Semantic Time Series Analytics Leveraging Incremental Knowledge Graph Enrichment

Von der Fakultät für Informatik der RWTH Aachen University
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
genehmigte Dissertation

vorgelegt von

Alexander Graß, M. Sc.

aus

Köln, Deutschland

Berichter: Univ.-Prof. Dr. rer. pol. Stefan Decker
Univ.-Prof. Dr. rer. nat. Christian Beecks

Tag der mündlichen Prüfung: 09.09.2026

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

Abstract

The development and utilization of modern, digital technologies, such as wearable devices or automated, intelligent machines, have led to a substantial increase in time series data across many domains including healthcare, finance and industry. Although this data holds significant potential for deriving actionable insights regarding strategic decision-making, a common problem of organizations is to systematically interpret and reuse the knowledge generated in analytical processes. While different task-specific analyses are carried out on collaboratively used storage systems, produced insights commonly stem from isolated workflows and remain implicit, fragmented, or tied to particular models and projects. This independent processing limits accessibility and reusability. Furthermore, generated insights are often produced without incorporating scenario-specific context and expert knowledge, which reduces their interpretability and degree of validation. In fact, no common solution for classifying and linking prior knowledge and derived insights across the full scope of time series analytics exists, which prevents synergies such as knowledge transfer and propagation.

This thesis addresses the challenges of systematic knowledge integration from and for time series analytics in combination with related contextual facts. It proposes a comprehensive, semantically enriched knowledge graph, which serves as a central, machine-interpretable repository that enables the structured storage, contextualization, and exploitation of knowledge associated with time series data. The core idea is a continuous enrichment of facts, making the graph a dynamically growing repository of exploitable insights. Conceptually, the work investigates research questions on

(i) how to semantically classify analysis results and domain knowledge,

(ii) how to represent and manage this knowledge and its propagation efficiently, and

(iii) how to exploit the knowledge effectively in subsequent analytical workflows.

The goal is to improve time series analytics through structured reuse of insights across analyses and support for informed decision-making.

Core contributions of this thesis include:

- Hierarchically-structured vocabularies for the classification of insights derived from data analysis and contextual knowledge associated with time series data.
- An ontology for representing the segment-based management, classification, and propagation of time series knowledge.
- An AI-driven approach for populating and validating ontologies from program code and its associated runtime context.
- An algebraic specification to define knowledge propagation in informed analytical pipelines.
- A systematic evaluation of embedding-based retrieval strategies to assess the suitability of the developed ontology's structural patterns for diverse retrieval tasks.
- The formalization of grammar-based rules and conditional logic to link model specifications with concrete data setups for automated time series knowledge exploration.
- The integration of natural language query interpretation with incrementally evolving knowledge graphs to enable automated, semantically enriched data analyses.

Kurzfassung

Die Entwicklung moderner, digitaler Technologien, wie etwa smarter Nutzergeräte oder automatisierter, intelligenter Maschinen, hat in vielen Domänen zu einer erheblichen Zunahme von Zeitreihendaten geführt. Obwohl diese Daten ein enormes Potenzial für die Gewinnung verwertbarer Erkenntnisse bergen, besteht ein häufiges Problem von Organisationen darin, das in analytischen Prozessen erzeugte Wissen systematisch zu interpretieren und wiederzuverwenden. Während unterschiedliche, auf bestimmte Aufgaben ausgerichtete Analysen auf gemeinsam genutzten Speichersystemen durchgeführt werden, entstammen die erzeugten Erkenntnisse häufig isolierten Workflows und bleiben implizit und nicht integriert. Diese voneinander unabhängige Verarbeitung begrenzt die Zugänglichkeit und Wiederverwendbarkeit. Darüber hinaus werden gewonnene Erkenntnisse oft ohne Einbeziehung szenariospezifischen Kontexts und Expertenwissens erzeugt, was ihre Interpretierbarkeit und den Grad ihrer Validierung verringert. Tatsächlich existiert nach unserem Kenntnisstand keine frei verfügbare Lösung zur Klassifikation und Verknüpfung von Expertenwissen und abgeleiteten Erkenntnissen aus Zeitreihenanalysen. Dieser Umstand reduziert Synergieeffekte und potenzielle Wissenstransfers.

Diese Arbeit befasst sich mit den Herausforderungen der systematischen Wissensintegration aus und für Zeitreihenanalysen. Sie schlägt einen umfassenden, semantisch angereicherten Wissensgraphen vor, der als zentrale, maschineninterpretierbare Einheit für die strukturierte Nutzung von mit Zeitreihendaten verbundenem Wissen dient. Konzeptionell untersucht die Arbeit Forschungsfragen dazu,
(i) wie Analyseergebnisse und Domänenwissen semantisch klassifiziert werden können,
(ii) wie dieses Wissen repräsentiert und seine Weitergabe effizient erfolgen kann und
(iii) wie das Wissen in nachfolgenden analytischen Workflows effektiv genutzt werden kann.
Das Ziel ist, Zeitreihenanalysen zu verbessern, indem eine systematische

Wiederverwendung von Erkenntnissen ermöglicht, Synergien zwischen Analysen gefördert und fundierte sowie automatisierte Entscheidungsprozesse unterstützt werden.

Zu den zentralen Beiträgen dieser Arbeit gehören:

- Hierarchisch strukturierte Vokabulare zur Klassifikation von Erkenntnissen aus Datenanalysen sowie kontextuellem Wissen in Verbindung mit Zeitreihendaten.
- Eine Ontologie zur Repräsentation der segmentbasierten Verwaltung, Klassifikation und Propagation von Zeitreihenwissen.
- Ein KI-gestützter Ansatz zur Instanziierung und Validierung von Ontologien aus Programmcode und dessen zugehörigem Laufzeitkontext.
- Eine algebraische Spezifikation zur Definition der Wissenspropagation in analytischen Pipelines.
- Eine systematische Evaluation embedding-basierter Retrievalstrategien zur Bewertung der entwickelten Ontologie in verschiedenen Retrieval-Aufgaben.
- Die Formalisierung grammatikbasierter Regeln und aussagenlogischer Formeln für die automatisierte Exploration von Zeitreihenwissen.
- Die Integration von natürlichsprachlichen Anfragen mit inkrementell wachsenden Wissensgraphen zur Ermöglichung automatisierter, semantisch angereicherter Datenanalysen.

Acknowledgments

First and foremost, I would like to thank my supervisors.

Prof. Stefan Decker, thank you for guiding me throughout this journey, and especially for reaching out and offering to support me with this topic. Without that initial step, I wouldn't be where I am today. Whenever I felt stuck, you managed to point me in the right direction, and I could always come to you for help.

Prof. Christian Beecks, working with you over many years has been both an educational and an enjoyable experience. Thank you for your continuous guidance, your support, and all the good times we've shared along the way.

Prof. Sandra Geisler, thank you for taking on the role of chair of the examination committee. Prof. Wil van der Aalst, thank you for taking the time to serve as my oral examiner.

I'm also grateful to all my colleagues from the Data Science and Artificial Intelligence department at Fraunhofer FIT and the Chair of Databases and Information Systems (i5) at RWTH Aachen University. Thanks for your support, the conversations, and the good times we've shared over the years. Especially, I would like to thank a few people who have made a difference along the way.

Leany, navigating the administrative side of work and a PhD can be quite a challenge, and I'm especially grateful to you for making it so much easier. I honestly can't remember a single time when you weren't willing to make time to help me, always with a smile.

Andrea, thank you for all your help during my early years at Fraunhofer FIT.

Reinhard and Tatiana, thank you both for your help and support before and especially during my thesis defense.

Sisay, thank you for your encouragement. Your support during those times helped a lot.

Marc, even though we haven't been in the same department for quite a few years now, I'm grateful that I could always count on you for advice or help.

Diego and Christoph, thank you both for your understanding and support throughout this journey. I believe that pursuing a PhD becomes much more difficult without group and department heads who truly understand what the process involves.

Rohit, thank you for your valuable support with SemTS and for introducing me to the field of Semantic Web research when I was just getting started.

Johannes, thank you for putting together the dissertation template, which made the process of writing and formatting this thesis a lot easier.

To everyone I had the pleasure of collaborating with on publications, as well as the students I had the opportunity to work with throughout this research, thank you for your contributions and for making this journey a rewarding one.

Finally, my deepest gratitude goes to my family.

To my parents and my brother, thank you for always being there for me. Your unwavering support has been the foundation that made all of this possible. I owe you more than words can express.

To my wife and children, this has been a long and exhausting chapter, and I know it has come at the cost of far too many precious moments together. Thank you for standing by me through it all, for your endless patience, and for your unconditional love and support. I couldn't have done this without you, and you mean everything to me.

Aachen, September 2026
Alexander Graß

> *"Having knowledge, but lacking the power to express it clearly is no better than never having any ideas at all."*
>
> *– Pericles*

Declaration of Authorship

I, Alexander Graß

declare that this thesis and the work presented in it, are my own and have been generated by me as the result of my own original research.

I do solemnly swear that:

1. This work was done wholly or mainly while in candidature for the doctoral degree at this faculty and university
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated
3. Where I have consulted the published work of others or myself, this is always clearly attributed
4. Where I have quoted from the work of others or myself, the source is always given. This thesis is entirely my own work, with the exception of such quotations
5. I have acknowledged all major sources of assistance
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself
7. Parts of this work have been published before. A detailed list can be found in Section 1.4

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, I used different models of Anthropic's[1] and OpenAI's[2] generative AI and DeepL[3] to improve the writing, make suggestions, and for rephrasing. After using these tools/services, I carefully reviewed and edited the content as needed.

[1] https://www.anthropic.com
[2] https://www.openai.com
[3] https://www.deepl.com

Contents

1 Introduction **1**

1.1 Motivation . . . 2
1.2 Concept . . . 4
1.3 Research Questions . . . 6
1.4 Publications . . . 8
1.5 Thesis Outline & Contributions . . . 14

2 Research Context and Fundamentals **16**

2.1 Time Series Analytics . . . 17
2.1.1 Exploratory Data Analysis . . . 20
2.1.2 Machine Learning . . . 21
2.1.3 Automated and Explainable Data Analysis . . . 24
2.2 Semantic Web Technologies . . . 26
2.2.1 Ontologies and Vocabularies . . . 27
2.2.2 Knowledge Graphs . . . 30
2.2.3 Knowledge Retrieval, Reasoning and Validation . . . 32

3 Time Series Knowledge **34**

3.1 Definitions of Time Series Knowledge . . . 35
3.2 General Dataset Properties . . . 42
3.3 Time Series Characteristics . . . 43
3.3.1 Descriptive Statistics . . . 44
3.3.2 Structural Profiles . . . 46
3.3.3 Instance-specific Descriptions . . . 47
3.4 Task-driven Analysis Results . . . 47
3.4.1 Inferential Statistics and Causality . . . 48

3.4.2 Event Discovery and Prediction . . . 49
3.4.3 Regressions and Forecasts . . . 50
3.4.4 (Dis-)Similarities . . . 51
3.4.5 Groupings . . . 52
3.4.6 Patterns and Associations . . . 53
3.4.7 Projections and Mappings . . . 53
3.5 Scenario Knowledge . . . 54
3.6 Taxonomy . . . 61
3.6.1 Data Knowledge Taxonomy . . . 62
3.6.2 Scenario Knowledge Taxonomy . . . 64
3.6.3 Vocabularies . . . 65
3.6.4 Evaluation . . . 67

4 SemTS - The Semantic Time Series Ontology 71
4.1 Motivation . . . 72
4.2 Related Work . . . 74
4.3 Ontology Design . . . 76
4.3.1 Requirements and Design Methodology . . . 76
4.3.2 Integration with External Ontologies . . . 78
4.3.3 Challenges and Limitations . . . 80
4.4 Ontology Description . . . 80
4.4.1 Data Management . . . 81
4.4.2 Knowledge Management . . . 83
4.4.3 Knowledge Generation . . . 84
4.4.4 Knowledge Hierarchies and Vocabularies Integration . . . 88
4.4.5 Axioms, Constraints, and Reasoning Capabilities . . . 90
4.5 Technical Specification, Documentation & Maintenance . . . 92
4.6 Validation and Evaluation . . . 94
4.7 Impact and Ongoing Adoption . . . 99

5 SemTS Knowledge Graph 101
5.1 Real-World Application Scenarios . . . 102
5.1.1 Computer Vision for Animal Welfare . . . 102

5.1.2 Forecasting of Grinding Beads Wear . . . 103
5.1.3 Anomaly Detection for Blow Molding . . . 106
5.2 Knowledge Graph Generation . . . 108
5.2.1 SemTS Ontology Population Library . . . 108
5.2.2 Code2Onto: AI-based Ontology Population . . . 113
5.3 Knowledge Retrieval . . . 127
5.3.1 Semantic Indexing . . . 128
5.3.2 Embedding-based Retrieval . . . 132
5.4 Knowledge Propagation and Contextual Reasoning . . . 151
5.4.1 Knowledge Propagation Algebra . . . 151
5.4.2 Contextual Knowledge Reasoning . . . 173

6 SemTS-based Data Analytics 182
6.1 SemTS Explorer . . . 183
6.1.1 Motivation and Related Work . . . 183
6.1.2 Approach . . . 186
6.1.3 Application Example . . . 191
6.1.4 Summary and Future Work . . . 191
6.2 Conditioned Knowledge Discovery . . . 192
6.2.1 Motivation and Related Work . . . 193
6.2.2 Approach . . . 194
6.2.3 Application Example . . . 198
6.2.4 Summary and Future Work . . . 200
6.3 LLM-based Data Science Agents . . . 201
6.3.1 Motivation and Related Work . . . 202
6.3.2 Approach . . . 203
6.3.3 Application Example . . . 205
6.3.4 Evaluation . . . 207
6.3.5 Summary and Future Work . . . 211

7 Conclusion and Future Work 213
7.1 Limitations . . . 215
7.2 Future Work . . . 217

7.3 Vision 220

Bibliography **222**

List of Figures **277**

List of Tables **283**

Listings **287**

Appendices **288**

A Categorical Overview of Scenario Knowledge **289**

B SemTS Vocabularies: Evaluation Details **292**

C SemTS Requirements **299**

D Real-World Application Scenarios: Code Listings **306**

E Knowledge Propagation Algebra: Formal Validation Sketch **327**

1 Introduction

The rapid development of modern technologies such as ubiquitous connectivity and enhanced sensor technology [YHB24; BAB24] has led to ongoing digital transformation across various application domains including healthcare, finance and industry and initiated a significant increase in data generation. By the end of 2025, experts estimated the number of connected IoT devices to be 21.1 billion[1]. Although scalable, distributed storage systems [Shu24] have been developed to resolve emerging data management challenges [Nae+21; Li+24; AAH23], these solutions inherently amplify the complexity of identifying and retaining valuable insights by allowing for larger volumes of available data. Driven by a resultant shift in research and development towards data analytics [Sar21; Jia+22], companies increasingly use data analytics to improve their strategies and adapt services. While these technologies already changed the operational dynamics of many organizations [Bha23; CKK24], new AI-based innovations like Large Language Models (LLMs) [Nav+25] further raise the degree of automation and lower the barriers for individuals with varying levels of expertise. This makes AI-driven data analytics a central component of operational workflows and decision-making processes.

Motivated by the continuously growing demand for effective and efficient data analysis approaches and the prevalence of time series data[2] in various domains and analytical contexts, in this thesis we pursue the goal of establishing a comprehensive, semantically structured knowledge graph for time series insights. This graph should facilitate a centralized resource for the retrieval, incremental utilization and propagation of semantically enriched, machine-interpretable knowledge derived from and associated with time series data. Our main goal is to improve time series analytics by examining open questions concerning the semantic

[1] https://iot-analytics.com/number-connected-iot-devices/, last update: Oct' 2025
[2] https://market.us/report/time-series-databases-software-market, last update: Sep' 2024

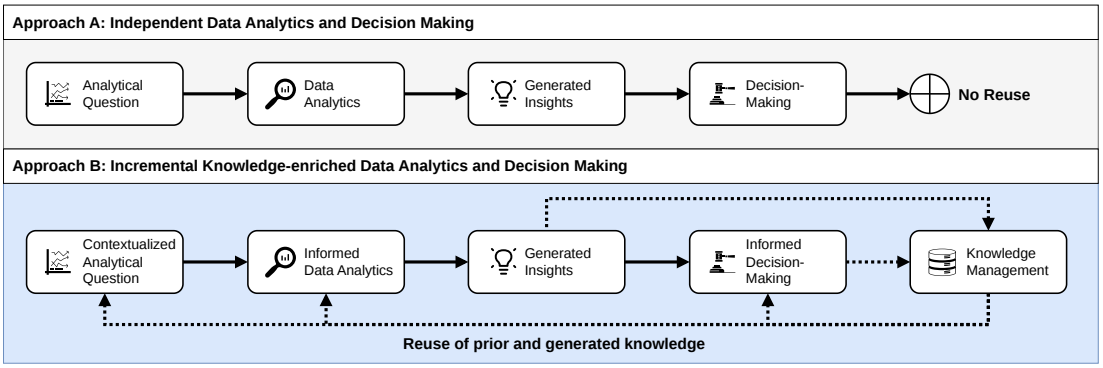


Figure 1.1: Comparison of an independent, linear analytical workflow (Approach A) against a cyclically informed workflow (Approach B).

classification, efficient management, and effective exploitation of knowledge that represents associated domain specifics and analytically inferred insights.

Chapter Outline

This chapter begins by highlighting the motivation and conceptual idea leading to this work. It identifies key research questions that aim to address topic-related challenges and opportunities. It concludes with an outline of the thesis structure and summarizes the main contributions of the study.

1.1 Motivation

The question of *how to support users in generating semantic insights from time series data* has become a crucial task across most application domains. From autonomous inspections of electrocardiograms [Han+19] and forecasts of retail sales data [MSA22] to anomaly detection in manufacturing [Sch+23], time series data constitute the input of several data analysis pipelines [Ham20]. However, as various analyses are carried out to infer valuable and exploitable insights like patterns and trends, there is a lack of structuring, persisting, and maintaining the revealed insights at the end of each data pipeline. This likely results in a missed opportunity to optimize subsequent analyses and decision-making processes via knowledge transfer. Examples include the consideration of previously identified anomalies in upcoming forecasts, or a utilization of discovered patterns for a

clustering of time series sequences. Discarding exploitable insights like identified anomalies, patterns, or cluster assignments after a single inspection consequently limits reusability and prevents the inclusion of obtained information in future analyses. While analysis artifacts usually comprise trained AI models and corresponding evaluation metrics [Zah+18], these artifacts do not account for the storage of identified insights that are placed in context. A resulting, repeated generation of insights thus equals an avoidable use of resources and time.

Another associated problem is the absence of a standardized classification for data analysis results and a priori knowledge, which impedes a semantic retrieval and exploitation of predefined or already inferred knowledge. Such a classification would define concrete categories describing, for instance, the semantic meaning of knowledge, its quality aspects, or its analytical context. Concretely assigning a meaning to initially available or derived insights through classified metadata would facilitate a categorization and interpretation within certain contexts. Without a structured approach to maintain classified insights from time series data, a utilization in informed analyses is restricted, as related models and methods are unable to harness previously identified knowledge for succeeding analytical tasks.

Especially when prior domain knowledge, such as technical process specifics, is exclusively available through human experts or stored in isolated systems disconnected from the data itself, contextual integration is mitigated. This results in an inability to exploit semantically enriched data effectively, leading to invalidated analyses and reduced interpretability. The problem can be further delegated to the user level, where organizations manage various projects and teams. A classification and reuse of knowledge could potentially compensate for different user expertise, personnel changes via knowledge preservation or enable a streamlining of common analytical subroutines. By having a structured repository for time series knowledge, companies can maintain valuable insights and expertise, allowing these resources to be utilized across multiple analytical tasks and user levels.

Figure 1.1 exemplary contrasts an independent, linear analytical workflow against a cyclically informed workflow. In a conventional setting (Approach A), an analytical question triggers an analysis, produces insights, and informs a decision.

Subsequent analyses are performed independently. In contrast, a reuse of prior and generated knowledge (Approach B) leads to *contextualized* analytical questions and *informed* analyses. Decisions are therefore grounded in a broader, semantically enriched basis. After each cycle, the resulting insights and decisions are maintained, making them available to all subsequent iterations.

1.2 Concept

The lack of structured mechanisms to systematically maintain and reuse prior and derived insights causes each subsequent analysis to start from scratch. To address the previous problems and challenges, throughout this work we discuss questions and solutions concerning the concept of an *incrementally enriched time series knowledge graph* that serves as a persistent, semantic memory for time series analytics. The knowledge graph does not replicate the underlying time series or context data stored in existing databases. Instead, it functions as a *semantic index* that maintains machine-interpretable references and classifications to assign meaning to the knowledge derived from it. Building upon Figure 1.1 depicting the comparison of linear and cyclically informed data analytics, Figure 1.2 illustrates the conceptual solution of an additional semantic layer. The semantic layer comprising the knowledge graph bridges the interpretability gap between raw data and processing, supporting different stakeholders, services and AI models by a *cyclic reuse of prior and generated knowledge*. The idea is that task-independent iterations of data analyses, result interpretations and decision-making processes contribute potentially valuable and factual knowledge, which is stored in dedicated databases, but first classified and indexed by the knowledge graph. Any addition therefore not only extends the graph but potentially increases its connectivity, allowing for further relationships between previously independent findings. With each enrichment, the graph evolves from a sparse initial index into a comprehensive knowledge base that reflects the accumulated analytical history and domain understanding and allows for informed steps along the entire data analytics lifecycle (see Figure 1.1). This benefits different key stakeholder groups, where data analysts and data scientists can build on previ-

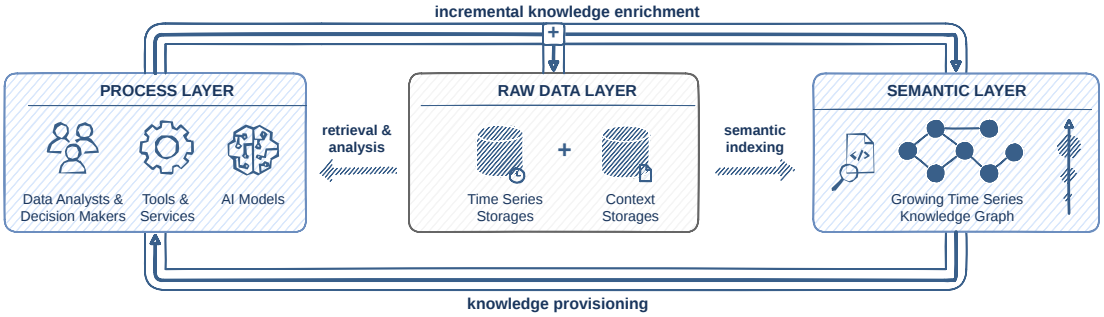


Figure 1.2: Concept of an incrementally enriched *Time Series Knowledge Graph* serving as common, standardized and evolving knowledge resource and semantic index for various consumers and processes. Gathered knowledge is iteratively used in subsequent queries and analyses.

ously classified findings to optimize, inform and interpret analyses, while decision makers gain a contextualized view of collected knowledge to ground their decisions.

1.3 Research Questions

Aligned with the design and implementation of a knowledge graph for time series knowledge, in this thesis we focus on discussing and answering the following research questions corresponding to defined requirements and objectives:

RQ 1. *In the context of time series analytics, which information can be regarded as time series knowledge and what are the associated semantic and hierarchical relationships?*

Considering the wide range of information related to time series analytics - comprising contextual and domain-specific facts beyond raw data, as well as insights derived from time series analysis methods - this question aims to define time series knowledge through semantic classification and hierarchical structuring. The main objective is to establish a standardized foundation to define time series-related knowledge.

