annotations and secure data transformations performed at ingestion time. We show the feasibility of our approach in the smart city domain using an Apache-based big data engine.