RQ 2. *How can time series knowledge be represented and utilized in a machine-interpretable form that maximizes its accessibility, reusability and potential for deriving new insights from existing knowledge?*

This research question asks how time series knowledge as defined in RQ 1 can be captured in a formal, semantic representation and subsequently leveraged for structured access and knowledge propagation. It encompasses both the design of a representational model and the mechanisms required to make classified knowledge actionable. More specifically, it asks:

- *How can time series knowledge be semantically represented, classified, and organized such that it is machine-interpretable, traceable, and structurally suited for efficient management?*
- *How can such a semantic representation be effectively populated, retrieved, and leveraged to facilitate knowledge querying, reasoning and propagation?*

Both aspects are closely interrelated, as the representation's structure constrains how it can be utilized, while the intended usage defines how it must be structured.

RQ 3. *How can modern techniques and methods from the field of data analytics facilitate the exploitation and incremental enrichment of a time series knowledge graph?*

As innovative data analysis solutions continue to emerge at a rapid pace, this research question seeks to explore how to leverage and support these solutions through the provision and integration of a time series knowledge graph. It addresses the consolidation of domain expertise and inferred time series knowledge in informed analysis pipelines by a central knowledge graph. This involves the systematic inclusion and propagation of knowledge in the input stage and an incremental storage and semantic enrichment of newly derived insights at the end of analysis pipelines.

1.4 Publications

Parts of this dissertation are based on the previously published, peer-reviewed, core publications listed in the first section below. A second section details additional publications, which are only listed for context and reference. For all entries, contributors are listed and individual contributions are attributed. Papers still under review or in press at the time of writing are marked accordingly.

Core Publications

[GBC18] Alexander Graß, Christian Beecks, and José Ángel Carvajal Soto. "Unsupervised anomaly detection in production lines". In: *Machine Learning for Cyber Physical Systems: Selected papers from the International Conference ML4CPS 2018*. AG developed the approach, JAC and CB reviewed and revised the publication. Springer. 2018, pp. 18–25.

[GBD24] Alexander Graß, Christian Beecks, and Stefan Decker. "Structural and Semantic Data Layers in Time Series Analyses". In: *International Conference on Intelligent Data Engineering and Automated Learning*. AG drafted the publication, CB helped to refine the conceptual idea, SD reviewed and revised the publication. Springer. 2024, pp. 505–511.

[GDB22] Alexander Graß, Till Döhmen, and Christian Beecks. "Sample-based Kernel Structure Learning with Deep Neural Networks for Automated Structure Discovery". In: *2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW)*. AG developed the approach, TD and CB reviewed and revised the publication. IEEE. 2022, pp. 79–83.

[Gra+23] Alexander Graß, Christian Beecks, Sisay Adugna Chala, Christoph Lange, and Stefan Decker. "A Knowledge Graph for Query-Induced Analyses of Hierarchically Structured Time Series Information". In: *European Conference on Advances in Databases and Information*

Systems. AG developed the concept, CB and SD guided the development, SC and CL reviewed and revised the publication. Springer. 2023, pp. 174–184.

[Gra+25a] Alexander Graß, Rohit A. Deshmukh, Christian Beecks, and Stefan Decker. "Towards an Ontology for Representing Time Series Knowledge: Motivation, Requirements and Concept". In: *Intelligent Information Systems. CAiSE 2025*. Ed. by Luise Pufahl, Kristina Rosenthal, Sergio España, and Selmin Nurcan. SD initiated the idea, AG and RD developed the ontology, CB reviewed and revised the publication. Cham: Springer Nature Switzerland, 2025, pp. 103–110.

[Gra+25b] Alexander Graß, Jonathan Lehmkuhl, Diego Collarana, Stefan Decker, and Christian Beecks. "Code2Onto: Multi-Agent System for Code-Driven Ontology Population". In: *2025 IEEE International Conference on Big Data (BigData)*. AG developed the approach, JL helped with literature review, SD optimized the initial idea, DC and CB reviewed and revised the publication. 2025, pp. 5642–5651.

[Gra+25c] Alexander Graß, Christopher I Pack, Diego Collarana, Stefan Decker, and Christian Beecks. "Semantic Intelligence: Graph RAG-Driven Agents for Time Series Analytics". In: *International Conference on Intelligent Data Engineering and Automated Learning*. AG developed the knowledge integration and CP developed the agentic approach, DC, CB and SD reviewed and revised the publication. Springer. 2025, pp. 226–231.

[Gra+26] Alexander Graß, Rohit A Deshmukh, Christoph Lange, Diego Collarana, Christian Beecks, and Stefan Decker. "SemTS: Ontology and Vocabularies for the Semantic Categorization of Time Series Knowledge". In: *European Semantic Web Conference*. SD initiated the idea, AG and RD developed the ontology, CL provided support with technical details, DC and CB reviewed and revised the publication. Springer. 2026, pp. 172–190.

Additional Publications

[Alb+24] Enrico Alberti, Sergio Álvarez-Napagao, Víctor Anaya, Marta Barroso, Cristian Barrué, Christian Beecks, Letizia Bergamasco, Sisay Adugna Chala, Víctor Giménez-Ábalos, Alexander Graß, Daniel Hinjos, Maike Holtkemper, Natalia Jakubiak, Alexandros Nizamis, Edoardo Pristeri, Miquel Sànchez-Marrè, Georg Stefan Schlake, Jona Scholz, Gabriele Scivoletto, and Stefan Walter. "AI Lifecycle Zero-Touch Orchestration within the Edge-to-Cloud Continuum for Industry 5.0". In: *Systems* 12.2 (2024). Publication in the scope of the knowlEdge EU project (under grant agreement 957331). All authors contributed with their individual expertise to the corresponding sections., p. 48.

[Alv+21] Sergio Alvarez-Napagao, Boki Ashmore, Marta Barroso, Cristian Barrué, Christian Beecks, Fabian Berns, Ilaria Bosi, Sisay Adugna Chala, Nicola Ciulli, Marta Garcia-Gasulla, Alexander Grass, Dimosthenis Ioannidis, Natalia Jakubiak, Karl Köpke, Ville Lämsä, Pedro Megias, Alexandros Nizamis, Claudio Pastrone, Rosaria Rossini, Miquel Sànchez-Marrè, and Luca Ziliotti. "knowlEdge Project – Concept, Methodology and Innovations for Artificial Intelligence in Industry 4.0". In: *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*. SAN orchestrated the publication in the scope of the knowlEdge EU project (under grant agreement 957331). All authors contributed with their individual expertise to the corresponding sections. 2021, pp. 1–7.

[Ara+19] Gustavo Aragón, Harsh Puri, Alexander Grass, Sisay Chala, and Christian Beecks. "Incremental deep-learning for continuous load prediction in energy management systems". In: *2019 IEEE Milan PowerTech*. HP developed and implemented the discussed approaches, GA drafted the publication and supported with domain expertise, AG and SC supported with technical expertise, CB reviewed and revised the publication. IEEE. 2019, pp. 1–6.

[Bac+25] Nicolas Bach, Christian Beecks, Gesa Benndorf, Julia Burr, Franziska Dechert, Frederik Diederichs, Hartmut Eigenbrod, Georg Fuchs, Alexander Graß, Varun Gowtham, Jan Hamaekers, Dirk Hecker, Sonja Holl-Supra, Maximilian Hörstrup, Bjarne Huuck, Wilhelm Jaspers, Wolfgang Kratsch, Christopher Krauss, Jonathan Krauß, Christian Kühnert, André Kummerow, Matthis Leicht, Gerhard Paass, Angelika Voß, Pascal Nicolai Linden, Miriam Mock, Tim Wirtz, Daniel David Scherer, Christian Ufrecht, Maximilian Röglinger, Gyunam Park, Wil van der Aalst, Niklas Uhl, Florian Rippstein, Jonathan Ott, Christopher Mutschler, Maximilian Stahlke, Alper Yaman, Christian Seidler, Julian Raible, Mathias Ziebarth, Andreas Wilde, Ina Steinmetz, Janko Petereit, Alexander Zorn, Christian Pionzewski, Andrej Plenne, Florian Strohm, and Felix Ott. *Jenseits von Text und Bild - Generative KI für vielfältige Datenwelten: Praxisnahe Einblicke und innovative Anwendungsbeispiele zur Erschließung komplexer Unternehmensdaten*. Ed. by Fraunhofer-Allianz Big Data und Künstliche Intelligenz. Publication in the scope of the Fraunhofer-Allianz Big Data und Künstliche Intelligenz. All authors contributed with their individual expertise to the corresponding sections. Fraunhofer Verlag, 2025. 64 pp. ISBN: 978-3-8396-2133-2.

[Bee+19] Christian Beecks, Kjeld Willy Schmidt, Fabian Berns, and Alexander Graß. "Gaussian Processes for Anomaly Description in Production Environments". In: *Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference*. Ed. by Paolo Papotti. Vol. 2322. CEUR Workshop Proceedings. CB drafted the publication, KWS and FB supported the implementation process. AG reviewed and revised the publication. 2019.

[Bee+24a] Christian Beecks, Anandraj Amalraj, Alexander Graß, Marc Jentsch, Felix Kitschke, Maximilian Norz, and Patric Schäffer. "Leveraging YOLO for Real-Time Video Analysis of Animal Welfare in Pig Slaughtering Processes". In: *German Conference*

on Artificial Intelligence (Künstliche Intelligenz). CB, AG and AA took care of the methodology and the implementation. FK supported the implementation, MJ and MN reviewed and revised the publication, PS supported with domain expertise. Springer. 2024, pp. 275–281.

[Bee+24b] Christian Beecks, Alexander Graß, Anandraj Amalraj, Marc Jentsch, Felix Kitschke, Maximilian Norz, and Patric Schäffer. "Entwicklung einer KI für automatisierte Tierschutzkontrollen in der Schweineschlachtung". In: *INFORMATIK 2024*. Ed. by Maike Klein, Daniel Krupka, Cornelia Winter, Martin Gergeleit, and Ludger Martin. Lecture Notes in Informatics (LNI) P-352. CB, AG and AA took care of the methodology and the implementation. FK supported the implementation, MJ and MN reviewed and revised the publication, PS supported with domain expertise. Bonn: Gesellschaft für Informatik e.V., 2024, pp. 1463–1477.

[Ber+19] Fabian Berns, Kjeld Schmidt, Alexander Grass, and Christian Beecks. "A new approach for efficient structure discovery in IoT". In: *2019 IEEE International Conference on Big Data (Big Data)*. FB drafted the publication, KS supported the implementation process. AG and CB reviewed and revised the publication. IEEE. 2019, pp. 4152–4156.

[BG16] Christian Beecks and Alexander Graß. "Multi-step threshold algorithm for efficient feature-based query processing in large-scale multimedia databases". In: *2016 IEEE International Conference on Big Data (Big Data)*. CB drafted the publication, AG supported with the development and implementation of the discussed approach. IEEE. 2016, pp. 596–605.

[BGD18] Christian Beecks, Alexander Grass, and Shreekantha Devasya. "Metric indexing for efficient data access in the internet of things". In: *2018 IEEE International Conference on Big Data (Big Data)*. CB drafted the publication and conducted experiments. AG supported

with technical expertise. AG and SD reviewed and revised the publication. IEEE. 2018, pp. 5132–5136.

[CG23] Sisay Adugna Chala and Alexander Graß. "Human-AI Interaction for Semantic Knowledge Enrichment of AI Model Output". In: *Artificial Intelligence in Manufacturing: Enabling Intelligent, Flexible and Cost-Effective Production Through AI*. SC drafted the publication. AG reviewed and revised the publication. Springer Nature Switzerland Cham, 2023, pp. 43–54.

[Col+25] Diego Collarana, Christopher Ingo Pack, Yan-Ying Liao, Marlena Flüh, Jonathan Lehmkuhl, Abhishek Nadgeri, Alexander Graß, Moritz Busch, Prinon Das, Lara Dingels, Stefan Decker, and Christian Beecks. "Graph RAG in the Wild: Insights and Best Practices from Real-World Applications". In: *Semantic Web – Interoperability, Usability, Applicability* (2025). Under review (minor revision); All authors contributed to their corresponding application listed in the publication.

[Kar+23] Md Rezaul Karim, Md Shajalal, Alexander Graß, Till Döhmen, Sisay Adugna Chala, Alexander Boden, Christian Beecks, and Stefan Decker. "Interpreting black-box machine learning models for high dimensional datasets". In: *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*. RK developed the approach and drafted the publication, the remaining authors reviewed and revised the publication. IEEE. 2023, pp. 1–10.

[Sch+23] Jona Scholz, Maike Holtkemper, Alexander Graß, and Christian Beecks. "Anomaly Detection in Manufacturing". In: *Artificial Intelligence in Manufacturing: Enabling Intelligent, Flexible and Cost-Effective Production Through AI*. JS drafted the publication. JS and AG developed and implemented the presented methodology, MH supported with writing the technical sections. CB supported with writing the general sections and reviewed and revised the publication. Springer Nature Switzerland Cham, 2023, pp. 351–360.

1.5 Thesis Outline & Contributions

To answer the research questions formulated in section 1.3, this thesis is structured as follows:

Chapter 2 outlines and categorizes the research context to provide the necessary fundamentals and to position our research within the current academic landscape.

Chapter 3 addresses RQ 1 by defining time series knowledge and discussing the different categories of analysis results and associated contextual knowledge in the scope of time series analytics. This chapter further contributes to the definition of a common taxonomy for the hierarchical classification and description of time series knowledge.

Chapter 4 primarily focuses on answering RQ 2. As a main contribution of this thesis, we present an ontology, entitled *The Semantic Time Series Ontology (SemTS)*, to semantically specify time series meta-information, associated knowledge entities and corresponding knowledge propagation processes.

Chapter 5 complements the answer to RQ 2 by further addressing how specific time series knowledge defined through SemTS can be utilized to generate one common knowledge graph that serves as a repository for exploratory data analysis, machine learning and other AI-based technologies. This chapter includes approaches to an instantiation of concrete knowledge entities as well as their retrieval, propagation and validation.

Chapter 6 is concerned with a solution to RQ 3 by addressing the exploitation of structured knowledge in modern data analytics. Aligned with a description of potential and developed applications, we primarily discuss the utilization of a time series knowledge graph in automated data science pipelines and agentic solutions.

Chapter 7 concludes this thesis by summarizing our contributions, addressing their limitations and suggesting directions for future research.

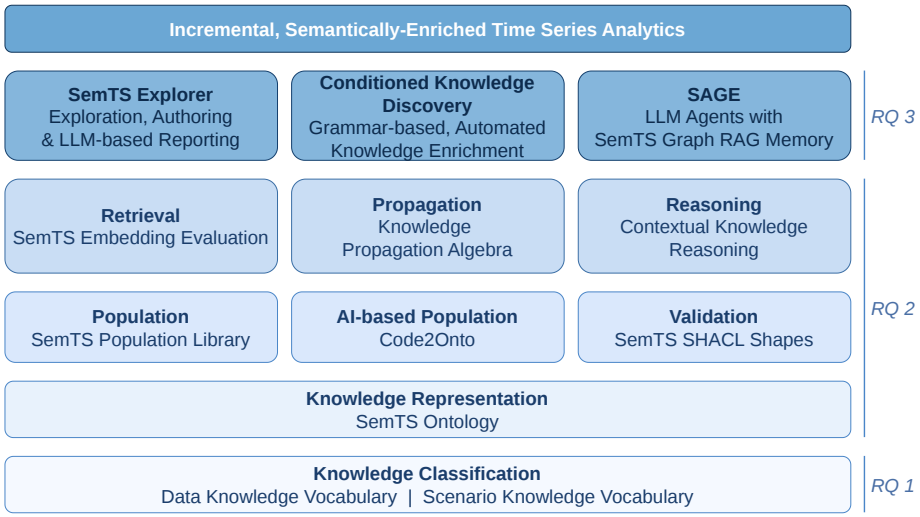


Figure 1.3: Layered overview of the thesis contributions, organized by increasing abstraction from knowledge classification at the base to application-level exploitation at the top. Each layer is annotated with the corresponding research question it primarily addresses.

Apart from the mapping of contributions to individual chapters presented above, Figure 1.3 additionally illustrates how the research questions align with the listed contributions.

Chapter Summary

This chapter motivated the need for a semantically structured knowledge graph that transforms isolated, linear analytical workflows into cyclically informed processes. We identified three research questions spanning the classification (RQ 1), management (RQ 2), and exploitation (RQ 3) of time series knowledge and outlined the conceptual solution of a semantic layer that incrementally bridges raw data and analytical processes.

What follows. Before addressing these research questions, Chapter 2 establishes the necessary technical foundations in time series analytics and Semantic Web technologies that underpin the contributions of this thesis.

2 Research Context and Fundamentals

The ability to analyze and interpret time series data often contains the identification of critical patterns and trends that drive informed decision-making [Sar21]. Acquiring a foundational understanding of the concepts and complexity of modern data-driven challenges [HAN21] is essential for effectively discussing the topics within this work and the questions of how data insights are represented, generated and exploited. To navigate the landscape of related methods (cf. Section 3), it is beneficial to comprehend the core analytical paradigms that describe the discovery and inference of meaningful insights from time series data. These paradigms can be classified to emphasize individual methodologies like automated [Shc+23; Als+22] and informed [Kar+21; MN23; Von+23] data analytics.

Since a major aspect of this work is the semantic classification of time series knowledge, another important topic to elaborate is Semantic Web technologies [BHL23] including ontologies and vocabularies. Additionally, this encompasses other, related technologies from the field of Semantic Web [HKR09; PJ21], such as knowledge graphs [Hog+21; Pen+23] and methods for retrieving, reasoning, and validating relevant information.

This chapter introduces time series analytics and Semantic Web technologies as the two foundations this work builds upon.

Chapter Outline

This chapter presents relevant contexts and serves as a reference throughout this thesis. It covers necessary topics such as the definition of time series data, the analytics involved in processing this data type and the integration of Semantic Web

technologies to represent and manage associated time series insights. The chapter begins with an overview on time series analytics (2.1), highlighting key methodologies such as exploratory data analysis, Machine Learning and approaches for automated and explainable data analysis. Afterwards, the chapter introduces Semantic Web technologies (2.2), focusing on their role in systematically classifying and representing knowledge.

This includes discussions on vocabularies and ontologies, as well as knowledge graphs and enhancing technologies. Each section provides an overview of related topics, illustrating their relevance and role in the research context.

2.1 Time Series Analytics

Time series are ordered sequences of uni- or multivariate data points collected at successive time steps. A univariate time series consists of data points defining instances of a single variable over time, while a multivariate time series involves multiple variables observed simultaneously. Concrete examples of time series include sensor readings that capture environmental or physical phenomena, audio and video data comprising temporally ordered sequences of sound waves and images respectively, or trajectories tracking spatial movement. Similar to other data types, the value of time series data lies primarily in the latent insights they provide in different application contexts [Ham20]. Such contexts for instance comprise financial applications, where time series data is used to track and forecast stock prices [STK22], or healthcare domains where vital signs are monitored and classified based on blood pressure and heart rate [Nwi+23]. The key characteristic of time series, which sets them apart from other types of data, is temporal dependency. Corresponding records are not necessarily independent over time and thus can be inferred from past values due to the inherent sequential nature [Box+15]. Therefore, time series data may additionally show local trends, long-term movements or comprise regular patterns like seasonalities and cycles. Identifying the aforementioned time series characteristics or uncovering other potentially significant insights can be a challenging task. Beyond temporal dependency, time series data is often characterized by significant noise, especially in real-world scenarios

where factors such as measurement errors or uncertainty can impact data quality [WW20]. One example is sensor data collected in industrial environments, which may be noisy due to external factors or equipment limitations. Another important challenge with time series data is their diverse semantics, which complicates the generation of universal representations. Unlike textual data, where words typically maintain consistent meanings across contexts, time series subsequences can represent entirely different phenomena in other scenarios.

Concerning the formal definition of a time series, we specify a time series as follows:

Definition 1. *Time Series*
Given an ordered sequence of T observations indexed by time t, a multivariate time series is defined as

$$\{\mathbf{x}_t\}_{t=1}^{T} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}, \quad \mathbf{x}_t \in \mathcal{D}^n,$$

where $\mathbf{x}_t$ is an n-dimensional vector of time-dependent observations at time t. In case of $n = 1$, the time series is called univariate.

Although it generally is the case that $\mathcal{D} \equiv \mathbb{R}$ or $\mathcal{D} \equiv \mathbb{N}$, depending on whether values are continuous or discrete, data points are not strictly required to be of these types and can belong to other types or structures based on context and application. However, as many analysis models only process numeric values, other data types are commonly transformed or encoded into numeric representations. A related aspect, required for interpreting context and scale is the concrete unit associated with each dimension.

An important property of time series, particularly of interest in the scope of time series analytics, is the possibility to represent time series as partitions of coherent subsequences with potentially lower dimensionality, known as data segments. Analyzing data subspaces is well-known from Online Analytical Processing (OLAP) [CD97] applications, where relational data is organized in multidimensional cube structures and slices corresponding to segments refer to specific analyzable subsections. For time series, segments may represent notable areas such as important events, unique signal behaviors, or other meaningful charac-

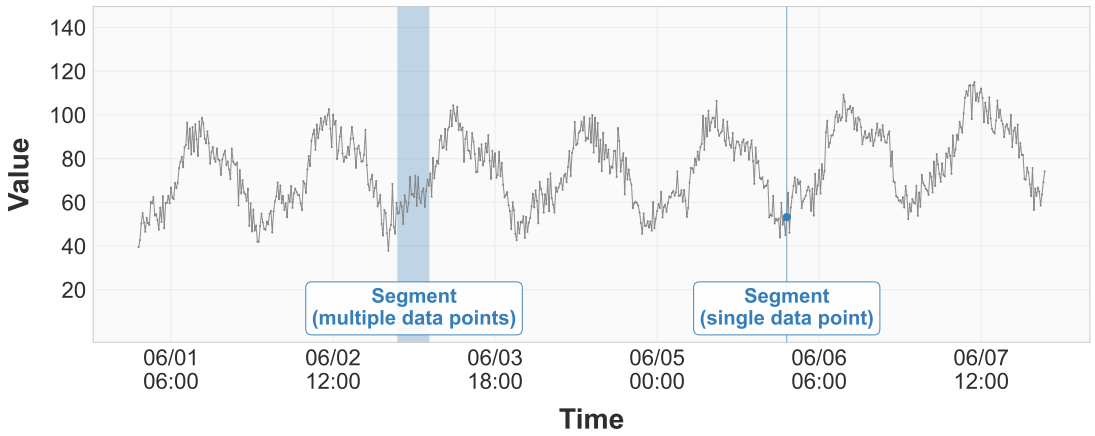


Figure 2.1: Illustration of a univariate time series and an exemplary representation of two segments defining a time range and a single point in time. In case of multivariate time series data, a segment can also define a (sub-)space consisting of more than one variable.

teristics. Segmentation [LMS14; SEL21], a specialized field within time series analysis, is dedicated to identify these segments in a time series by differentiating between subsequences with varying characteristics and properties. By focusing on segments, analysts are able to gain deeper insights into the temporal structure of the data and identify patterns or changes that may not be evident in the entire series. In alignment with Definition 1, a segment of a time series can be formalized as:

Definition 2. *Time Series Segment*
A segment over an interval $[t_s, t_e]$ *of a time series is defined as*

$$\{\mathbf{x}_t\}_{t=t_s}^{t_e} = \{\mathbf{x}_{t_s}, \mathbf{x}_{t_s+1}, \ldots, \mathbf{x}_{t_e}\}, \quad 1 \leq t_s \leq t_e \leq T, \quad \mathbf{x}_t \in \mathcal{D}^m.$$

where $[t_s, t_e]$ *specifies the index interval w.r.t. time and* m *indicates a potential reduction in dimensionality w.r.t. Definition 1 with* $m \leq n$.

Figure 2.1 illustrates both the definition of a time series and a segment. In this example, a univariate time series is depicted, highlighting both a segment that spans multiple adjacent data points and another segment that only consists of a single data point. By definition, the entire time series itself is also considered a

segment.

2.1.1 Exploratory Data Analysis

Although unprocessed data points in a time series represent the most uncompressed form of information, they do not automatically convey actionable insights, i.e. insights usable for decision-making processes. The primary reason is that many data insights including anomalies, patterns and local trends are inherent and not immediately obvious in large data collections or lack contextualization. A common procedure in various process models [SKG21; Fir23] and data engineering approaches [RH22] is to initially comprehend the data by converting raw time series information into measurable properties with concrete semantics [Bar+20], often called features. The reason is twofold: On the one hand, these features potentially allow to classify and interpret time series data based on characteristics. On the other hand they can serve as inputs for further analyses to uncover more intricate relationships within or between time series data [Ful18]. In both cases, the significance of a feature is determined by how well it explains the structure of the time series and contributes to the intended target of the analysis [Chr+18]. Beyond comparatively simple statistical descriptors such as mean or variance and series-specific measures like auto-regression or linear slope, features might further correspond to latent, more complex knowledge generated by conducted data analyses. The establishment of a taxonomy for time series knowledge classifying and relating various types of knowledge, is a contribution of this work and is discussed in Section 3. Two prominent subfields within the realm of data science that focus on extracting higher-level knowledge from data are Data Mining [EA12; Aga13; FV17] and Knowledge Discovery [LKK01; CI21]. Often used interchangeably, these subfields aim at exploring and analyzing data sets to discover characteristic groups and other types of relationships. While not restricted to unsupervised methods, which perform analyses without depending on pre-existing labels or outcomes, approaches mainly intersect and encompass algorithmic concepts such as clustering [ASW15; Alq+21], pattern recognition [Lin+12a; Spi+11] and association rules [Das+98; Sho+15]. In the scope of time series analysis, examples of specific applications comprise a clustering of individual seg-

ments [Zha+19] or discovering motifs and discords [Lu+22]. However, the vast amount of potentially available knowledge can make it challenging to identify which information is truly valuable and actionable. Furthermore, the selection of knowledge is difficult as it depends on the specific analytical scenario and the choice of method, requiring a careful consideration to ensure the right information is used for effective analysis. Feature selection techniques [CK10] and automated knowledge discovery approaches like Deep Feature Synthesis (DFS) [KV15] address this issue. Still, the automated discovery of knowledge remains a difficult task due to data inconsistencies and data complexity, making it complicated to discern valuable insights [Al-21]. To address this problem, representation learning [Eld+21; Men+23; FDJ19] has emerged as an alternative approach, allowing models to learn important characteristics directly from data, where Machine Learning techniques including deep learning [Gam17; Zer+21], are used to automatically discover general representations. It further reduces the dependency on domain-specific expertise and increases the generalizability of models across different tasks and datasets. Task-specific models (cf. Section 3.4) adapt to specific analytical scenarios by inherently extracting relevant features without explicit pre-selection, which enhances flexibility and accuracy.

2.1.2 Machine Learning

Machine Learning (ML) [Nie19] is a category of data-driven analysis approaches, which automatically extract insights and patterns from raw data, reducing the need for manual feature engineering typically required in traditional analytical methods [Box+15].

Machine Learning models can be categorized into various types (cf. Section 3) based on their learning paradigms. At a higher abstraction level, these models are distinguished based on their objectives and how they interact with data.

One primary category of distinction among Machine Learning models is the separation of supervised and unsupervised approaches. In the previous context of data exploration, we already mentioned the relationship between knowledge discovery and unsupervised learning. The reason is that unsupervised Machine Learning models [Liu+23; LKL14] are particularly adept at identifying characteristic

structures within data without requiring labeled outcomes as ground truth representatives. Techniques such as clustering [ASW15; Alq+21], dimensionality reduction [Ash+23] and anomaly detection [Mej+24] enable the exploration of datasets to facilitate a deeper understanding of the data, potentially leading to actionable insights. Formally, we define unsupervised learning as:

Definition 3. *Unsupervised learning*
Let X be a random variable on $\mathcal{X}$ with distribution P_X, and let $D = \{x_i\}_{i=1}^n$ be an i.i.d. sample from P_X. Given a task-specific hypothesis class $\mathcal{H}$ and an objective J that depends only on the inputs, unsupervised learning corresponds to learning:

$$h^\star \in \arg\min_{h \in \mathcal{H}} J(h; D).$$

such that the goal is to find a solution h which minimizes the objective J. Without a loss of generality, maximization objectives can be written as a minimization of $-J$.

In comparison, supervised learning models involve a training on labeled data. This enables the model to learn a mapping from inputs to outputs (targets) based on concrete examples, making it suitable for tasks like classification [Ism+19; Fao22] and regression [Moh+24; JPJ24].

Definition 4. *Supervised learning*
Let (X, Y) be random variables on $\mathcal{X} \times \mathcal{Y}$ with joint distribution P_{XY}, and let $D = \{(x_i, y_i)\}_{i=1}^n$ be an i.i.d. sample from P_{XY}. Given a hypothesis class $\mathcal{F} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ and an objective J that both depends on inputs and targets, supervised learning corresponds to learning:

$$f^\star \in \arg\min_{f \in \mathcal{F}} J(f; D)\,.$$

Learning $f^\star$ is typically achieved by estimating the conditional probability $P_{Y|X}$ using labeled examples. However, as we will observe in another forthcoming distinction, supervised learning also includes models that approximate the joint probability P_{XY} of input X and output Y.

Further types which complement and combine the learning paradigms of supervised and unsupervised learning are reinforcement learning [FWB22] and semi-supervised learning [VH20]. While reinforcement learning focuses on achieving specific goals through environmental interaction and reward systems, semi-supervised learning enhances model performance by utilizing both labeled and unlabeled data to enhance learning capabilities.

Another layer of distinction is between models that focus on understanding and predicting the relationships within data (discriminative) [Bag+17] and those that seek to comprehend and replicate inherent data characteristics (generative) [OE18]. Models used for discriminative learning, are concerned with drawing distinctions and making decisions based on the input data by approximating the conditional probability. The goal is to learn decision boundaries or direct mappings from inputs to outputs. Thus this type of learning is typically, but not exclusively associated with supervised learning and can be regarded a subset.

Definition 5. *Discriminative Learning*
Let (X, Y) *be random variables over* $\mathcal{X} \times \mathcal{Y}$ *with joint distribution* P_{XY} *and let* $D = \{(x_i, y_i)\}_{i=1}^n$ *be an i.i.d. sample from* P_{XY}*. Discriminative learning models the conditional distribution* $P_{Y|X}$ *directly:*

$$P_{Y|X}(y \mid x) = \frac{P_{XY}(x, y)}{P_X(x)},$$

Unlike models used for discriminative learning, models used for generative learning approximate the joint or marginal distribution. Nonetheless, generative models like the Naive Bayes classifier [PKK24], can also be used for discriminative prediction and by design support supervised and unsupervised learning, depending on the task they are intended to perform. However, for many tasks utilizing generative models, a combination like for semi-supervised learning is used.

Definition 6. *Generative Learning*
Let X *be random variables over* $\mathcal{X}$ *and optionally* Y *over* $\mathcal{Y}$*. When* Y *is observed,* (X, Y) *has a joint distribution* P_{XY}*. Let* $D = \{(x_i, y_i)\}_{i=1}^n$ *be i.i.d. samples from*

P_{XY}. Generative learning models:

$$P_{XY}(x, y) = P_{X|Y}(x \mid y)\, P_Y(y).$$

If Y is not observed, let $D = \{x_i\}_{i=1}^n$ be i.i.d. samples from the marginal distribution P_X and generative learning models

$$P_X(x).$$

Knowing about the various categories of analysis methods is crucial since it provides an overview of the knowledge that can be obtained from each category and how corresponding models might reuse and propagate existing information. As an example, Figure 2.2 illustrates the difference and similarity between unsupervised and supervised anomaly detection for time series data. Unlike the unsupervised case, supervised anomaly detection utilizes labeled data to identify anomalies similar to those learned during training. Thus, approaches of both algorithmic paradigms detect anomalies as uncharacteristic time series segments, but with differences in the amount and type of available information provided as input.

While key challenges in utilizing Machine Learning models for time series data include capturing complex dependencies and managing large volumes of potentially high-dimensional data, the complexity of corresponding models usually increases to the same extent [VF05]. This results in models that depend heavily on effective optimization, resembling black-box systems [Kar+23] that generate intractable knowledge, thereby enabling the identification of correlations without respecting interpretability. The following section will briefly discuss solutions to these problems that facilitate the automation and explainability of Machine Learning.

2.1.3 Automated and Explainable Data Analysis

The rapid advancement of AI-based approaches continues to lead to a large variety of models that are additionally subject to the tradeoff between enhanced accuracy and increased structural depth [GNT03]. This not only impedes optimal model

selection, but also results in a lack of transparency with regard to interpretability and explainability.

As a response to the high amount of available Machine Learning models, Automated Machine Learning (AutoML) has emerged to enable the automation of tasks such as feature engineering [CMS24; HPR19], architectural search and hyperparameter optimization for improved accessibility and efficiency. In the context of time series data this specifically involves automating intelligent model selection and optimization for tasks like forecasting [Wu+24] or anomaly detection [Yin+20].

However, while AutoML provides convenience, it does not prevent from a selection of "black box" models, where the internal workings are opaque and difficult to interpret. This opacity has led to the growing demand for Explainable AI (XAI) [Roj+21], which aims to elucidate decision-making processes to enhance the trustworthiness of these models. XAI techniques, such as Shapley Additive Explanations (SHAP) [LL17] and Local Interpretable Model-agnostic Explanations (LIME) [RSG16], help to highlight feature importance and to interpret complex model behaviors. A more validated alternative to improve explainability is Informed Machine Learning (IML) [MN23; Rue+22]. Instead of analyzing the model after processing, this methodology integrates validated, domain-specific information directly into the model, enhancing both its reasoning and explainability. In the case of time series data IML can help to incorporate expert insights about periodic patterns or causal relationships to yield semantically-enriched predictions. By ensuring models are both data-driven and context-aware, informed ML can generate more reliable and interpretable outcomes.

The most recent innovation, driven by the rise of generative AI [Man+24], especially Large Language Models (LLMs) [Zha+23b], is LLM-based data analytics [Nas+24]. Multiple solutions exist to utilize LLMs or related technologies for time series analysis, including adaptations of architectures like transformers [Wen+23] or pretrained foundation models [Lia+24]. Moreover, due to advancements in reasoning and code generation capabilities of LLMs [Jia+24b], modern approaches utilize LLMs directly to select and apply suitable data analysis methods for specified tasks defined in natural language. Related technologies,

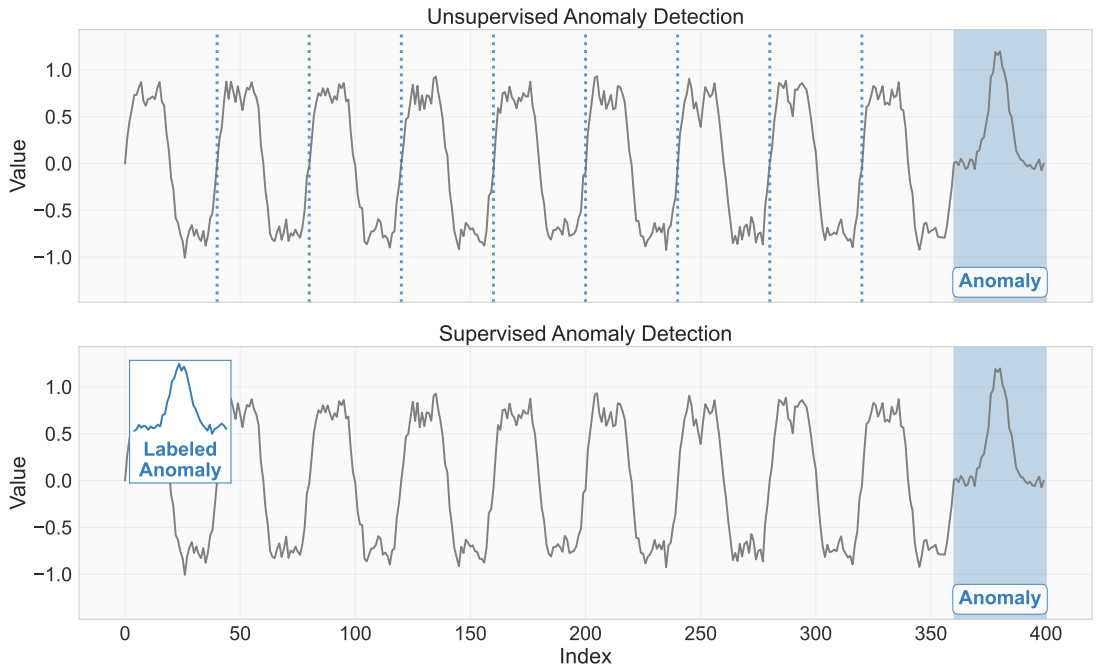


Figure 2.2: Unsupervised vs. supervised anomaly detection: In the unsupervised setting, anomalies are detected using only unlabeled data; for example, by comparing individual segments as indicated by dotted lines. In the supervised setting, labeled examples of anomalies are available, turning anomaly detection into a classification problem by learning to distinguish normal from anomalous segments.

like Retrieval Augmented Generation (RAG) [Lew+20], enable the integration of supplementary information from contextual data, with graph-based variants (Graph RAG) [Han+24a] specifically leveraging graph structures. This approach inherently combines the benefits of previously outlined paradigms, such as AutoML and IML. Consequently, it allows LLM-based data analytics to handle complex data exploration and analysis intuitively, making it accessible to users without deep technical expertise.

2.2 Semantic Web Technologies

The previous section established the basis for examining discussions around RQ 1 and RQ 3 by outlining the topic of time series analytics. The following overview on

Semantic Web technologies supports to address RQ 2, explaining how these technologies can be applied for representing, contextualizing and validating knowledge related to time series.

Semantic Web technologies [HKR09] offer a way to semantically organize and link data, allowing machines to access and interpret the meaning of information. The benefits of Semantic Web technologies can generally be summarized by the FAIR data principles (Findable, Accessible, Interoperable, and Reusable) [Wil+16], ensuring that data is semantically structured, easily discoverable and can be effectively shared and reused across various systems and domains. At the core of Semantic Web technologies are schemas [HK87], which include ontologies [Din+07] and vocabularies providing structured, formal definitions of concepts and their relationships within specific contexts. These schemas are complemented by knowledge graphs [Hog+21] and corresponding embeddings [Cao+24] that represent the concrete data structure behind semantic concepts, allowing to implement and manage complex relations. Technologies extending the previous structures play an important role by validating knowledge graphs [KK17] against established schemas or to enable precise querying and retrieval of information [AP11] by leveraging the defined relationships and concepts, ensuring that data can be efficiently accessed and utilized for various analytical purposes. Concrete frameworks relying on Semantic Web technologies to facilitate interoperability across applications and domains, are known as Data Spaces [Bac+24].

In the upcoming subsections, we will investigate each topic in more detail, starting with ontologies and vocabularies.

2.2.1 Ontologies and Vocabularies

Ontologies and vocabularies are fundamental components of Semantic Web technologies that enable structured, interoperable and machine-readable metadata. Ontologies are formal, expressive frameworks that define concepts, relationships and rules within a domain, enabling reasoning and inference, whereas vocabularies are sets of standardized terms that provide shared meanings for data elements and focus on semantic consistency. Both, ontologies and vocabularies rely on a layered stack of technologies (see Figure 2.3) that specify the structural and

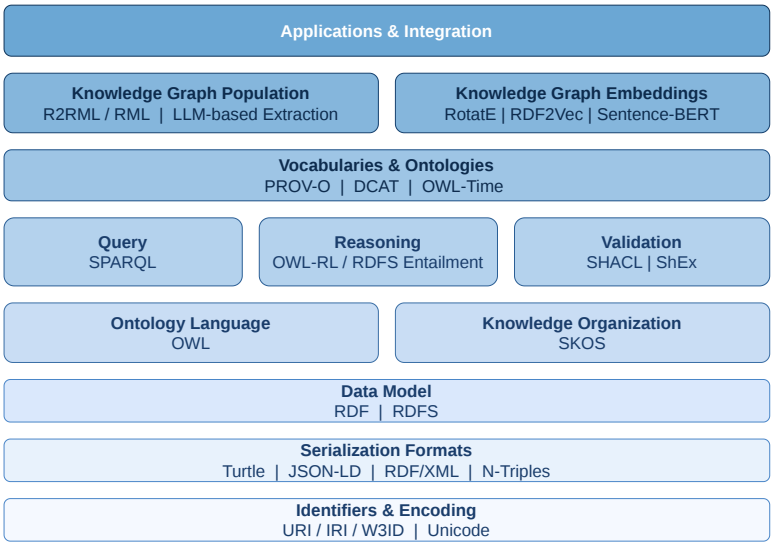


Figure 2.3: An adapted representation of the Semantic Web technology stack, emphasizing technologies and standards employed in this work including concrete examples. Layers are ordered by increasing abstraction, ranging from identifiers and serialization formats at the base to application-level integration at the top.

semantic capabilities necessary for their design and implementation. To implement the structured exchange of information, the Extensible Markup Language (XML) [Bra+06] and the XML Schema Definition (XSD) [Tho+04] offer basic functionalities for encoding and validating data in a standardized format. XML provides both human-readable and machine-readable ways to structure data, facilitating interoperability across systems. XSD complements XML by defining the rules and structure for XML documents, ensuring data integrity and compliance. Building on these structures, the Resource Description Framework (RDF) [Mil98] enables a graph-based model for representing data with semantic capabilities. RDF employs triples consisting of subject, predicate and object, to encode relationships between entities, offering a flexible and extensible structure for linking data. Moreover, RDF Schema (RDFS) [BG14] extends this framework by providing mechanisms to define classes and properties, establishing hierarchies and constraints that increase the expressiveness of RDF data. One popular solution to complement a standardized data representation by rich semantics is the Web Ontology Language (OWL) [W3C12]. OWL builds on RDF and RDFS by al-

```
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix ssn:  <http://www.w3.org/ns/ssn/> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix ex:   <http://example.com#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:TemperatureSensor a sosa:Sensor ;
    sosa:observes ex:RoomTemperature .

ex:Room1 a sosa:FeatureOfInterest ;
    ssn:hasProperty ex:RoomTemperature .

ex:RoomTemperature a sosa:ObservableProperty ;
    rdfs:label "Room Temperature"@en .

ex:Observation1 a sosa:Observation ;
    sosa:madeBySensor ex:TemperatureSensor ;
    sosa:hasFeatureOfInterest ex:Room1 ;
    sosa:observedProperty ex:RoomTemperature ;
    sosa:resultTime "2023-10-01T10:30:00Z"^^xsd:dateTime ;
    sosa:hasSimpleResult "22.5"^^xsd:float .
```

Listing 2.1: Example use of the Semantic Sensor Network Ontology represented in the Turtle serialization format.

lowing the creation of detailed ontologies that support automated reasoning and enabling systems to infer new information. Among further approaches this is accomplished through class and property definitions, formal constraints and logical axioms. One representative of a comprehensive ontology used in industrial settings is the Semantic Sensor Network Ontology (SSN) [Hal+17], which is an ontology for describing sensors and their observations to enable interoperability across different sensor systems. Listing 2.1 illustrates an example of instances based on the lightweight SOSA core ontology, which is included in SSN, demonstrating the machine-readable description of a sensor measuring room temperature. Vocabularies, on the other hand, consist of standardized terms and serve as dictionaries that provide shared meanings for data elements. A particular example is the Dublin Core [DCM12], which is a widely-used vocabulary for metadata, providing standard terms such as *dcterms:publisher* or *dcterms:format* to describe digital resources. While less expressive than ontologies, vocabularies establish a common semantic ground for particular applications or domains.

In the context of data analysis, ontologies and vocabularies are used to structure and enhance the classification of complex datasets and analytical models. Further details on concrete examples of ontologies and vocabularies within the context of

data science and Machine Learning are provided in Section 4.2. In this work we particularly focus on structuring knowledge related to or derived from analysis processes to allow for reusability and permit experts to draw semantically enriched conclusions that may not be immediately apparent (cf. Section 2.1.3). A self-developed ontology and associated vocabularies for the categorization of time series knowledge, are introduced as central contributions in Section 3.6 (Vocabularies) and Section 4 (Ontology).

2.2.2 Knowledge Graphs

Knowledge graphs [Hog+21] correspond to concrete data structures behind semantic concepts. They are networks of interconnected entities, properties and relationships, effectively representing how data relates within a particular context. Knowledge graphs leverage ontologies to embed semantics into data within a graph, where nodes represent entities or concepts and edges specify their relationships. More formally a knowledge graph can be defined as follows:

Definition 7. *Knowledge Graph*
Following the directed edge-labelled graph model [Hog+21], a knowledge graph KG is defined as a tuple:

$$KG = (E, R, T)$$

where E denotes a set of nodes representing entities, R is a set of relation types (edge labels), and $T \subseteq E \times R \times E$ is a set of triples (directed labelled edges), each connecting a subject entity $e_i \in E$ to an object entity $e_j \in E$ via a relation $r \in R$.

Knowledge graphs allow for the modeling of complex information as interconnected knowledge, forming the basis for advanced applications like semantic search [GMM03] and recommendation systems [Wu+22]. Particularly in artificial intelligence, knowledge graphs facilitate reasoning and inference by offering structured knowledge bases, thereby enhancing the ability of AI applications to process and understand complex data. Graph databases [AG08; Kal15] are specifically designed to store and query graph structures, making it easier to manage

and analyze interconnected data in knowledge graphs while aiming to support scalable and efficient operations. Popular examples [FB18] of graph databases include Neo4j [Mil13] and GraphDB [Ont24].

One recent innovation in the context of AI is the previously mentioned utilization of knowledge graphs by LLMs (cf. Section 2.1.3), a combination that enhances the models' ability to provide context-aware, knowledge-grounded reasoning. Graph RAG [Han+24a], a graph-based variant of Retrieval Augmented Generation (RAG) specifically leverages the structured relationships within a knowledge graph to improve the performance of LLMs. By utilizing Graph RAG, the LLM makes use of relevant nodes, edges, or subgraphs from the knowledge graph instead of relying solely on unstructured data. The graph's explicit structure helps to ensure that the retrieval is both contextually accurate and semantically meaningful.

In many analytical scenarios, directly processing data from knowledge graphs presents a challenge as most methods inherently rely on numerical data representations. Knowledge graph embeddings [Cao+24] provide a solution by mapping graph entities and their relationships into a continuous vector space. These embeddings transform complex, structured information into a format that can be utilized by Machine Learning models. Prominent examples include RDF2Vec [RP16], which learns node embeddings from random walks on RDF graphs and TransE [Bor+13], which learns entity embeddings with relation offsets. When applied to temporal data, knowledge graph embeddings can offer benefits by capturing inherent temporal relationships and patterns. Therefore, practical applications of knowledge graph embeddings in time series analysis include link prediction [Ros+21], which involves forecasting future interactions based on historical data, and event prediction [Li+23b], where future occurrences are inferred by combining temporal patterns and graph-based relationships. Overall, structured data from knowledge graphs enable the provision of semantic context for time series models. The integration of knowledge potentially improves the accuracy of predictions and also enhances their explainability.

2.2.3 Knowledge Retrieval, Reasoning and Validation

Beyond defining semantic structures, the practical value of a knowledge graph depends on the effectiveness of knowledge retrieval, the reliability with which new conclusions can be derived and how well data can be validated throughout its lifecycle. Retrieval, reasoning and validation usually represent major processing stages when interacting with knowledge graph data, where retrieval supplies existing facts, reasoning transforms available facts into new insights and validation enforces quality standards that maintain trust in both evidence and derived insights. Knowledge retrieval is concerned with issuing queries to model intent by exact lookups, path pattern matching or more comprehensive operations like embedding-based similarity search. Technically, retrieval includes a translation of information needs into graph schema and constraints, leveraging semantics such as subclass relations or information about domains and ranges. Query languages like SPARQL [AP11], the standard for RDF and Cypher [Fra+18], a query language for property graphs (e.g., Neo4j), enable common database operations such as filtering and joining data, as well as advanced tasks like recursive path traversal [Ros+86].

Reasoning extends retrieval by making implicit knowledge explicit under formal semantics. Deductive reasoning based on RDFS/OWL or rule systems utilizes type inheritance, property characteristics such as transitivity, symmetry, functionality or equivalence relations to derive new facts. In practice, specific tools like HermiT [Gli+14], Pellet [Sir+07] or ELK [KKS12] provide reasoning functionality, consistency checking and justifications. Particularly in property graphs, reasoning is often approximated via constraints, rule engines or graph algorithms that emulate inference patterns.

Finally, validation ensures that both asserted and inferred data adhere to intended semantics and quality demands. While ontologies define semantic expectations, constraint languages assure that these expectations are retained. One prominent example of a constraint language is SHACL [KK17], a W3C standard for validating RDF graphs. SHACL defines shapes that describe expected data patterns and constraints like cardinalities, datatypes and allowed value ranges and checks RDF data against these shapes to identify non-conformance. Besides SHACL, further

solutions for validation include ShEx (Shape Expressions) [Tho+19], which corresponds to a shape-based alternative and SHACL-SPARQL [KS25] to perform custom rule checks using SPARQL.

A more restrictive approach to simplify the complexity of schemas in advance to validation are Application Profiles (AP). These profiles define a set of rules and conventions for using ontologies and vocabularies in a specific context to meet particular requirements. Examples are DCAT-AP [Eur24], to specialize DCAT for European public data portals or the Dublin Core Application Profile (DCAP) [Bak+05], to constrain metadata records. Application Profiles are often specified and enforced via SHACL or ShEx, enabling automated validation and consistent documentation across systems.

Chapter Summary

This chapter provided the foundational context required to position the contributions of this thesis. On the analytics side, we formally defined time series and segments (Definitions 1–2), reviewed exploratory data analysis, Machine Learning paradigms, including supervised, unsupervised, discriminative, and generative learning and discussed automated, explainable, and LLM-based analysis approaches. On the Semantic Web side, we introduced ontologies, vocabularies, knowledge graphs, and the associated mechanisms for retrieval, reasoning, and validation. Together, these two pillars frame the research in which knowledge derived from time series analytics can be semantically classified, managed, and exploited.

What follows. Having discussed the foundations, Chapter 3 addresses RQ 1 by investigating which information qualifies as time series knowledge and how it can be systematically categorized into a hierarchically structured taxonomy.

3 Time Series Knowledge

The aim of developing a graph for the semantic organization of knowledge associated with or derived from time series data, primarily requires an understanding of what information qualifies as time series knowledge. Besides common analytical insights including anomalies, patterns or trends, many other types of information that could enhance data understanding or support data analysis, inherently relate to time series data. This information may include details on data quality or scenario-specific aspects such as associated event annotations.

As it may not be entirely possible to identify and encompass all information classifiable as time series knowledge in accordance with RQ 1, the goal of this section is to lay the groundwork for a rich, extendable taxonomy and semantic description in this regard. The following subsections provide an overview of common knowledge categories, which are either inferred from or implicitly linked to time series data.

Chapter Outline

This chapter explores the concept of time series knowledge, aiming to establish a foundation for its semantic structuring and taxonomy. It begins by defining what qualifies as time series knowledge including related challenges, requirements (3.1) and a derived, underlying time series information model. Next, it delves into the general properties of datasets (3.2) and the specific characteristics that define time series data (3.3). The chapter further discusses the various results inferred through time series analysis, such as event detection, forecasting and similarity measures (3.4). Additionally, it examines scenario-specific knowledge that contextualizes time series data and its applications (3.5). Finally, it proposes a structured taxonomy and modular, semantic vocabularies for organizing time

series knowledge (3.6) including an evaluation (3.6.4). The vocabularies developed in this thesis are published as open, versioned resources under persistent identifiers.

3.1 Definitions of Time Series Knowledge

Distinguishing between the terms *data, information* and *knowledge* is inherently challenging due to their interconnected nature, the often ambiguous boundaries between their definitions and the tendency to use them interchangeably. These terms are commonly used to represent different levels of significance or "meaningfulness". The study of what constitutes meaningful information and qualifies as knowledge falls under the domain of *epistemology*. While epistemology mainly addresses this question philosophically, this chapter rather focuses on defining knowledge in the context of time series data from a practical standpoint to determine a more accurate scope when creating a corresponding taxonomy.

Over time, many models have been developed to define different grades of data meaningfulness [BC04; BCM04; Zin07]. A widely recognized framework for differentiating layers of information significance is the DIKW hierarchy or also DIKW pyramid [Ack89; Row07; McD21], which stands for Data, Information, Knowledge and Wisdom. The DIKW hierarchy illustrates how raw data transforms into actionable insights through successive stages of processing, contextualization and understanding, where each stage adds complexity and value. However no official standardized definition of boundaries between these terms exist [AL01], which is why knowledge is sometimes called "actionable information" and wisdom is referred to as "accumulated knowledge" [McD21].

> *"Knowledge is an intrinsically ambiguous and equivocal term."*
> ***(S. Barnes, 2002)*** *[Bar02; Row07]*

Furthermore, the assignment of individual significance levels in practice is not a static process, and the importance of data does not always evolve linearly. A practical example supporting this assumption is the application of truth maintenance systems (TMS) [Doy79; RP11]. These systems are designed to dynamically

adjust and ensure the consistency of knowledge, as typically data relevance shifts over time depending on new insights.

Given such ambiguities, in this work we adopt a simplified view that mainly focuses on the distinction between raw data versus processed data, since the transition from information to knowledge and wisdom depends on the level of processing and contextualization. Thus, processed data collapses information and wisdom into knowledge, because a clear distinction is further complicated by multiple challenges involved when trying to precisely define the meaning of knowledge and individual layers:

Subjectivity: Knowledge is specific to domains and tasks. The same data can be valued differently depending on the user or application.
Example: Data may provide valuable insights for a particular application scenario, but might be irrelevant noise for another one.

Dynamic: Data, especially time series data, changes over time and so does the knowledge derived from it. What is considered knowledge today may lose relevance tomorrow.
Example: Patterns may shift due to changing influential factors and identified anomalies might turn out to be reoccurring, unimportant events in the future.

Confidence: Identifying the influence of complex information and whether it can consequently be considered knowledge is challenging.
Example: Does a particular observation in the data represent a meaningful event, or is it merely a negligible observation?

Originality: Not all additional information adds new value. Some may rather be considered an alternative data representation than new knowledge which adds further value.
Example: Can the result of a data transformation automatically be regarded as new knowledge?

Overall, the previous challenges on what identifies as knowledge depend on time, application domain and exploitation scenario. By summarizing these factors under context, in the scope of time series, knowledge can be expressed

as the relationship between temporal data and its context defined by a mapping function that assigns semantic meaning to raw observations. While there may not be an exact formalism for this mapping, the core idea that knowledge is defined by contextualized relationships, is widely supported in various fields, including knowledge representation and data analytics. For example, [Dey01] defines context as "any information that can be used to characterize the situation of an entity" and highlights that knowledge arises when data is transformed using contextual information. Similarly, [MB94] introduces the formalization of context in knowledge representation, emphasizing its role in assigning meaning to data. In accordance with these views, Davenport and Prusak [DP98] made a popular statement that underscores how knowledge arises from context, including experience and expertise:

> *"Knowledge is a fluid mix of framed experience, values, contextual information and expert insight that provides a framework for evaluating and incorporating new experiences and information."*
> *(**Davenport & Prusak, 1998**) [DP98]*

In time series analysis, the importance of domain-specific context in interpreting temporal patterns and structures is highlighted in different publications. For instance, [Box+15] emphasize the role of domain expertise in understanding the behavior of time series data, stating that the interpretation of time series models cannot depend on mathematical criteria alone. Similarly, [Cha00] highlights that findings such as anomalies in time series gain significance only when contextualized within the underlying system.

To specify the role of context in time series knowledge, we treat contextualization as the basis for assessing validity, quality and the potential influence of particular knowledge. In practice, context comprises situational metadata (e.g., domain constraints, operational settings or annotations) that make observations interpretable and evaluable. For example, knowing that a signal should obey a particular differential equation provides necessary context, without which the signal may otherwise not be semantically interpretable. One important problem with this definition is that some context-agnostic but statistically significant ob-

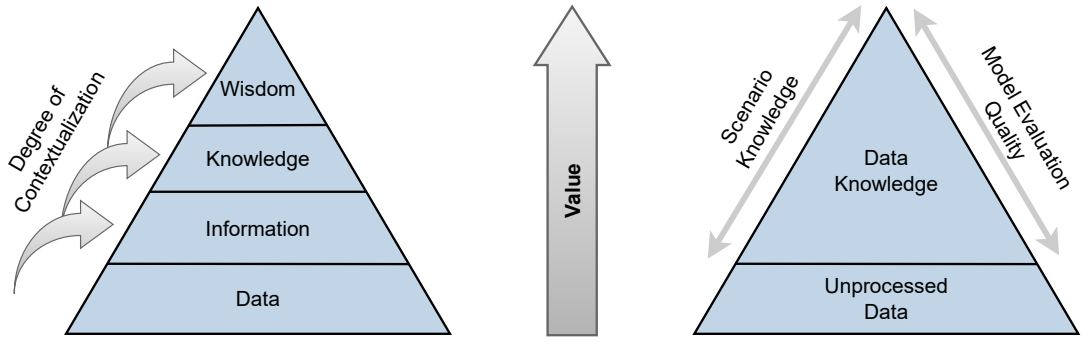


Figure 3.1: Two different models describing the value of processed data: DIKW [Row07; McD21] model with discrete, unidirectional progression (left side) and our customized model with continuous, bidirectional progression (right side).

servations (e.g., emergent deviations and motifs) can be valuable as provisional or potential evidence even before contextual validation. This circumstance is particularly true for knowledge repositories that support later reinterpretation under new or evolving contexts, or allow for reframing questions as understanding evolves. In other words: Subsuming any processed data as knowledge is a pragmatic choice given that the significance of existing data can dynamically depend on newly received information and as such any knowledge is potentially actionable.

Throughout this work we propose a definition that primarily distinguishes two complementary types of knowledge - *Data Knowledge* and *Scenario Knowledge*. Rather than explicitly defining discrete significance levels, classifying knowledge into categories associated with one of the two types allows for the development of a generic, application-agnostic scheme. Data knowledge represents insights extracted or inferred directly from time series data, describing its inherent characteristics and analytical findings determined by an underlying model. The second type, Scenario Knowledge, captures contextual information (e.g., labels, scenario parameters, notes, events) and can be used to assess the value, relevance and interpretation of Data Knowledge. Although additional categories - such as methodological knowledge capturing specifics of analysis methods - may be identified and defined as separate types in a taxonomy, at the most abstract level

they can be viewed as belonging to one of the two types or intersecting both. In the example of methodical knowledge, this knowledge may be derived from model training or inference, which places it under Data Knowledge, or it may come as a priori expertise from process experts, such as a process-driven model parameterization, in which case it constitutes Scenario Knowledge. Using the proposed approach, the significance of knowledge is generally assessed by combining results of evaluation metrics derived from a specific model's inference and the type of Scenario Knowledge involved. For instance, in the first case, this significance could correspond to a concrete value for a metric like the Root Mean Squared Error (RMSE), while in the latter case, it might involve the semantic difference between a hypothesis and a fact. Figure 3.1 shows a comparison of DIKW with the proposed alternative model. While DIKW assumes a layer-wise, value-adding ascent from data to wisdom through contextualization, we suggest modeling the change of significance through new information as a bidirectional and continuous flow. In the latter case, the value of available knowledge depends on the amount and type of Scenario Knowledge, as well as on the quality of inferred knowledge measured via corresponding metrics.

Throughout the following definitions, we distinguish between unprocessed data including raw temporal data, contextual information and uninterpreted analysis results, and their semantic representation as knowledge. While analytical functions produce results from raw temporal data and potentially unstructured contextual information, inputs and outputs become classified knowledge only through a representation function $\mathcal{R}$ that assigns formal semantic meaning. The concrete specification of $\mathcal{R}$, realized through the vocabularies and ontologies developed in Section 3.6.3 and Chapter 4 constitutes a core contribution of this work. We define both knowledge types as follows:

Definition 8. *Data Knowledge*
Let X denote temporal data (cf. Definition 1). Data Knowledge K consists of semantically classified characteristics derived from X via a function $\phi \in \Phi$ parameterized by p and represented through $\mathcal{R}$:

$$K = \mathcal{R}(\phi_p(X)),$$

where Φ denotes a family of mathematical or algorithmic models generating potentially significant insights from X, and $\mathcal{R}$ maps these insights to structured, semantically typed knowledge entities.

Different choices of $\phi \in \Phi$ and p represent different analytical methods with varying parameterization (e.g. trained ML models) with concrete examples of ϕ being discussed in the following sections, starting from Section 3.2.

Definition 9. *Scenario Knowledge*
Let Y denote raw contextual information. Scenario Knowledge C is the semantically classified representation of Y, comprising class labels, rules and further scenario-specific information:

$$C = \mathcal{R}(Y).$$

C can be used to condition the interpretation and valuation of Data Knowledge K via a mapping $\psi \in \Psi$:

$$K^* = \mathcal{R}(\psi(K; C)),$$

where Ψ denotes a family of contextualization mappings and K^* is the resultant contextualized, semantically enriched knowledge obtained by interpreting K under C.

In case that C serves as input for informed analyses (cf. 2.1.3), $\psi \in \Psi$ might instead be formalized as $\phi \in \Phi$ augmented by an additional argument for Scenario Knowledge such that C is leveraged to contextualize ϕ directly via $\mathcal{R}(\phi_p(X; C))$.

Although the previous definitions remain slightly informal due to the application-specific and ambiguous specification of knowledge significance, they enable a practical classification of different knowledge types and their mutual progression. In the next sections, we introduce categories of Data Knowledge and Scenario Knowledge to derive a taxonomy that will guide the development of extendable, semantic knowledge vocabularies. We make no claim of completeness, given the ongoing evolution of applications and associated methodologies.

It is important to note that the artifacts for knowledge representation developed in this work, namely the taxonomy, the ontology, and the vocabularies, were designed through an iterative process whose development order differs from the

presentation order chosen for readability. In practice, the taxonomy was developed first to identify and categorize what constitutes time series knowledge and to establish the conceptual scope for subsequent formalization steps. Based on this scope, the ontology was developed next to define a coarse structural abstraction that organizes how knowledge is represented, linked to time series segments, and propagated through analytical workflows, independent of specific knowledge types. The vocabularies were then created to populate the ontology's concept layer with concrete, hierarchically structured terms that instantiate the taxonomy as machine-readable version. For readability, this thesis presents the taxonomy and vocabularies together in this chapter as they primarily constitute the representation function $\mathcal{R}$ introduced in the preceding definitions. The ontology is then introduced in Chapter 4.

Concerning RQ 1, we aim to provide a comprehensive overview of knowledge categories starting with Data Knowledge. Apart from a description of each category, we will discuss candidates of knowledge per category and outline how they are generated or utilized in the scope of time series analysis drawing on selected approaches, including own contributions. Beyond expert input, for a categorization of Data Knowledge, we primarily considered publications [And11; EA12; Ham20; Wan+24] from literature research, with some publications being cited in the corresponding section. Furthermore we made use of relevant information available in online repositories [Chr26; Hu26] providing curated lists of further publications and prominent analysis libraries (cf. Section 3.6.4).

In advance to the individual subsections, a preliminary overview of categories and corresponding Data Knowledge examples is provided in the following table (Table 3.1).

Data Knowledge: Categories And Examples
General Dataset Properties
Observations Per Series, Time Span, Sampling Frequency, Missing Rate And Gap Lengths, Duplicates, Value Range.
Descriptive Statistics
Mean, Median, Standard Deviation, Quantiles, Interquartile Range, Skewness, Kurtosis, Entropy.

Continued on next page

Structural Profiles
Trend Slope, Seasonality Period And Strength, Dominant Frequency, ACF/PACF Lags, Volatility Level, Kernel Expressions.
Instance-Specific Descriptions
Shapelets
Inferential Statistics And Causality
Granger-Causality, Causal Graph, Dominant Lag Effects, Cointegration Vectors, Causal Ordering, Counterfactuals, Conditional-Independence Test Results.
Event Discovery And Prediction
Anomaly Intervals And Scores, Change Points, Inferred Event Timestamps And Probabilities, Threshold Breaches, Lead Time, Hazard Rates, Survival Curves, Time-To-Failure.
Regressions And Forecasts
Polynomials, Prediction Values, Confidence Intervals
Dissimilarities / Similarities
Distance Matrix, Cosine Similarity, Nearest Neighbors, Alignment Paths.
Groupings
Clustering, Outliers, Representatives, Hierarchy Levels, Class Membership Estimates.
Patterns And Associations
Matrix Profile, Pattern Chains, Frequent Subsequences With Support, Association Rules, Lagged Co-Occurrences, Sequential Rules.
Projections And Mappings
Symbolic Strings, Principal Components And Explained Variance, Projection Coordinates, Fourier Coefficients, Latent Vectors.

Table 3.1: Categories of time series analysis approaches, represented by associated knowledge output examples.

3.2 General Dataset Properties

The category of dataset properties corresponds to summarizing data information used in the initial stage of most data analysis workflows [RD00; AGN15]. Typical representatives are definitions of data types or quantitative metrics such as the number of unique values. While knowledge of this category could be considered basic metadata, it enables a preliminary evaluation of the dataset's structure and suitability for further analyses. It is crucial for assessing the dataset's reliabil-

ity and the potential accuracy of generated insights. We partition this type of knowledge into two major subcategories:

Structure and Scale: This category defines properties that can be regarded basic data summary descriptors. One example is the total number of records to assess the dataset's extent. A small number of records, for instance, might require the addition of synthetically generated data when being used to train a model. Similarly the dimensionality of the data provides valuable insights, enabling adjustments to model selection strategies ensuring they are better equipped for handling high-dimensional data. Another example is the sampling frequency to determine the temporal granularity and resolution of the data. This property not only serves as an indicator for the level of detail captured in the dataset, but also has a direct impact on subsequent processing steps such as interpolation or imputation. Furthermore, it can help to align the dataset with the temporal requirements of specific analytical models.

Quality: This category can be regarded complementary, qualitative knowledge derived from the previous category, as this information already evaluates the reliability of the data. Examples include the number of missing values to reflect the dataset's completeness, providing a potential reason for both underperforming and overfitted models, or the number of duplicates to uncover biases or inconsistencies.

The purpose of general data information thus goes beyond descriptive metrics. Data information serves as a practical foundation for data preparation, inform model selection and support the interpretation of model evaluation. By understanding the impact of individual properties from this category, one can enhance the validation of methodical and semantic correctness.

3.3 Time Series Characteristics

We define time series characteristics as a category of knowledge that captures the unique, characteristic traits of time series data, reflecting its visual and structural dynamics. These characteristics provide standalone insights but are often used

as additional features to inform task-specific analysis methods [Ful18]. Examples include insights into temporal patterns like seasonalities and localized behaviors such as sudden changes, offering a deeper understanding of the dataset's nature. This section classifies different characteristics of time series data, each focusing on complementary aspects. *Descriptive Statistics* reveal central tendencies, variability and basic trends. *Structural Profiles* represent patterns and periodicities that define the data's inherent behavior. *Instance-specific Descriptions* emphasize sample-based representative knowledge, such as shapelets [Li+21] to categorize individual observations.

3.3.1 Descriptive Statistics

Descriptive statistics [And11] are concerned with summarizing statistical key features of data. The knowledge derived from descriptive statistics allows for an understanding of the dataset's central tendencies, variability and overall distribution. For example, descriptive statistics reveal insights such as whether the data is evenly distributed, skewed, or clustered around certain values. In the context of time series, knowledge of this category can be both global and local with respect to time. While descriptive measures including the mean, variance and quantiles are commonly used to describe global properties of a dataset, aggregation operations based on predefined time windows enable a segment-specific (cf. Definition 2), local view on the data. Like other time series characteristics, descriptive statistics are often used as additional input features [Ful18] to analysis models, where they are especially helpful to enhance model interpretation by enriching semantic information. In many cases, they are part of an associated feature engineering and selection stage of the model.

One example for the use of descriptive statistics in time series classification is DrCIF (Diverse Representation Canonical Interval Forest) [Mid+21]. Random intervals are sampled and descriptive statistics (mean, standard deviation, slope, quantiles) are computed for each segment to capture local and global structure. These summaries form the feature vector for a tree-based ensemble, acting as a built-in preprocessing step that enriches the representation and supports interpretable decisions. In forecasting frameworks like FFORMA (Feature-based

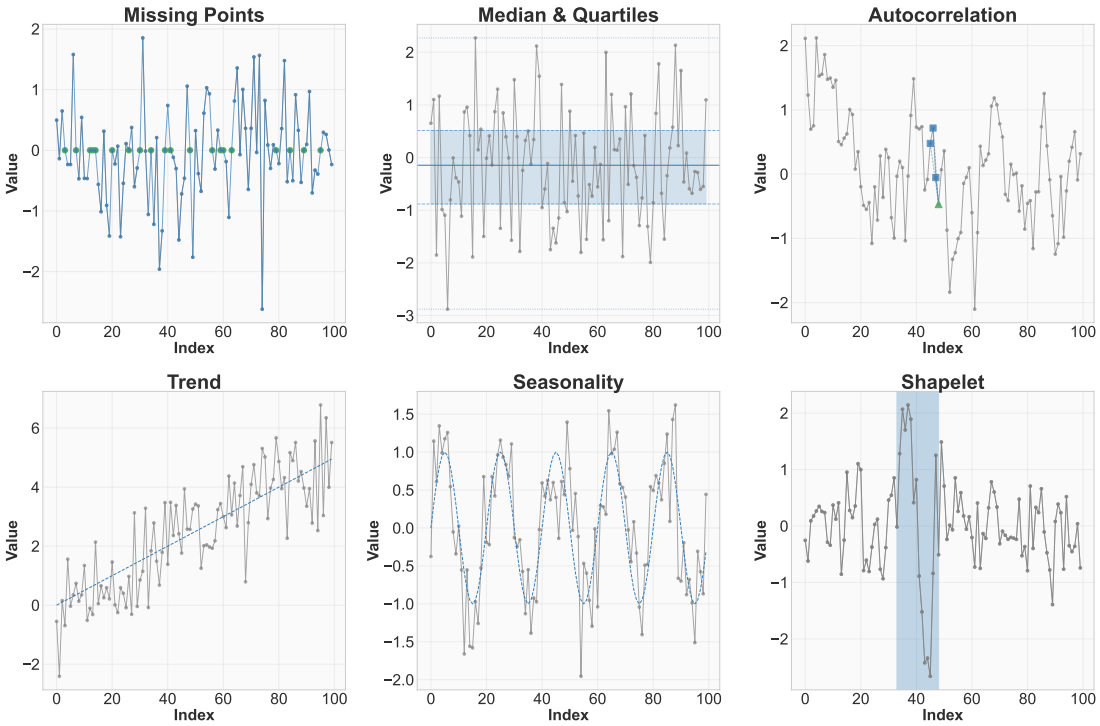


Figure 3.2: Examples of general dataset properties (Section 3.2) and time series characteristics (Section 3.3).

Forecast Model Averaging) [Mon+20], descriptive statistics (skewness, autocorrelation at key lags, spectral entropy) are extracted to characterize each series. These meta-features are fed to a learner that selects and weights base forecasters, improving predictive performance. Finally, representing our own contribution to the exploitation of descriptive statistics for unsupervised anomaly detection, GADPL (Generic Anomaly Detection for Production Lines) [GBC18] is a configuration-aware analysis pipeline that describes windows of categorized time series data by statistical features. It then computes an average k-nearest-neighbor distance in feature space as a dissimilarity score and flags windows as anomalous when this score exceeds a threshold.

3.3.2 Structural Profiles

Structural profiles provide insights into the underlying patterns and structural properties of time series. Whereas the category of descriptive statistics represents a high-level summary, structural profiles aim at capturing inherent behaviors and regularities, enabling local or component-level semantic descriptions of signals. An important knowledge class within structural profiles comprises kernel expressions, which can be used to describe functional relationships in the data using individual kernels or kernel compositions [Duv14]. By leveraging the semantics of each kernel in tasks like regression or classification, these representations denote concrete observation behaviors like an oscillating signal with increasing amplitude. The central idea is to identify kernel expressions whose semantics align with the signal's properties. Algorithms that discover such kernel-based structural profiles include ABCD [Llo+14], CKS [Duv+13] and own contributions [GDB22]. The latter approach, for instance, identifies suitable kernel expressions from artificially generated knowledge samples learned by neural networks, illustrating how generated knowledge can propagate recursively to derive new knowledge. Another, more task-specific contribution for anomaly detection is presented in [Bee+19].

Further structural components of time series can be revealed by various decomposition methods. In addition to techniques such as LOESS (Locally Estimated Scatterplot Smoothing) [Jac00] for describing trends and seasonal patterns, frequency-domain-specific decomposition methods exist to allow for the identification of recurring cycles or hidden oscillations in time series data. Examples of associated methods to generate such knowledge include the Fast Fourier Transform [CLW69] or Power Spectrum Analysis [Koo95]. By examining where the signal is concentrated across frequencies they can identify spectral profiles and highlight harmonic structures. These representations reveal regime organization and disorder, where a rising entropy can indicate transitions from stable to chaotic behavior. This property can for instance be used in change-point detection, where abrupt shifts in entropy or information-theoretic divergence indicate boundaries between adjacent segments.

3.3.3 Instance-specific Descriptions

Instance-specific descriptions are characteristic patterns or behaviors representing concrete, individual segments of a time series or subsets of a dataset. Compared to previous categories the goal is not to describe a time series based on inferred properties but rather on representative candidates.

An example of instance-based descriptions are shapelets [YK09], being compact, discriminative subsequences within time series data that capture meaningful particularities. Shapelets encapsulate localized behaviors, such as sudden spikes, dips, or transitions, that may be indicative of specific phenomena or events. For instance, in a medical dataset, a distinct shapelet could correspond to an abnormal heart rhythm in an electrocardiogram signal, providing valuable insights for a diagnosis. By focusing on these instance-level patterns, shapelets enable a detailed differentiation between observations and help in tasks such as classification or anomaly detection. An example of utilizing shapelets for anomaly detection was already illustrated in Figure 2.2. Algorithms that discover or exploit shapelets comprise the Shapelet Transform [Lin+12b], Shapenet [Li+21] and HIVE-COTE (Hierarchical Vote Collective of Transformation-based Ensembles) [Mid+21].

3.4 Task-driven Analysis Results

The previous section focused on the categorization of descriptive characteristics of time series data. Figure 3.2 provides an illustrative recap of some of the candidates from these categories. In contrast, the upcoming subsections explore knowledge gained through analytical tasks. Although there exists semantic overlap, since most of the discussed knowledge is generated via some analytical procedure, analysis results are still to be considered a distinct category. Knowledge of this category generally encompasses more complex insights into the data, derived from advanced analytical methods rather than for instance descriptive statistics. Another reason for this distinction is that most of the previous knowledge categories describe general, scenario-agnostic data characteristics that are not necessarily specific to tasks but only to data. Therefore, a central aspect of knowledge in this category is its underlying purpose. Unlike knowledge defining

descriptive properties of the data, insights derived from analytical methods are tied to answering questions associated with a particular task. A relevant question might be whether anomalies exist in the data, with anomaly detection serving as the task and the discovered anomalies representing the resulting knowledge.

3.4.1 Inferential Statistics and Causality

The category of inferential statistics and causality explores relationships between variables, distinguishing associative patterns from directed influences to provide an understanding of dependencies and driving factors within the data.

Inferential statistics focus on generalizing insights from sample data to a broader population. The primary aim is to estimate population properties, determine relationships between variables, and evaluate the likelihood that observed patterns are due to chance. For example, inferential statistics help to assess whether a factor significantly impacts an outcome or whether differences between groups are statistically meaningful. These methods are essential for making predictions, validating hypotheses, and guiding decisions when dealing with uncertain or incomplete data. Common inferential approaches for instance comprise hypothesis testing methods like the Augmented Dickey-Fuller test [DF79; SD84] to assess stationarity. However, while inferential statistics can identify associations, they do not inherently establish causality without additional assumptions or experimental designs.

In contrast, methods for causal determination go beyond identifying associations to uncover and quantify cause-and-effect relationships. Figure 3.3 shows such a relationship, visually indicated by a significant cross-correlation between two dimensions. State-of-the-art methods include PCMCI+ [Run20], which performs causal discovery in multivariate time series via lagged conditional independence testing or Bayesian structural time series models [Bro+15], which estimates counterfactuals to quantify intervention effects. Other approaches specific to temporal data are Granger causality [Gra69; SF22] and Transfer Entropy [Sch00; Bos+16]. Granger causality extends simple correlation by checking whether past values of one variable improve predictions of another, revealing directional relationships.

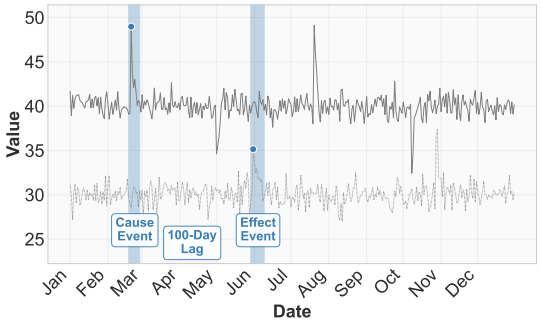


Figure 3.3: Causal Effects

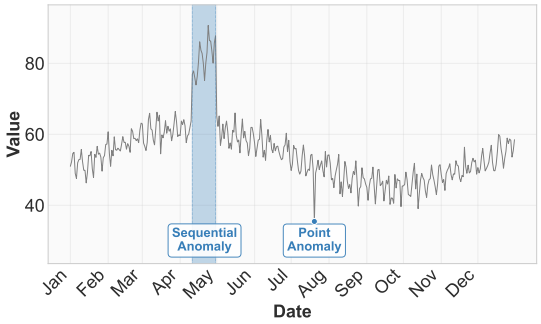


Figure 3.4: Anomalies / Discords

Transfer entropy complements Granger causality by capturing nonlinear dependencies and measuring the directional flow of information between variables.

3.4.2 Event Discovery and Prediction

Event discovery focuses on identifying specific occurrences of phenomena within time series data. Beyond simply identifying these phenomena, such knowledge is useful for deriving potential causes or impacts. This category often includes tasks, where the goal is to identify deviations from expected behavior or tasks to predict system failures.

An example of event discovery is anomaly detection [SWP22], which identifies irregularities including sequential anomalies or individual point outliers in the data. For instance, in network traffic data, an unusual spike in activity might indicate a potential security breach, while in sensor data, a sudden fluctuation could signal a machine failure. Figure 3.4 depicts a univariate time series with two detected anomalies. Methods identifying anomalies include Anomaly Transformers [Xu+22] leveraging attention to detect context-dependent anomalies or the previously mentioned GADPL [GBC18] method, representing an own solution to the problem of anomaly detection. Another common example of knowledge derived from event discovery are changepoints, which allow for a segmentation of time series segments into subsegments with distinct signal behaviors. A concrete state-of-the-art approach is PELT [KFE12], a method that is able to leverage distance or entropy-based cost functions for localizing changes in adjacent seg-

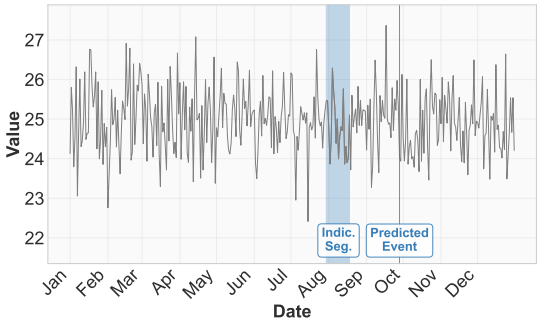


Figure 3.5: Predicted Event

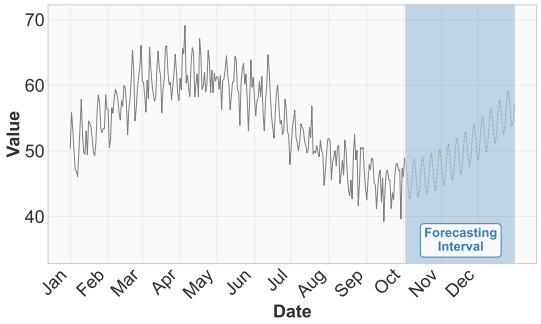


Figure 3.6: Forecast

ments.

In comparison, event prediction relies on information corresponding to indicators or early warning signals that precede such events. Examples are natural disasters like earthquakes indicated by preceding characteristic patterns including changes in groundwater levels and seismic foreshocks. Figure 3.5 illustrates an example of event prediction based on the occurrence of a learned predictive signal (indicative segment). The knowledge gained from event prediction enables actionable insights by linking observed data to future outcomes, making it valuable for risk management and decision-making. A representative method to infer events of this class is the deep learning based approach DeepHit [Lee+18].

3.4.3 Regressions and Forecasts

In contrast to a prediction of discrete events, regression and forecasting primarily aim to model and predict continuous variables. The knowledge derived from these methods provides insights into relationships between variables and enables a comprehension of future trends.

Common examples of regression are linear and polynomial modeling [Jam+23], where the relationship between single or multiple dependent variables and one or more independent variables is established. For instance, a regression model might be used to explain energy consumption based on historical data and external factors like weather conditions. This knowledge helps to identify key drivers and quantify their impact on the outcome.

Forecasting, on the other hand, focuses on predicting future values based on historical time series data. Methods such as ARIMA [BJ70; SS17], exponential smoothing, or machine learning-based approaches like LSTMs [HS97; STN19] are commonly used. Application scenarios for forecasting include the prediction of stock prices, sales demand, or traffic patterns. The knowledge derived from forecasting is crucial for understanding trends, planning resources and anticipating challenges. One specific example is a solution [Ara+19] for continuous load prediction in energy management systems using LSTMs. Given that forecasts represent knowledge about time series, their value primarily lies in their interpretation through subsequent analyses, such as identifying patterns and trends within the forecasted data. With respect to the example in Figure 3.6 it might potentially be of interest that the predicted signal shows a clear positive trend.

3.4.4 (Dis-)Similarities

Analyses of (dis-)similarities involve quantifying the extent to which time series segments, complete time series or even time series collections resemble or differ from one another. Dissimilarity measures can for instance be used to compare seismic waveforms when detecting similar earthquake events [BB19] or to perform query-based multimedia retrieval [BG16] in databases. Furthermore, this knowledge is often established implicitly when identifying patterns, detecting anomalies or clusterings, since a measurement of (dis-)similarity frequently is an essential part of corresponding algorithmic solutions.

One representative of a popular dissimilarity measure is dynamic time warping (DTW) [BC94], which measures the aligned distance between two time series sequences using a base distance metric, even when they differ in length. Figure 3.7 provides an example of DTW being applied for a query-based identification of similar time series segments based on the degree of dissimilarity.

Conversely, similarity analysis [Cas+12] focuses on quantifying how closely for instance time series segments resemble each other in terms of their shape, trends, or patterns. One widely used similarity measure is the cosine similarity, which evaluates the cosine of the angle between two vectors in a multidimensional space. Cosine similarity can be particularly useful when comparing normalized time series,

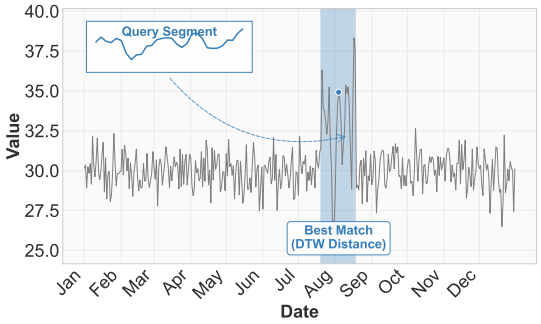


Figure 3.7: (Dis-)Similarity

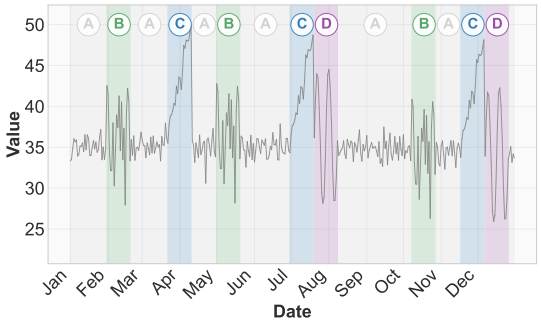


Figure 3.8: Subsequence Clustering

as it focuses on the alignment of their patterns rather than their magnitude.

3.4.5 Groupings

The category of groupings represents the organization of time series data into clusters, classes or other groups based on shared characteristics or behaviors. This knowledge is crucial for understanding the distribution of particular segments, for describing sets of representatives for different time series and for enabling further tasks like segmentation or prediction.

One example for a grouping is clustering [Zha+19; ASW15; Alq+21], where time series segments are partitioned into groups that exhibit similar behavior. Figure 3.8 shows a corresponding example for a univariate time series. Clustering techniques mainly refer to methods from unsupervised learning (cf. Definition 3) and include time series specific variants of k-Means [NR07; Hua+16], hierarchical clustering [RGP08], or DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [Est+96; BK07]. For instance, in motion capture analysis, clustering might group individual persons based on their movements, revealing distinct characteristics when performing actions [Ari+20].

The supervised equivalent of clustering can be regarded classification [Fao22], which allows to infer the assignment of new observations into concrete classes. In contrast to clustering, groups are already predefined, which is why corresponding methods belong to the category of supervised learning approaches (cf. Definition 4). For example, in healthcare, classification is used to categorize ECG sig-

nals into different types of arrhythmias, aiding in diagnosis. [Xia+23; Ans+23].

3.4.6 Patterns and Associations

The category of patterns and associations defines recurring structures, relationships, or dependencies within time series data. Similar to clustering results, such knowledge provides insights into the underlying dynamics of time series, enabling better understanding and prediction of system behaviors.

One type of pattern discovery is motif discovery (see Figure 3.9), which identifies repeatedly occurring subsequences within time series data. A popular method to determine motifs in time series data is the Matrix Profile [YZU+16; Lu+22]. The Matrix Profile computes the pairwise similarity between all subsequences in a time series, enabling the identification of motifs and discords (anomalies). Motifs extracted using the Matrix Profile can then be used in further analytical tasks, such as clustering similar patterns, building predictive models, or detecting meaningful correlations across different time series datasets.

Another example of this category are association rules, which describe relationships between variables or events. For instance, in retail, association rules might identify that an increase in sales of a particular product often coincides with a rise in demand for complementary items. Thus, the category of association rules is related to the category of causality. Concrete representatives of algorithms identifying sequential rules are SPADE (Sequential Pattern Discovery using Equivalence classes) [Zak01] and SPAM (Sequential Pattern Mining) [Ayr+02].

3.4.7 Projections and Mappings

Projections and mappings focus on transforming time series data into alternative representations that reveal knowledge representing hidden structures or simplifying subsequent data analyses. The knowledge derived from these transformations enables better visualization, feature extraction and model interpretation.

Examples of projections include results from dimensionality reduction techniques like PCA (Principal Component Analysis) [Pea01; AW10] or t-SNE (t-Distributed

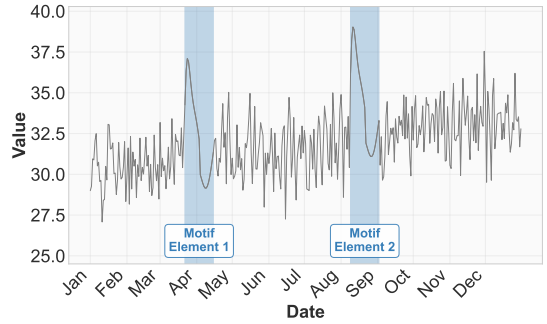


Figure 3.9: Pattern / Motif

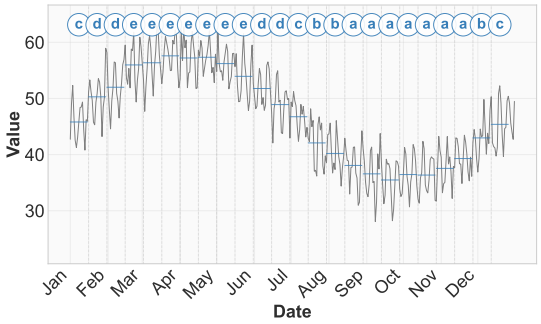


Figure 3.10: Discrete Projection

Stochastic Neighbor Embedding) [MH08]. For instance, PCA can reduce the complexity of multivariate time series data by identifying the principal components that explain most of the variance and thus compress the data to the most informative aspects.

Other examples of mapping time series data to alternative feature spaces are wavelet transform [Zha19] and symbolic aggregation [Lin+07]. In the case of wavelet transforms, time series data is decomposed into frequency bands, highlighting localized features or trends. Similarly, SAX (Symbolic Aggregate approXimation) transforms time series into a sequence of discrete symbols, enabling efficient pattern discovery while preserving a coarse, discretized representation of the data structure. Figure 3.10 shows an application of SAX to discretize a time series of continuous values, resulting in a sequence of discrete classes.

3.5 Scenario Knowledge

Scenario Knowledge captures the contextual information that provides meaning to characteristics of time series data. Section 3.1 already discussed the importance of contextualized information and defined Scenario Knowledge in a formalized way (cf. Definition 9). Scenario Knowledge encompasses information like data annotations, practitioner notes, formal expressions of domain logic and the parameters that define concrete analysis settings.

In most scenarios, the knowledge is generated through a mutual interaction between human expertise and automated processes [CG23]. Domain experts inte-

grate background knowledge from experience, documentation and facts, or provide annotations for available historical data. Detection systems additionally identify events or derive insights and propose candidate labels, while also verifying generated knowledge based on predefined formalism and conditions. Finally, human experts might further validate predicted findings based on ground truth information and use the resultant knowledge to redefine and optimize dependent parameters or derive decisions.

Similar to Data Knowledge, the development of a taxonomy and a comprehensive vocabulary for different types of Scenario Knowledge involved identifying a wide range of information sources. In fact, Scenario Knowledge exhibits an even wider range and significantly greater levels of granularity, as it is highly domain- and application-dependent. This complexity is particularly evident in the manufacturing domain [Sap+25]. Furthermore, Scenario Knowledge comprises more implicit information, which is only indirectly linked to temporal data. For instance compared to event data or anomalies, a guideline from a manual does not immediately look like data which is aligned with temporal information. However, if this guideline defines a suggested temperature range, this input is valuable information to specify significant outliers in time series data. Another example would be informal notes on the severity level of a machine failure, where the machine failure is the concrete temporally-linked information and the severity level specifies this information.

The selection of sources for deriving Scenario Knowledge categories followed an iterative procedure. Starting from an initial iteration that identified broad contextual categories through brainstorming and project experience - some of them are discussed in Section 5.1 - we systematically searched for established standards that provide normative definitions for the identified types of contextual information. The guiding criterion was for each thematic category to identify recognized specifications that defines relevant concepts to derive vocabulary terms. We organized the search along the stages of a typical time series analytics workflow: from data acquisition and sensing infrastructure, through event logging and operational context, to interpretation, formal documentation and validation. This workflow-oriented structuring should ensure that the resulting vocabulary covers

the contextual information that practitioners encounter at each stage of working with time series data. Where multiple standards addressed overlapping concerns, we prioritized those from industrial contexts due to the prevalence of sensor-equipped devices. However, standards specific to narrow application domains were generally excluded to maintain a domain-agnostic taxonomy, ensuring that domain-specific knowledge can later be appended without modifying the core vocabulary. The following categories were identified together with their normative sources and examples of derived taxonomy terms:

Provenance, Annotations, and Data Quality. *Standards for attaching metadata, tracking provenance, and assessing data quality.*
The W3C Web Annotation Data Model [SCY17] focuses on annotations, motivations and temporal selectors, enabling the linkage of textual comments to specific time periods or media fragments. Together with the W3C PROV-O ontology [Leb+13], which provides a standardized model of activities, entities and agents along with corresponding relations, these specifications support core Scenario Knowledge notions such as `Annotation`, `Event`, `Involvement`, `AffectedEntity`, `CausalFactor` and `Fact`. Complementary, the W3C Data Quality Vocabulary (DQV) [AI21] defines quality dimensions, metrics and quality measurements, supporting the representation of `QualityAnnotation`, assumptions about data (`Assumption`), certainty measures (`Certainty`) and status-like quality descriptors (`Status`) associated with observations and events.

Measurement and Sensing Infrastructure. *Standards describing the physical and technical context in which time series data is acquired.*
The W3C/OGC SOSA/SSN ontology [JHC+19] provides a semantic framework for describing sensors, actuators, sampling procedures and deployments, while the OGC SensorThings API [Lia+21] complements this by providing a standardized interface for managing sensor metadata and historical location information. These standards motivate candidate concepts that capture the infrastructure context underlying time series acquisition: `Deployment` for representing the installation and configuration of measurement systems, `SensorDeployment` for specifying the deployment of individual sensors, and `Location` for capturing the spatial context where measurements are taken. Additionally, `EnvironmentalCondition`

can describe ambient factors such as temperature, humidity or pressure that may influence sensor behavior or measurement quality. Understanding this infrastructure context is essential for interpreting time series data, as sensor characteristics, deployment configurations and environmental factors can all affect observed values and their reliability. Note that while these sources also define observation and measurement concepts for the actual data acquisition process, this category and Scenario Knowledge in general do not include information about raw data.

Events, Processes, and Operations. *Standards covering event logging, process modeling, lifecycle transitions, operational context, and user interactions.*
The IEEE 1849-2016 XES (eXtensible Event Stream) [IEE16; Aca+17] standard enables the logging of events, traces and lifecycle transitions for process mining, motivating concepts such as `LifecycleEvent`. OMG BPMN 2.0 [Obj11] provides a detailed framework for modeling business processes, encompassing various event types and milestones, supporting concepts such as `CategorizedEvent`, `Error`, `ProcessStep`, `Milestone` and `UserEvent`. W3C ActivityStreams 2.0 [SP17] defines structured ways to represent user actions and activity streams, supporting `BusinessEvent` and `Action`, while iCalendar (RDF) [CM05] defines calendar entries and scheduled events, motivating `CalendarEntry`. Protocols such as RFC 5424 Syslog [Ger09] define the structure of log messages, including severity and facility classifiers, aiding in the categorization of system events like `Alert`, `Severity` and `Code`.

In industrial production environments, operational metadata like maintenance schedules, machine settings or production shifts provide meaningful context for time series. The ISO 55000 series [Int24] provides a framework for asset management, establishing principles and terminology that link asset lifecycle considerations to time series segments, while ISO 14224 [Int16a] specifies reliability and maintenance data collection standards with a comprehensive taxonomy for failure modes, maintenance activities and equipment hierarchies; although originally developed for the petroleum and natural gas industries, it can be adopted as a reference model. Together these standards support concepts such as `MaintenanceEvent`, `PreventiveMaintenanceEvent`, `CorrectiveMaintenanceEvent`, `PredictiveMaintenanceEvent`, `MaintenancePlan`,

`AffectedEntity`, `CauseCode`, `DefectType` and `SensorReplacement`. ISO 22400 [Int14] specifies key performance indicators and state models for manufacturing operations, including downtime categories, planned and unplanned stops, changeovers, production cycle boundaries and equipment operating modes, while the OMAC PackML / ISA-TR88.00.02 specification [ISA08] defines a standardized machine state model. These standards directly transfer into vocabulary concepts such as `PlannedDowntime`, `UnplannedDowntime`, `SetupChangeover`, `ProductionCycleStart`, `ProductionCycleComplete`, `OperatingMode` and `Status` as contextual descriptors of time series behavior. ISA-88 (IEC 61512) [ISA95] defines batch control concepts including recipes and procedural steps, while ISA-95 / IEC 62264 [ISA10] addresses operations management and event contexts; together these support `Recipe`, `DispatchInstruction` and `WorkInstruction`.

Alerting and Alarm Management. *Protocols and standards for alerts, alarm rules, and their severities and urgencies as explanations of critical time series segments.*
The OASIS Common Alerting Protocol (CAP) [OAS10] offers a standard format for exchanging emergency alerts and warnings, with fields for severity, urgency, certainty and recommended response types. OPC UA Alarms & Conditions [OPC22] provides industrial alarm semantics and state handling, while ISA-18.2 and IEC 62682 [Int16b; Int22] provide a lifecycle framework for alarm system management, including the definition of alarm rules, severity and priority (urgency) levels, operator guidance and performance monitoring. Together these standards support the representation of `Alert`, `AlarmRule`, `Urgency`, `Acknowledgment`, `ResetOrClear`, `ThresholdBreach` and `Guideline` as Scenario Knowledge that explains anomalous or critical time series segments.

Formal Knowledge and Documentation. *Standards for expressing mathematical formulas, physical laws, logical rules, and procedural documentation.*
MathML [Car26] provides structured representations of mathematical expressions and operators, while model specification languages such as Modelica [Fri04] enable the formal description of dynamic systems; together these standards support concepts such as `MathExpression`, `AlgebraicEquation`, `PolynomialEquation`, `ExponentialEquation`, `LogarithmicEquation`, `TrigonometricEquation`,

`IntegralEquation` and `DifferentialEquation` for capturing system dynamics in time series contexts. For declarative rules and constraints, RuleML [BPS10] defines logic rules and conditions, supporting concepts such as `Rule`, `Constraint` and `Condition`. Standards like ISO 9001 [Int15] for quality management outline procedures and compliance requirements that can contextualize raw data and generated knowledge. Building on ISO 9001, ISO 10013 [Int21] offers detailed guidance on the preparation and management of quality management documentation, including procedures, work instructions, control plans and related maintenance or safety documentation. Derived concepts include `Documentation`, `Guideline`, `WorkInstruction`, `ControlPlan`, `MaintenancePlan` and `SafetyProcedure`, which can be used to explain why certain operating modes or parameter changes appear in the time series.

Failure and Risk Analysis. *Sources for systematic failure and risk analysis that explain causes, defects, and severities of anomalies.*
AIAG/VDA FMEA [Klu20; AV19] and IEC 60812 [Bif+24; Int18] define standardized notions of failure modes, causes, defect types and severity rankings, while ISO 13381-1 [ISO15] addresses prognostics and health management for machinery, including remaining useful life estimation and condition-based maintenance. Together these standards provide a normative foundation for representing `CausalFactor`, `DefectType`, `Severity`, `Assumption`, `NonConformance`, `RUL` and `Certainty` that are essential for explaining failures and irregularities observed in time series data.

Socio-Behavioral and Organizational Context. *Project and organizational standards that relate internal and external factors to temporal variations.*
Events such as resource allocation changes or new policy implementations might align with observed variations in time series data. Frameworks such as Project Management Institute (PMI) [Les06] standards or PRINCE2 (PRojects IN Controlled Environments) [Ben12] provide tools for tracking project milestones, resource usage and progress, which can be connected to fluctuations in productivity and project-related `Milestone` events. Socio-behavioral contexts including customer feedback and user satisfaction metrics can also be important for understanding specific time series trends or anomalies, justifying concepts such

as `BusinessEvent` that provide explanatory context for observed temporal patterns.

Validation and Reasoning. *Concepts supporting hypothesis testing, evidence management, and validation of derived knowledge.*
The analysis and interpretation of time series data often involves iterative reasoning processes where hypotheses are formulated, evidence is gathered, and conclusions are drawn or revised. The W3C PROV-O ontology [Leb+13] provides foundational concepts for tracking derivation and influence relations, including invalidation of previously asserted information, supporting reasoning about how knowledge was obtained and evolved. Toulmin's argumentation model [Tou03] defines a structured schema for reasoning with concepts such as claims, evidence (data), warrants and rebuttals, which align with candidate concepts like `Evidence`, `Justification` and `Conclusion`. The Information Artifact Ontology (IAO) [SC15] from the OBO Foundry explicitly defines terms including *hypothesis* and *conclusion*, providing direct alignment with proposed concepts `Hypothesis` and `Conclusion`. ISO 9001 [Int15] quality management principles define verification and validation processes along with approval workflows, motivating concepts such as `ValidationEvent`, `ValidationStatus`, `ValidationOutcome`, `Approval` and `Rejection`. Knowledge lifecycle actions including `Revision` and `Retraction` enable tracking of how interpretations evolve over time, while derived knowledge can be represented through `Inference`, `InferredFact` and `Explanation`. The concept `Contradiction` captures identified inconsistencies that require resolution. These candidate concepts are essential for maintaining transparency and traceability in automated and human-guided time series analysis workflows.

Table A.1 presents an overview of initially identified classes together with the mentioned sources determined as relevant, to infer corresponding examples of categorical classes.

3.6 Taxonomy

Creating a taxonomy and finally a vocabulary for time series knowledge demands multiple contrasting requirements. Although the taxonomy should provide comprehensive coverage encompassing all aspects from basic statistical measures to results from advanced analytical models, it should remain domain-agnostic and adaptable to individual scenarios. Especially in the case of relevant Scenario Knowledge there is a clear trade-off between completeness and generalization as the vocabulary must be applicable across diverse contexts without requiring significant adjustments and on the other hand balance complexity to simplify the selection of appropriate terms. Further difficulties include maintaining the flexibility and scalability needed to adapt to emerging techniques in the field of time series analytics, the semantic representation of hierarchical structures and a consideration of contextually overlapping concepts.

In addition to these context-specific challenges, we addressed common requirements aligned with best practices in the domain of Semantic Web. Examples include a standardized interoperability via machine-readable formats such as RDF, the provisioning of shared, publicly available resources and the reuse of established solutions.

Our taxonomy for time series knowledge was derived through the methodology of Nickerson et al. [NVM13]. We chose this methodology because of its dynamic, iterative process, which follows both empirical-to-conceptual and conceptual-to-empirical procedures. In alignment with this approach, our initial structured elicitation provided a first scope of candidates by collecting information from literature as well as expertise from industrial and research projects. Identified classes and concepts served as input for a categorization based on entropy level and semantic meaning. The first category for instance contrasts informal notes with concrete facts, while semantic opposites include anomalies and patterns. We then identified more application-specific state-of-the-art sources to assemble diverse validation subsets, which were used to test coverage and robustness and to identify gaps. We addressed these gaps through iterative refinement until the subcategories became too specific or redundant. Finally, we organized the identified knowledge into a hierarchy to create the final taxonomy. An overview

Step	Corresponding step in Nickerson et al.	Goal
Brainstorming, Literature analysis	Conceptual-to-empirical	Scope, Initial set
Elicitation from research/industry projects	Empirical-to-conceptual	Diversify information
Define classes	Define/Refine dimensions, Classification	Mutually Exclusive, Collectively Exhaustive check
Build evaluation subsets from SotA	Empirical iterations, Evaluate ending conditions	Test coverage
Gap analysis	Iterative refinement to stability/coverage	Restructuring and renaming
Hierarchical structuring	Presentation of taxonomy	Consistent with requirements

Table 3.2: Alignment between our taxonomy-development procedure and the method of Nickerson et al. [NVM13]

of the individual steps together with an alignment of the utilized methodology can be found in Table 3.2.

3.6.1 Data Knowledge Taxonomy

The development of a comprehensive taxonomy for Data Knowledge is inherently challenging due to the diverse and dynamic nature of such knowledge (cf. Section 3.1). The aim is to establish a flexible and extendable framework that can evolve over time and accommodate emerging forms of knowledge. It is important to emphasize that this taxonomy focuses on the organization and classification of knowledge derived from time series data, rather than the methods or algorithms used to generate such knowledge. One important aspect is that a single generation method might produce knowledge of multiple, different categories. Vocabularies serving a categorization of generation methods are for instance defined in the scope of the Machine Learning Sailor Ontology [DYD24]. While methods for analysis may vary significantly, the types of knowledge they produce often overlap or align within broader conceptual categories. For example, classification methods, regardless of their specific approaches, generally assign inputs to predefined classes. Similarly, knowledge derived from different methodological categories may overlap; for instance, both clustering and classification methods can

produce group assignments, although belonging to different analysis paradigms. As a starting point for this categorization we used the initial classification scheme defined through Sections 3.2-3.4.

At the top, we basically separate predictive outcomes from data characteristics, relationships among segments and concrete model artifacts. The prediction branch distinguishes data-level and annotation-level predictions, where data-level predictions represent knowledge corresponding to newly generated data rather than only an enrichment of available data by new insights. Examples in this regard are results from forecasting or imputation. In contrast, knowledge belonging to the category of annotation-level predictions comprises inferred events such as anomalies or class assignments. We deliberately define both the class identifiers of clusterings and classifications to be represented by a single class, even though one is unsupervised and the other supervised. By normalizing outputs to a shared identifier, we enable method-agnostic consumption in downstream workflows, straightforward comparisons and alignments between approaches, while methodological differences remain visible through their respective provenance information.

Data characteristics are organized into coherent families that mirror similar semantics when describing time series. The underlying purpose is to allow for comparisons and queries targeting groups of knowledge describing specific properties like trends. This organization supports focused feature engineering and interpretable reporting, allowing to address questions like “Which dispersion measures change between regimes?”. Concrete scenarios for the exploitation of this categorization are discussed in Section 5.3.1. A separate branch of the taxonomy captures relationships among segments, including measures of relatedness, groupings of similar segments and representatives that summarize group behaviors. Elements of these categories comprise pairwise distances, motifs and shapelets. Finally, as a last main category, we include classes to describe model instances as artifacts to cover knowledge of trained models or related schemas such as ONNX [BLZ+19].

3.6.2 Scenario Knowledge Taxonomy

Creating a taxonomy for Scenario Knowledge is a complex task due to the vast range of domain- and application-specific contexts. Nevertheless, establishing a domain-agnostic foundation is essential for developing a vocabulary that is reusable and easily extendable for concrete scenarios, while preserving its general structure. The aim is thus to define broad concepts that can serve as a foundation for more detailed information. For instance, the concept of a machine alarm might include numerous specific subtypes, yet all of these subtypes fall under the general "alarm" category. By structuring knowledge hierarchically in this manner, the taxonomy creates a framework which defines a common ground across various applications.

Similar to Data Knowledge, in the first layer Scenario Knowledge is divided into four main categories. We organize scenario-specific knowledge around the ways practitioners contextualize, annotate and operationalize analyses. The first branch covers explicit annotations attached to data, used for classification and indexing. Examples include labels, keywords or calendar entries. A second branch captures information that provides context at two levels: informal notes and assumptions on the one hand and verified contextual information including concrete rules and facts on the other hand. This branch further formalizes domain logic through mathematical expressions, ranging from logical conditions to algebraic formulations and mainly defines knowledge serving for interpretation and decision-making. A third category describes events and is again divided on entropy-level into events that are known and categorized and unknown, uncategorized events. Examples of the first subcategory include alarms and actions, while the second subcategory comprises unforeseen observations. Finally, we complement the main categories with knowledge corresponding to scenario parameters. This knowledge corresponds to variables that define concrete configurations and settings.

With respect to Data Knowledge this taxonomy is designed to be compatible with analytic outputs. Annotations and rules can reference derived artifacts such as assigned categories, enabling consistent interpretation and action across different methods. By separating annotations, notes, formal expressions and parameters,

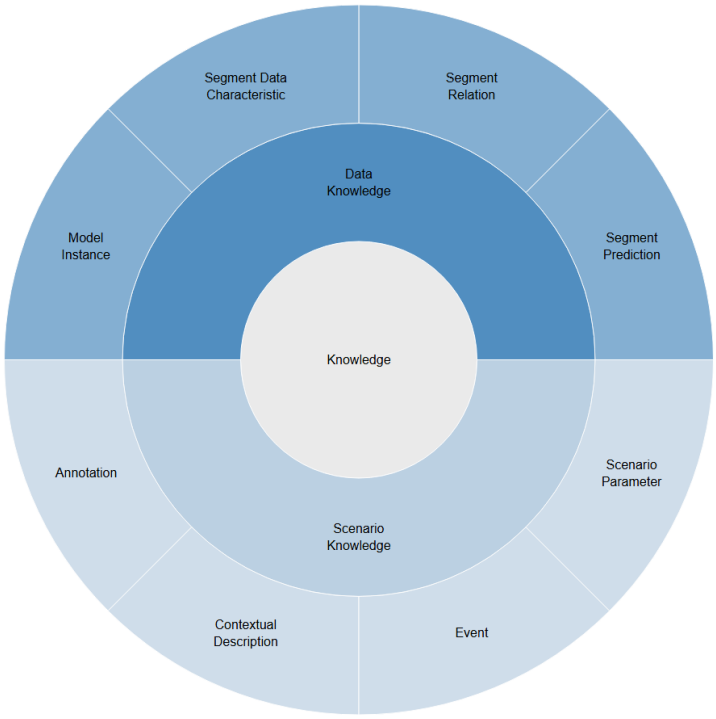


Figure 3.11: Major categories of the two knowledge taxonomies. Both, Data Knowledge and Scenario Knowledge are presented with a hierarchical depth of 2 levels.

we preserve the natural workflow of experts, as they usually label and describe what happens, validate the associated process logic and fix the variables that define or influence a scenario. Figure 3.11 displays the main categories of both taxonomies up to a hierarchical depth of 2. A description of all classes is referenced in the final vocabulary specified in the following Section 3.6.3.

3.6.3 Vocabularies

Based on the defined taxonomies we introduce two machine-readable, interoperable vocabularies for representing time series knowledge: Data Knowledge, i.e., knowledge derived from analytics over time series segments and Scenario Knowledge, i.e., contextual, annotative and parametric knowledge that can be associated with such data. Both vocabularies are linked to the Semantic Time Series (semts)

namespace in order to be integrated with the equally-named ontology presented in the upcoming Chapter 4. The vocabularies describe instances of class *KnowledgeConcept* and extend the SKOS [MP07] framework for a consistent, structured knowledge representation and to reflect hierarchical relations. Each term is equipped with human-readable metadata via skos:prefLabel and skos:definition, enabling documentation and intuitive labeling. The top-level nodes *DataKnowledge* and *ScenarioKnowledge* reference a common parental node *Knowledge*, making both vocabularies siblings within a broader knowledge space.

In addition to *skos:Concept*, *skos:prefLabel*, and *skos:definition*, the vocabularies utilize *skos:Collection*, along with *skos:broader* and *skos:member*, to establish hierarchical relationships. To enable the definition of knowledge regardless of whether it corresponds to *skos:Concept* or *skos:Collection*, we extend the union of both classes by *KnowledgeConcept*.

From RDF, RDFS and OWL, the vocabularies further use *rdf:type* to type individuals, *owl:Class* and *owl:NamedIndividual* for metamodeling and *rdfs:subClassOf* for inheritance.

Apart from a wide compatibility, we primarily decided to use SKOS to encode hierarchical child-parent links from the taxonomy through *skos:broader*. Hierarchies are expressed exclusively via *skos:broader* to simplify the structure of both vocabularies, since the inverse relation *skos:narrower* can be inferred via reasoning tools (cf. Section 2.2.3).

Conceptual groupings are expressed using *skos:Collection* and *skos:member* to reflect collections of analytical results and contextual knowledge. As an example, *Clustering* aggregates instances of class *Cluster*, each of which can carry both a representative (*SegmentGroupRepresentative*) and a category identifier (*InferredClassID*). Mirroring relations via hierarchies and groupings consequently allows for an easier validation of concrete knowledge relations, ensuring that outputs of analytical pipelines remain consistent (cf. Section 4.4.4).

To support reproducibility, third-party reuse, and community-driven extension, both vocabularies are published and maintained as open, versioned resources under persistent w3id Identifiers (W3C Permanent Identifier) and a CC BY 4.0

license. We encourage researchers and practitioners to adopt, validate, or extend the vocabularies. The resources are accessible at:

- **Data Knowledge Vocabulary (semts-dv):**
  https://w3id.org/semts/vocabulary/data-knowledge#
- **Scenario Knowledge Vocabulary (semts-sv):**
  https://w3id.org/semts/vocabulary/scenario-knowledge#

3.6.4 Evaluation

While the methodology of Nickerson et al. [NVM13] guided the iterative construction of both taxonomies (cf. Table 3.2), we additionally assessed whether the resulting vocabularies achieve practical coverage and fulfill their intended purpose. In this section, we evaluate the Data Knowledge and Scenario Knowledge vocabularies through two complementary approaches: (i) a systematic coverage analysis against established time series libraries and standards and (ii) a saturation based on independent sources not used during development. Detailed evaluation results including per-item mappings are provided in Appendix B.

Data Knowledge Coverage

To evaluate whether the Data Knowledge vocabulary covers the analytical outputs practitioners encounter in practice, we systematically mapped the documented output types of seven prominent time series libraries to the categories defined and formalized in the vocabulary (cf. Section 3.6).

The seven libraries were selected to maximize complementarity rather than redundancy. Each library contributes a distinct analytical focus: *tsfresh* [Chr+18] for automated feature extraction, *STUMPY* [Law19] for Matrix Profile-based motif and discord discovery, *sktime* [Lön+19] as a comprehensive unified ML framework for time series, *Statsmodels* [SP10] for classical statistical modeling and hypothesis testing, *Tigramite* [Run18] for causal discovery, *lifelines* [Dav19] for survival analysis, and *Darts* [Her+22] for probabilistic forecasting and anomaly detection.

Libraries such as *tslearn* [Tav+20], *AEON* [Mid+24] and *Kats* [Fac21] were excluded because their primary output types are already represented by the selected set; for instance, *tslearn* largely overlaps with *sktime* in clustering and classification outputs, and *Kats* combines feature extraction and forecasting capabilities already covered by *tsfresh* and *Darts*, respectively.

For each library, we reviewed the documentation and identified the principal output types produced by core analysis functions. Since many individual functions within a library yield outputs of the same semantic category (e.g., multiple clustering algorithms all return cluster assignments), we report semantically distinct output categories per library rather than enumerating every individual function. This approach ensures that the evaluation reflects most of the output types without redundant entries. The complete set of 47 identified output categories together with their vocabulary classifications is provided in Table B.2. The central finding is that all identified output categories were classifiable under existing taxonomy terms. No output required the introduction of a new category.

However, four output categories required interpretive judgment during classification: feature relevance scores from *tsfresh* (classified as inferential statistics), snippets from *STUMPY* (classified as instance-specific descriptions), regression residuals from *Statsmodels* (classified as structural profiles) and backtesting metrics from *Darts* (classified as model artifacts). While all four cases are mappable under reasonable interpretation, we felt that some outputs did not align unambiguously with a single category. Such observations are to be considered during future refinements of the vocabulary definitions to improve clarity in borderline situations.

As a secondary verification, we also examined which taxonomy categories are used by the mapped outputs (cf. Table B.3). All categories were addressed by at least one library, confirming that no category remains unused or superfluous. The weakest category is *General Dataset Properties*, which receives only indirect coverage through counting features in *tsfresh*. This outcome is expected, as dataset-level metadata such as row counts, missing rates and sampling frequencies are typically produced by general-purpose data handling libraries rather than dedicated analysis tools.

To further strengthen the coverage argument, we examined ten additional time series libraries that were not used during taxonomy development. The purpose was to determine whether any of their output types would require the introduction of new categories. These sources were selected to be diverse in analytical focus. For each library, we identified main output types from available documentation and classified them using the existing taxonomy. Again, the result revealed that none of the ten independent sources required the introduction of a new category. Table B.1 summarizes the results. While a more fine-grained mapping is possible, we chose terms from higher-level layers to group similar knowledge concepts, since this is already sufficient to prove coverage.

Scenario Knowledge Coverage

The Scenario Knowledge vocabulary poses a different evaluation challenge than Data Knowledge, because its categories do mostly not correspond to algorithmic outputs but to contextual information from diverse domains and standards. We therefore evaluate cross-domain applicability to test for coverage. For each of five representative application domains, we identified the contextual information a practitioner would need to capture when working with time series data and classified each item using the vocabulary. The five domains were selected to span distinct operational contexts: manufacturing as the dominant domain of the source standards, healthcare and environmental monitoring as domains with strong sensing infrastructure requirements, finance as a domain with fundamentally different operational semantics, and smart building management as a domain combining infrastructure monitoring with scheduling and compliance. The complete per-item mappings are provided in Table B.4. Across 62 contextual information items spanning five domains, 60 (96.8%) are directly classifiable under specific vocabulary terms, and the remaining two are classifiable at the parent level. Both partial matches occur in the finance domain, where a trading halt imposed by an exchange is classifiable as *ExternalEvent*, but lacks a more specific term for operational suspensions outside the maintenance context. A market regime description can be classified as *Status* or *OperatingMode*, although both definitions lean toward physical systems. In the healthcare domain,

concepts such as medication administration, diagnosis confirmation and clinical protocols map naturally to *Action*, *Confirmation* and *Rule*, demonstrating that the vocabulary generalizes beyond industrial contexts. Similarly, in environmental monitoring, sensor calibration events, communication losses and data quality flags are directly captured by *CalibrationEvent*, *CommunicationLoss* and *QualityAnnotation*, respectively. However, given the sample size in comparison to the number of potential domain-specific terms, this coverage rate should be interpreted as indicative rather than statistically significant.

Chapter Summary

This chapter addressed RQ 1 by defining what constitutes time series knowledge and how it can be structured. We introduced a bidirectional, continuous model of knowledge significance as an alternative to the DIKW hierarchy and distinguished two complementary knowledge types: *Data Knowledge*, representing insights derived from analytical processes, and *Scenario Knowledge*, capturing contextual information from domain expertise and operational standards. Through a systematic, iterative taxonomy development, we organised these types into hierarchical categories, from general dataset properties and time series characteristics through task-driven analysis results to scenario-specific contextual information. The resulting taxonomies were formalised as two machine-readable SKOS vocabularies, published under persistent identifiers, and evaluated through coverage analysis against seven prominent time series libraries and cross-domain applicability testing across five representative domains, partially confirming practical coverage.

What follows. Having defined what constitutes time series knowledge, Chapter 4 introduces the SemTS ontology as a semantic model that provides a classification by defining how knowledge is represented, linked to time series segments, and propagated through analytical workflows.

4 SemTS - The Semantic Time Series Ontology

Across application domains that rely on temporal data, a recurring problem exists in which the insights produced by analytical workflows are consumed once and rarely preserved in a form that supports reuse. Ontologies developed for the machine learning lifecycle [Pub+18; DYD24; VS10; PSD13] capture how models are configured, trained, and evaluated. Nevertheless, the actual outcomes of those models, specifically the knowledge they generate about the data, remain without a principled representation scheme. Findings such as detected irregularities, temporal patterns, or structural groupings lack a shared vocabulary and formal organization, which makes it difficult to leverage them in downstream tasks or across organizational boundaries.

This chapter introduces the *Semantic Time Series Ontology (SemTS)* as a response to this gap and as a central contribution to RQ 2. The ontology treats time series characteristics as knowledge entities with clearly defined semantics, enabling them to be queried, validated, and propagated. Complementary vocabularies (Section 3.6.3) supply the fine-grained conceptual distinctions necessary to differentiate knowledge across multiple levels of abstraction. Beyond data-driven findings, SemTS also accommodates contextual information originating from domain expertise or operational constraints, thereby enriching the analytical context in which time series data is interpreted.

Subsequent sections demonstrate the practical value of SemTS by presenting its design rationale, its architecture, and concrete examples showing how classified time series knowledge can be queried, validated, and reused. By combining inferred insights with predefined facts under a common formal framework, SemTS

establishes a foundation for improved accessibility and exploitation of temporal data knowledge.

Chapter Outline

This chapter discusses SemTS as one of the core contributions of this work. The chapter starts with Section 4.1 to describe the motivation behind this ontology. Next, it summarizes related works in Section 4.2 and describes the design methodology that has been used to create SemTS including identified requirements, the integration with external schemas and associated challenges in Section 4.3 . After a comprehensive discussion on the individual components of SemTS in Section 4.4, technical details are listed in Section 4.5, before concluding with a syntactical and semantic evaluation of the ontology in Section 4.6. The content presented in this section is largely based on the corresponding publications [Gra+25a; Gra+26].

4.1 Motivation

Time series data supports decision-making across diverse domains, including manufacturing [WYZ+22], finance [SGO20], and clinical practice [PGP+21]. The demand for efficient temporal data handling is evidenced by the continued rise of time series database management systems in popularity rankings [DB-25]. These systems optimize storage and retrieval operations that feed into analytical pipelines [JPT17], yet their scope ends at data provisioning. Once an analysis pipeline has concluded and produced results, there is typically no mechanism to retain those results in a structured, accessible form for later use.

The consequence is a persistent gap in knowledge transfer [Von+23]. When a new analysis begins without access to previously established findings, whether from earlier runs on the same data or from expert assessments, the opportunity to build upon prior knowledge is lost. In certain cases, this lack of continuity even introduces systematic errors into subsequent work.

Figure 4.1 demonstrates this problem through a concrete case. A Fourier regression is fitted to a periodic signal that contains a known anomaly. Two variants are shown: one that accounts for the anomaly by excluding it from the regres-

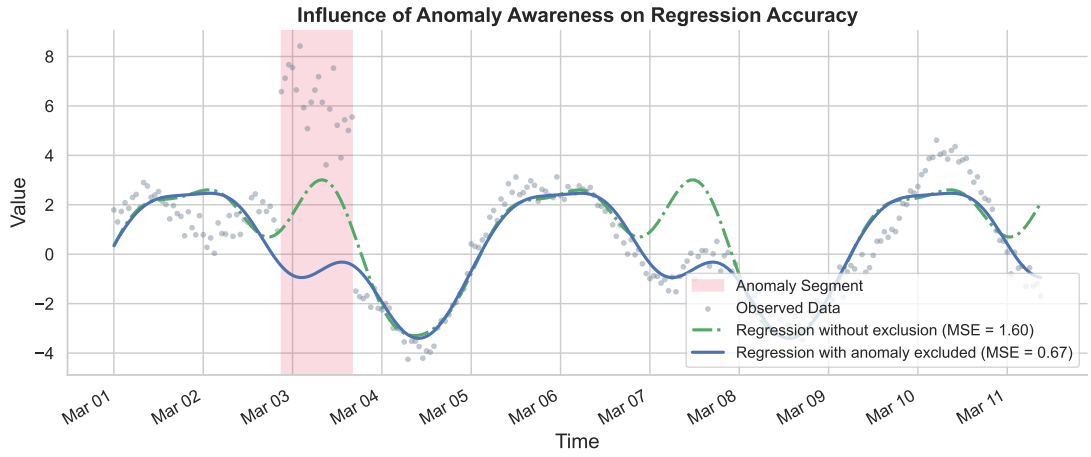


Figure 4.1: Impact of knowledge reuse: Example of two Fourier regression models, representing outcomes with knowledge integration (blue) and without it (green). Excluding the anomaly reduces the Mean Squared Error (MSE) by 58% from 1.60 to 0.67. Both, the time series and the anomaly represent knowledge segments.

sion (blue) and one that treats the entire signal uniformly (green). The variant that fails to incorporate the previously identified anomaly absorbs the disturbance into its model of the regular signal dynamics, yielding a substantially less accurate representation.

This example, though simple, points to a broader pattern. Once an anomaly has been detected and validated, possibly with additional expert annotations, it represents a reusable asset. The same holds for pattern occurrences, cluster memberships, or verified operating states. If such findings are used only at the moment of their creation and then abandoned, every future analysis that could benefit from them must either rediscover them or proceed without them.

Organizations often pose closely related questions to the same data at different points in time. The outcomes of earlier inquiries, once reviewed by analysts and domain specialists, frequently disappear from organizational memory because no adequate mechanism exists to preserve them in a retrievable form [LC19; Lev11]. Staff turnover accelerates this loss of institutional knowledge. What typically remains from an analytical workflow are the trained models and their associated performance metrics [ISB22], though these artifacts do not necessarily

capture what those models revealed about the data they processed. Reproducing the same findings at a later date therefore wastes both computation and human attention.

These observations lead to the research question that shapes the present contribution, as stated in [Gra+26]: *"How to semantically represent analytical and scenario-specific insights associated with time series data for reusability and future accessibility?"* This question targets the formal capture of condensed, valuable information in the sense of the information model introduced in Section 3.1. Because the outputs of different analytical tasks vary widely in type, granularity, and interpretation, a generic metadata store with loosely defined semantics cannot adequately address this question. The semantic schemas that currently exist in the analytics domain (cf. Section 4.2) were designed to describe datasets and model configurations rather than to systematically catalogue the insights that analytics produce.

4.2 Related Work

Data analytics lifecycles, as captured by established process models [Kut+23], have motivated the creation of ontologies and controlled vocabularies that formalize the entities relevant to data handling and analysis. This section surveys these contributions and identifies the gap that SemTS addresses.

At the level of data management and sharing, DCAT [Alb+24] standardizes how datasets and their access points are described across catalogs. PROV-O [Leb+13] provides a general-purpose model for documenting the lineage of entities and the activities that produced them. For analytical consumption of structured observations, the Data Cube Vocabulary [CR14] offers a framework tailored to multidimensional statistical data.

A separate family of ontologies targets the processes and models used in data science. OntoDM [PSD13] formalizes data mining tasks together with their inputs and outputs. DMOP [Kee+14] focuses on operator selection and algorithm performance. Exposé [VS10] models experimental setups for machine learning, covering parameter choices and result recording. Provenance of trained models

Approach	Data Management	Model Specifics	Quality & Evaluation	Pipeline Definition	Scenario Context	Knowledge Management
DMOP [Kee+14]	Partially	Yes	Yes	Partially	Partially	No
MEX [EMN+15]	Partially	Partially	Yes	No	Partially	No
OntoDM [PSD13]	Partially	Yes	Yes	Yes	Partially	No
Exposé [VS10]	Partially	Partially	Yes	Yes	Partially	No
MLS [Mac18]	Yes	Yes	Yes	Partially	Partially	No
MLSO [DYD24]	Yes	Yes	Yes	Partially	Partially	No
SOSA [JHC+19]	Yes	No	No	No	Partially	No
SemML [ZSG+21]	Yes	Partially	Partially	Yes	Yes	No
IoT-Stream [EER+20]	Yes	No	No	Partially	Yes	No

Table 4.1: Coverage of approaches from the related work section.

specifically is addressed by both the MEX vocabulary [EMN+15], which logs experiment metadata, and PROV-ML [SAL+19], a specialization of PROV-O for machine learning artifacts.

Because these ontologies each emphasize different lifecycle phases and trade coverage against depth in domain-specific ways, ML-Schema [Pub+18] was developed as a unifying integration, spawning a W3C community group [Mac18]. MLSO [DYD24] extends this integration further by incorporating data management vocabularies and classification taxonomies, thereby spanning the full arc from dataset description through model evaluation. Despite this comprehensive scope, a notable limitation persists. None of these approaches provides an explicit model for the knowledge that analytical processes generate about data. Structural time series properties, recurring patterns, inter-series relationships, and other data-level findings remain outside their representational scope.

In the sensor and IoT space, SOSA [JHC+19] and IoT-Stream [EER+20] do support event annotations on temporal data streams, but their focus lies with observation metadata rather than with a systematic classification of analytical findings or the integration of domain knowledge. SemML [ZSG+21; ZZZ+22] takes a complementary approach by incorporating prior domain information into machine learning pipeline orchestration. Yet its primary objective is guiding the execution of analysis workflows, not cataloguing the insights those workflows produce. Table 4.1 maps the discussed approaches to six functional dimensions,

making their respective coverage visible at a glance.

4.3 Ontology Design

The goal of SemTS is to provide formal structure for both preexisting and analytically derived knowledge about time series data. Its conceptual roots are described in [Gra+25a] and build upon the idea of using knowledge graphs to enable query-driven exploration of time series information [Gra+23].

The organizing principle of SemTS is a segment-based knowledge representation. As discussed in Chapter 3, what qualifies as characteristic knowledge is deliberately broad, encompassing statistical descriptors [Chr+18], structural phenomena such as anomalies [SWP22] or motifs [MIM+19], and expert-supplied prior information. In the example from Figure 4.1, the red-shaded region constitutes such a segment, which delineates a temporal interval and binds it to a formal description of the anomaly occurring within that interval.

The role of SemTS can be understood as providing a *queryable, semantic layer that indexes the facts and insights affiliated with time series data*, making them accessible for retrieval, reasoning, and reuse in subsequent analytical stages.

4.3.1 Requirements and Design Methodology

The requirements that guided the development of SemTS originate from two primary sources. First, from industrial settings [Sol24] where large volumes of time-stamped process data are produced. Second, concrete needs identified in collaborative research and industry projects, a selection of which is discussed in Section 5.1. Although the ontology already satisfies these requirements in its current form, they are stated here in the conventional "should" phrasing for clarity.

The requirements can be organized along four dimensions:

Scope and Generality. SemTS should be independent of particular deployment environments, tool ecosystems, or data platforms. The classification and de-

scription of time series characteristics together with associated knowledge should remain applicable across heterogeneous setups. Knowledge values of arbitrary complexity, from scalar numbers to deeply nested structures, should be representable without restricting the ontology to specific data formats.

Knowledge Organization and Coverage. A hierarchical concept system should distinguish analytical outcomes from methodological configurations and from domain-level or scenario-level facts. Knowledge should be assignable both to complete time series and to subsequences, including those with reduced dimensionality. Coverage of segment-level properties like temporal extent and number of dimensions should be thorough enough to handle a wide variety of time series configurations. Segments representing data that does not yet exist in the dataset (for instance, forecast horizons) require explicit support. Nested groupings of knowledge items should be possible to represent composite analytical results. Quality descriptors should allow consumers to judge the reliability of any given piece of knowledge.

Integration and Provenance. A key concern is the ability to combine insights produced by analytical methods with information supplied by domain experts, despite the diversity of formats that different source systems impose. Tracing how knowledge was derived, from which data and through which process, is necessary for reproducibility and trust. Where richer descriptions of analytical runs exist in external schemas, the ontology should permit references to them. Querying, inference, and reasoning should operate over the combined body of stored knowledge.

Technical Conformance. The ontology should follow established Semantic Web conventions regarding documentation, publication, and the reuse of existing schemas.

An overview of requirements together with their corresponding solution in SemTS can be found in Appendix C.

Regarding the process by which SemTS was constructed, we adopted an iterative, collaborative development style. Ontology engineers and domain specialists worked in repeated cycles of modeling, review, and validation. A backlog of open questions and modeling decisions was maintained throughout. We drew selectively on established ontology engineering methodologies [The24]: competency questions following OD101 [NM01] principles guided scoping and validation, resource reengineering pathways from NeOn [SGF11] informed the integration of external schemas, pair design sessions inspired by eXtreme Design (XD) [BHP16] supported collaborative modeling choices, and tool recommendations from LOT [Pov+22] shaped the selection of Protégé [Mus15] for authoring, OOPS! [PGS14] and FOOPS! [GCP21] for quality checks, and WIDOCO [Gar17] for documentation.

4.3.2 Integration with External Ontologies

For SemTS we reuse and extend classes and properties from several well-established ontologies. This reuse of external ontologies ensures interoperability with existing standards and facilitates integration with other semantic web resources.

OWL-Time (Time Ontology in OWL) [CL22] provides classes for defining temporal concepts such as `time:Interval` and `time:Instant`, which are used to index time series segments. These classes allow for precise temporal indexing of data points or intervals, enabling the representation of time-based relationships critical to time series analysis. For example, intervals can be used to describe a segment of data that spans a specific time period, while instants can represent discrete data points within that interval.

PROV-O (Provenance Ontology) [Leb+13] is reused to define provenance relationships, allowing SemTS to track the origin, derivation, and transformation of knowledge. Through its integration, concepts such as `prov:Activity`, `prov:Entity`, and related properties like `prov:wasDerivedFrom`, `prov:used` and `prov:generated` enable detailed tracking of the processes and entities involved in generating time series knowledge. This ensures reproducibility, transparency,

and traceability of analytical workflows, facilitating the validation and reuse of knowledge.

DCAT (Data Catalog Vocabulary) [Alb+24] is employed to describe datasets and their distributions, enabling integration with metadata standards for data sharing. Through the reuse of classes like `dcat:Dataset` and `dcat:Distribution`, SemTS can represent time series data segments as datasets, while their metadata and dissemination methods (e.g., via APIs or file downloads) are described as distributions. This integration enhances the ability to catalog, share, and discover datasets in compliant formats.

QUDT (Quantities, Units, Dimensions, and Types Ontology) [QUD11] is utilized to define quantitative and measurement-related concepts in SemTS. It provides classes such as `qudt:Unit` and `qudt:Datatype`, which enable the specification of measurement units (e.g., meters, kilos) and data types associated with values. For instance, a `DataDimension` in SemTS can be assigned a `qudt:Unit` to represent the measurement unit of the dimension. This integration ensures consistent and precise representation of numerical values and their associated units.

MLS (Machine Learning Schema) [Mac18] is reused to extend the expressiveness of the ontology regarding Machine Learning workflows, algorithms, and evaluation metrics. By incorporating classes such as `mls:Algorithm`, `mls:EvaluationMeasure`, `mls:Model`, and `mls:HyperParameter`, SemTS can extend its capability of representing details on analytical tasks.

SKOS (Simple Knowledge Organization System) [MP07] is leveraged to support the organization and representation of knowledge concepts, especially the ones from the SemTS vocabularies (cf. Section 3.6.3). The `skos:Concept` class is reused and extended in SemTS as `KnowledgeConcept`, `KnowledgeGenerationConcept` and `QualityMeasureConcept`. This enables a detailed and standardized classification and semantic description of knowledge entities, their generation methods and their associated quality measures.

By integrating and reusing established ontologies, SemTS increases interoperability while also avoiding duplicating efforts.

4.3.3 Challenges and Limitations

Two fundamental tensions complicate the design of an ontology for time series knowledge. The first concerns the pace at which analytical methodology evolves (cf. Section 3.1). When a newly developed technique yields outputs that do not map naturally onto established knowledge categories, the ontology must either absorb the novelty or risk becoming incomplete. Maintaining a balance between stability and adaptability over time is thus a recurring design challenge.

The second tension arises from the inherent ambiguity of what constitutes "knowledge" in this context. No universally accepted boundary separates a raw metric from a meaningful insight, which means that the ontology's classes must remain general enough to encompass different use-case interpretations while still providing enough semantic specificity for meaningful querying and reasoning.

These tensions manifest concretely in the representation of composite analytical outputs. Gaussian process regression [GDB22] on time series data, for instance, may yield a kernel composition whose components each carry hyperparameters describing structural characteristics at different scales. The resulting knowledge spans multiple levels of granularity within a single analytical act, and the associated value representations can involve deeply nested data structures that resist a one-size-fits-all encoding.

4.4 Ontology Description

In SemTS, the segment serves as the primary organizing unit. Each segment associates a temporal extent and a set of data dimensions with the knowledge that has been attributed to that portion of the time series. The attributed knowledge may range from simple statistical properties through model-derived predictions to expert-supplied contextual facts. Because both the temporal extent and the dimensionality of a segment are configurable, the same formalism handles scenarios as diverse as a single observation in a univariate series and a collection of multivariate segments drawn from different data sources.

Three layers of interconnected semantic entities constitute the conceptual archi-

tecture of SemTS. At the bottom lies the *Data Management* layer, which governs the structural representation of time series inputs and the concrete values carried by knowledge instances. Above it, the *Knowledge Classification* layer supplies a taxonomy that distinguishes different kinds of knowledge at a coarse level. The top layer, *Knowledge Generation*, formalizes how new knowledge is generated, whether through analytical computation or reasoning over existing knowledge, and how provenance is preserved in the process.

Figure 4.2 depicts the classes that populate these three layers. The subsections that follow describe each layer in detail, using this figure as a reference point.

4.4.1 Data Management

The class `TimeSeriesSegment` anchors the data management layer. Four facets define a segment: a dataset reference (inherited from `dcat:Dataset`), a temporal index, one or more dimensions, and the knowledge attributed to it. Temporal indexing uses `time:TemporalEntity` from OWL-Time, accommodating both contiguous intervals of multiple time steps and single instants. The class `DataDimension` describes each data dimension through a measurement unit (`qudt:Unit`) and a datatype (`qudt:Datatype`), both sourced from QUDT.

Not all segments correspond to already observed data. Forecast outputs, for example, refer to future time regions that have no counterpart in the current dataset. To make this distinction explicit, SemTS provides the subclass `ArtificialTimeSeriesSegment`. Structural relationships between segments, such as a daily segment being contained within a weekly one, are captured through the transitive property `isPartOf`, whose transitivity enables reasoners to derive containment chains automatically. The overall design choice to allow segments at any temporal granularity means that a single SemTS knowledge graph can simultaneously accommodate broad overviews and narrow, focused inspections of the same underlying data.

Knowledge values follow a parallel representational pattern, with each value carrying a unit and a datatype from QUDT. Where the value is a simple scalar or string, the class `EmbeddedValue` provides an inline representation. For richer

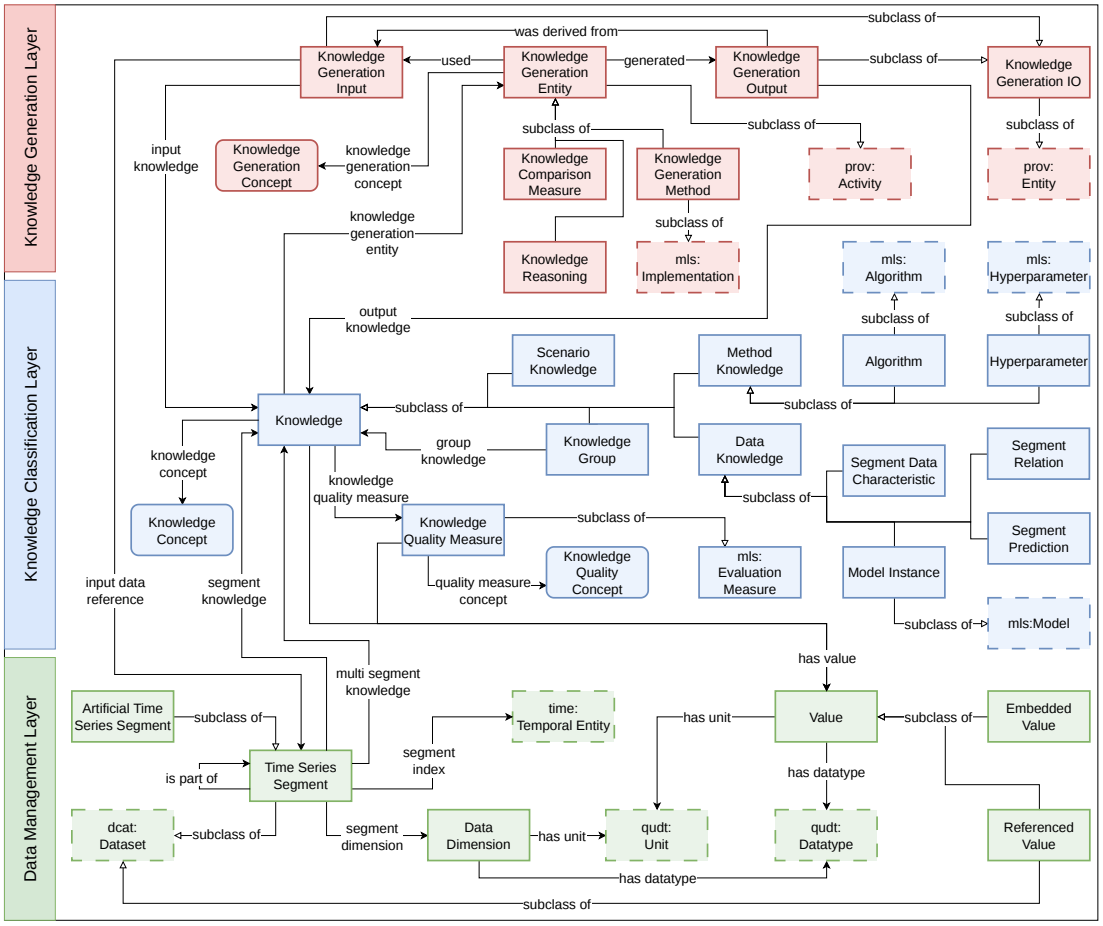


Figure 4.2: Overview of SemTS organized by its three conceptual layers: Data Management (green) handles segment structure and value representation, Knowledge Classification (blue) provides the coarse taxonomy of insight types, and Knowledge Generation (red) captures provenance and derivation processes. External schemas are indicated by namespace prefixes and dotted borders; classes rooted in SKOS appear with rounded corners to signal their role as extensible concept anchors. The image shows an updated representation of the architecture diagram from [Gra+26]

.

structures (multidimensional arrays, compound annotations, or external file references), `ReferencedValue` acts as a typed pointer. Because `ReferencedValue` inherits from `dcat:Dataset`, it can describe arbitrarily complex payloads stored outside the knowledge graph itself, such as bounding-box annotations over video frames.

4.4.2 Knowledge Management

Rather than attempting to enumerate every possible type of time series insight, SemTS partitions the knowledge space into three broad categories and delegates finer distinctions to the vocabulary layer. The three categories are `DataKnowledge`, `ScenarioKnowledge`, and `MethodKnowledge`. A fourth class, `KnowledgeGroup`, represents collections of knowledge items and is itself a subclass of `Knowledge`, enabling recursive nesting for composite results. Any knowledge item may carry one or more instances of `KnowledgeQualityMeasure`, which express either a qualitative or quantitative assessment of that item's reliability together with a concept identifying the type of assessment.

Data Knowledge. This category captures insights whose origin lies in the data itself or in analytical operations applied to it. Its subclasses serve as a coarse index rather than a detailed typology: `SegmentDataCharacteristic` covers per-segment features (e.g., a segment's variance), `SegmentRelation` captures inter-segment properties like distances or correlation coefficients, `SegmentPrediction` pertains to outputs or descriptors of predictive processes, and `ModelInstance` records concrete model specifications or trained parameter sets. This categorical split is mirrored in the connecting properties: `segmentKnowledge` links knowledge to a single segment, while `multiSegmentKnowledge` applies to relationships that involve two or more segments.

Scenario Knowledge. Verified or authoritative statements about the time series or its environment fall into this category. Class labels assigned by experts, textual annotations, event markers, mathematical expressions describing expected signal behaviour, and operational constraints all qualify. Although Scenario Knowledge

most commonly represents prior information contributed by domain specialists, it also serves as a container for facts that confirm or contextualize data-driven findings after human review. Its semantics align with the characterization of scenario-level knowledge developed in Chapter 3.

Method Knowledge. This category occupies a conceptual middle ground. It records analytical method configurations that have been validated for specific data or process contexts, essentially representing proven methodological experience bound to a concrete scenario. Its subclasses `Algorithm` and `Hyperparameter` extend the corresponding ML-Schema classes to allow linkage with richer metainformation where needed. A typical instance would be the calibrated parameterization of a model that reflects the physical dynamics of a particular machine. In Section 3.1, we stated that "we propose a definition that primarily distinguishes two complementary types of knowledge - *Data Knowledge* and *Scenario Knowledge*." However, we introduce an additional category, `MethodKnowledge`, as a conceptual design choice, since methodological information can arise from either data-driven inference or expert-driven parameterization rather than from a formal intersection of the two primary types. In the ontological representation, each knowledge instance is classified into exactly one of these three knowledge types based on its primary role.

Knowledge Groups. Analytical results often consist of multiple interrelated components. A clustering run, for instance, does not produce a single item but rather a collection of clusters, each comprising assignments, representatives, and quality indicators. To preserve this compositional structure, `KnowledgeGroup` can contain other knowledge items or groups, forming recursive hierarchies of arbitrary depth (cf. Section 4.4.4).

4.4.3 Knowledge Generation

Classifying knowledge makes it retrievable, but the greater value lies in enabling its onward flow: from one analytical stage to the next, from raw inference to expert-validated insight. Classified knowledge fulfills a dual role in this regard,

serving both as the output of a preceding analysis (available for semantic retrieval) and as potential input to future analyses (cf. Figure 4.1) or reasoning processes.

An important design boundary is that SemTS does not explicitly model dataset transformations like resampling or normalization. Such operations commonly belong to data pipelines in preprocessing stages and are already formalized in schemas like DMOP. In case a particular transformation belongs to an algorithm or model, this information is inherently covered by an associated generation entity described below. Provenance information is nonetheless essential to trace how any given insight was created, which requires representing the generation process itself.

SemTS formalizes generation through three classes. `KnowledgeGenerationInput` captures what enters a generation process, whether that is raw time series data, previously established knowledge, or a mixture. `KnowledgeGenerationOutput` captures what is produced, restricted to knowledge representations. The process that transforms input into output is modeled by `KnowledgeGenerationEntity`, which has three subclasses reflecting the nature of the transformation: comparison-based, reasoning-based, and algorithm-based. The algorithmic subclass (`KnowledgeGenerationMethod`) additionally inherits from `mls:Implementation`, which allows for detailed methodological descriptions via ML-Schema when needed (e.g., to distinguish between multiple configurations or experimental runs).

Provenance tracking builds on PROV-O. All input and output classes inherit from `prov:Entity`, while generation entity classes inherit from `prov:Activity`. This inheritance brings standard provenance properties (`used`, `wasDerivedFrom`, `generated`) into the SemTS framework. Generation entities, like knowledge items themselves, receive typing through dedicated concepts.

The ontology intentionally restricts its output representation to knowledge. However, for analytical tasks whose primary output consists of generated time series data, as is the case for forecasting models, the result can be captured as a `SegmentPrediction` whose value is a `ReferencedValue` pointing to the produced data.

Figure 4.3 illustrates how instances from all three layers connect in practice.

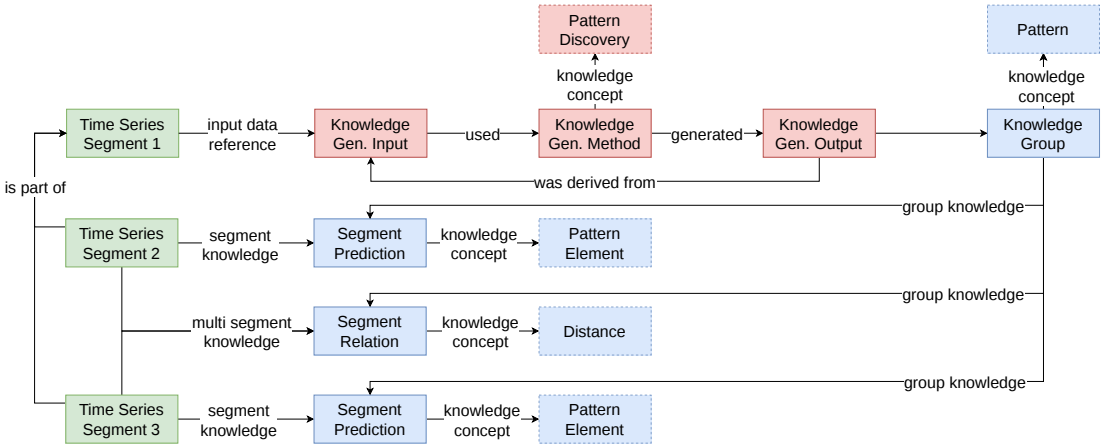


Figure 4.3: A simplified example showing how a pattern discovery process is defined by the three SemTS layers. A single input segment yields two new segments (linked via `isPartOf`) whose associated knowledge items demonstrate the distinction between per-segment properties (`segmentKnowledge`) and relational properties spanning multiple segments (`multiSegmentKnowledge`). Vocabulary concepts annotate each knowledge instance to provide machine-interpretable typing.

The depicted scenario shows a pattern discovery algorithm applied to a single time series segment. The algorithm identifies two occurrences of a pattern, each represented as a new segment declared to be part of the original input segment, reflecting that the discovered elements are temporally contained within the input's bounds.

Aligned with Figure 4.4 further simplified examples of corresponding knowledge representations and associated generation patterns are:

- **Model Training**: Multiple time series segments serve as input to a machine learning activity, producing a trained model whose parameters are stored externally. Quality metrics such as reconstruction error accompany the model to indicate its reliability.
- **Model-based Prediction**: A previously trained model, combined with a new segment as data input, generates a prediction that is categorized according to a predefined taxonomy of knowledge types.

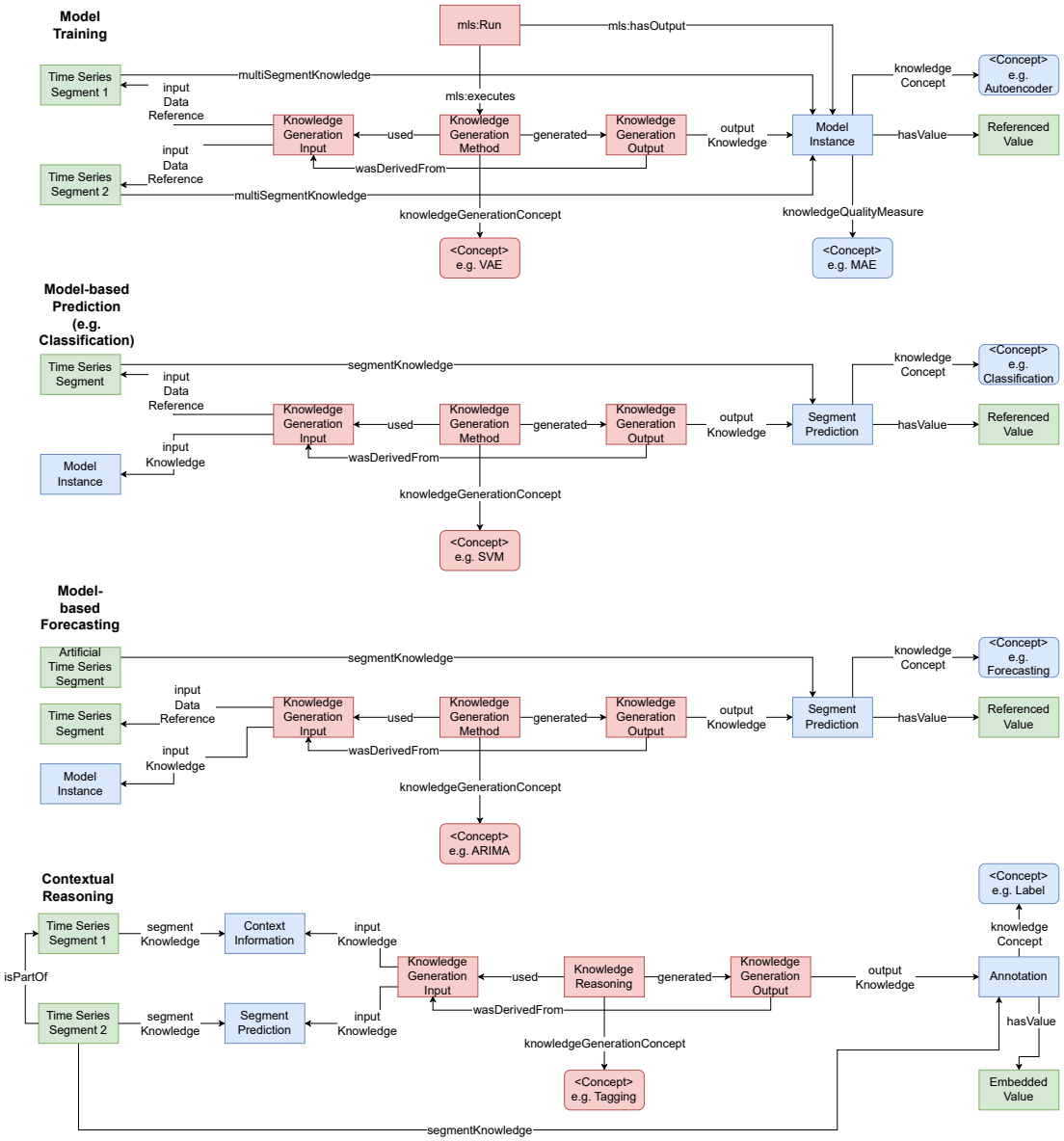


Figure 4.4: Further visually simplified knowledge generation patterns based on SemTS.

- **Model-based Forecasting**: A segment representing predicted future values from a forecast that extends beyond the observed data range.
- **Contextual Reasoning**: Combination of a data-driven prediction with domain-specific contextual information through a reasoning process, which enables the derivation of validated annotations, such as events confirmed by expert rules or domain constraints.

4.4.4 Knowledge Hierarchies and Vocabularies Integration

Composite analytical results demand a principled way to represent nesting. The design decision in SemTS is to couple the recursive nature of `KnowledgeGroup` with the hierarchical structuring facilities of SKOS. This coupling creates a bridge between the deliberately generic ontology classes and the more specific, extensible knowledge types supplied by the vocabularies. Figure 4.5 demonstrates this alignment for a clustering scenario, where data knowledge concepts and knowledge groups correspond to each other at successive levels of the hierarchy, with every knowledge entity consistently typed by its vocabulary concept.

The class `skos:Collection` plays a complementary role here: it permits the specification of admissible group compositions, thereby enabling validation of whether a given knowledge group exhibits the expected internal structure. This enriched semantic expressiveness translates directly into more powerful querying and constraint checking. Concretely, `KnowledgeConcept` extends the union of `skos:Concept` and `skos:Collection`, allowing each `KnowledgeGroup` to be annotated with a collection or concept that specifies its intended role and admissible members. The SKOS hierarchy provides the semantic backbone, where a group aligned to a higher-level concept can have a structured expectation about which subordinate concepts may appear inside such that membership can be checked or inferred via properties including `skos:broader` and `skos:member`.

In the presented example, one could validate if a `KnowledgeGroup` corresponding to a clustering indeed has at least one member of type cluster. Beyond clustering, this alignment generalizes to multiple analysis families. For classification, a group aligned to the `Classification` collection can validate outputs by the existence

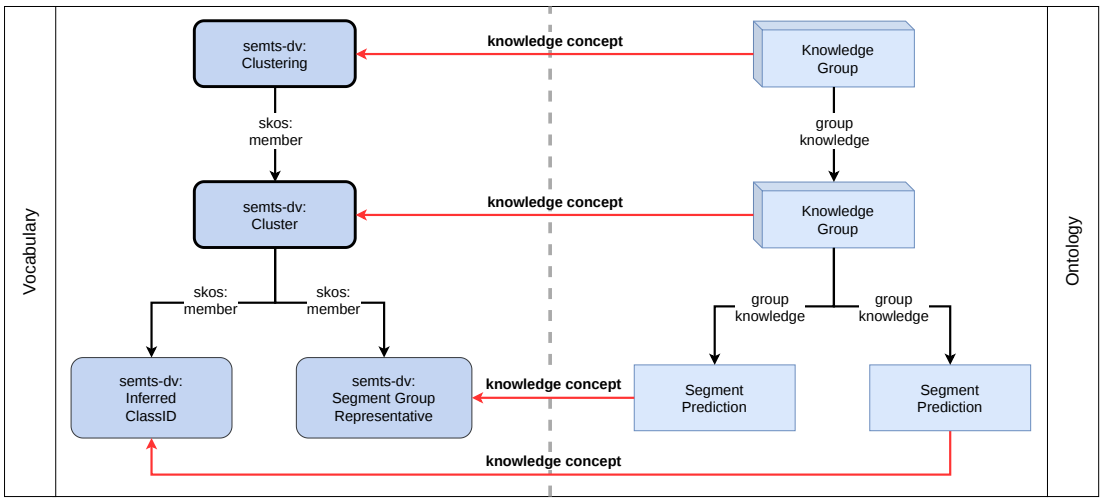


Figure 4.5: Example for the hierarchical mapping of particular knowledge concepts from the data knowledge vocabulary (`semts-dv`) to nested knowledge groups. Concepts extending `skos:Concept` are shown in rounded boxes, while those also categorized as `skos:Collection` are additionally emphasized by bold borders.

of a `InferredClassID`. Or in motif discovery a group representing a `Motif` must hold at least two `Motif Element` instances.

The same pattern holds for Scenario Knowledge. The `Setting` collection defines a group representing a composite operating configuration of multiple `ScenarioParameter` members. Additionally, events and actions like `MaintenanceEvent` and `Alert` can be organized in groups that document a single operational context.

The vocabulary-aligned approach brings several practical advantages: First, knowledge and knowledge groups retain a consistent meaning even if their string-based labeling differs across runs or datasets. Second, relations such as `skos:broader` and `member` enable queries at any abstraction level (e.g., retrieving any knowledge with concept `SegmentRelation` can include more specific knowledge representing `Distance`, `Similarity` or `Correlation`). Third, as demonstrated, nested, complex knowledge outputs can be preserved as coherent groups by a validation of SKOS hierarchies. Finally, because `KnowledgeConcept` spans both `skos:Concept` and `skos:Collection`, the same mechanism accommodates compatible taxonomies without changing how ontology classes

are represented. For instance, beyond the vocabularies presented here for `DataKnowledge` and `ScenarioKnowledge`, additional vocabularies covering `MethodKnowledge`, `KnowledgeQualityMeasure`, and `KnowledgeGenerationMethod` are provided in [DYD24].

4.4.5 Axioms, Constraints, and Reasoning Capabilities

In SemTS, structural consistency and reasoning are handled through a combination of OWL axioms, which operate under the Open World Assumption to infer new knowledge, and integrity constraints, which operate under the Closed World Assumption to validate data completeness. The logical framework of the SemTS ontology is categorized into standard automated reasoning components [Abb12]: **TBox axioms**, **RBox axioms**, and **ABox assertions**, alongside dedicated **integrity constraints**. In this section, we explore how these mechanisms are employed to enable reasoning, knowledge propagation, and data validation.

TBox Axioms

TBox axioms in the SemTS ontology are used to establish class hierarchies and define precise semantic boundaries. For instance, the class `Knowledge` is the most abstract representation of all knowledge, with three main subclasses: `Data Knowledge`, `Scenario Knowledge`, and `Method Knowledge`. SemTS uses `rdfs:subClassOf` axioms to create such hierarchies in order to allow reasoning engines to automatically infer implicit relationships. For example, if a piece of knowledge is classified as `SegmentPrediction`, a reasoner infers that it is also a type of `DataKnowledge`. To maintain semantic clarity and avoid logical contradictions, the ontology further employs disjointness axioms (`owl:disjointWith`). For example, `DataKnowledge`, `ScenarioKnowledge`, and `MethodKnowledge` are defined as disjoint classes, meaning that a single instance cannot belong to more than one of these categories. Furthermore, TBox axioms also include OWL class restrictions that define `TimeSeriesSegment` as a class of entities with exactly one `segmentIndex`.

RBox Axioms

RBox axioms define the characteristics of properties to enable the discovery of implicit relationships. At a basic level, SemTS uses domain and range axioms to infer entity types. For instance, the `hasUnit` property has a domain of `DataDimension` and a range of `qudt:Unit`. If a reasoner encounters a new entity linked via this property, it automatically infers that the subject entity is a `DataDimension` and the target entity is a unit of measurement, enhancing the knowledge graph without requiring explicit typing. The SemTS ontology further defines characteristics for properties such as `isPartOf`. This property is transitive, meaning that if Segment A is part of Segment B, and Segment B is part of Segment C, the reasoner automatically infers that Segment A is also part of Segment C. This is crucial for representing hierarchical relationships, such as identifying that a daily segment is implicitly part of a larger weekly segment.

Property chain axioms are another type of RBox axiom used in the ontology. For example, the `wasDerivedFrom` property can connect a `KnowledgeGenerationOutput` to a `KnowledgeGenerationInput`, which can itself be linked to the raw data of a particular `TimeSeriesSegment`. This allows reasoning engines to trace the provenance of knowledge back to the processes that created it.

Further defined axioms enable segment-based reasoning in SemTS regarding property definitions of Allen's interval algebra [AF94; All83]. The additional specification of interval relations in combination with OWL-Time allows for the inference of temporal offsets and correlations, where property chain axioms derive segment-level relations from index-level OWL-Time relations.

ABox Assertions

In the SemTS ontology, the ABox contains specific data assertions about instances of `TimeSeriesSegment`, `Knowledge`, and other classes. For example, an ABox assertion of the `SegmentRelation` class can represent a computed relationship between two or more time series segments, such as a correlation value or a distance metric. This instance is associated with the `multiSegmentKnowledge` property, linking knowledge that depends on or stems from multiple segments. This is in

contrast to `segmentKnowledge`, which links knowledge relevant to a single time series segment. This semantic distinction is beneficial because it allows the ontology to query and differentiate between knowledge derived from individual segments and knowledge arising from the interaction of multiple segments. Furthermore, instances of `KnowledgeGenerationEntity`, such as a clustering algorithm, can be asserted alongside links to their concrete inputs (`used`) and outputs (`generated`), grounding the provenance reasoning in specific data points.

Data Validation and Integrity Constraints

While OWL axioms are used for inference, ensuring that concrete data is structurally complete requires integrity constraints. To validate data, SemTS utilizes structural constraint schemas (e.g., via SHACL) alongside its OWL axioms.

These constraints determine the required shape and structure of the data graph. Cardinality constraints, for example, enforce that a valid `TimeSeriesSegment` must have exactly one `segmentIndex` and at least one `segmentDimension`. Similarly, the `Knowledge` class is constrained to have exactly one `knowledgeConcept`. Unlike OWL axioms, which assume that missing data might simply exist elsewhere, these integrity constraints actively validate the provided data, ensuring each segment is properly indexed and preventing ambiguous or incomplete knowledge representations. An example of a SHACL shape is provided in Section 4.6, while the complete collection of SHACL shapes can be found in the online repository defined in the subsequent section.

4.5 Technical Specification, Documentation & Maintenance

This section compiles the concrete technical choices underlying SemTS and the accompanying tooling that supports its practical adoption.

Ontology Formalization and Serialization: SemTS and its vocabularies are authored in Protégé and distributed in four serialization formats: RDF/Turtle,

JSON-LD, OWL/XML, and N-Triples.

Ontology Publication: All resources are published in accordance with the Linked Data Principles [Ber06] and documented following WIDOCO conventions [Gar23]. Persistent W3ID-based identifiers anchor the namespace at https://w3id.org/semts, with dedicated paths separating the ontology from its vocabularies and supporting assets:

- Ontology:
  https://w3id.org/semts/ontology/{version}
- Data Knowledge Vocabulary:
  https://w3id.org/semts/vocabulary/data-knowledge
- Scenario Knowledge Vocabulary:
  https://w3id.org/semts/vocabulary/scenario-knowledge
- Code Repository & Further Assets:
  https://w3id.org/semts/code

where the optional `{version}` path selects a specific release. WIDOCO [Gar17] produces both an HTML version for human readers and machine-readable serializations, served via HTTP content negotiation. The ontology is indexed in Linked Open Vocabularies (LOV) [Van+17] and its prefix is registered at *prefix.cc*.

Ontology Maintenance: The source repository resides on GitHub under a CC BY 4.0 license, where version control, issue tracking, and pull requests form the backbone of a community-oriented evolution process. The vocabularies are explicitly conceived as living artifacts that grow with analytical practice: new terms enter through community proposals and are consolidated in versioned releases, each of which triggers a regeneration of the WIDOCO-based documentation.

Population Support: Because the information modeled in SemTS typically originates from analytical program code, practical accessibility beyond manual RDF

authoring is essential. Two complementary tools, which will further be explained in Section 5.2, address different integration scenarios. The *SemTS Population Library* is a Python package built on RDFLib [Kre+25] that exposes programmatic abstractions for constructing SemTS-compliant knowledge graphs. This library targets developers who require fine-grained, explicit control over the entities they instantiate. A second pathway, the *Code2Onto* framework [Gra+25b], automates ontology population from analytical source code and its runtime context by employing Large Language Models within a multi-agent architecture (cf. Section 5.2.2).

4.6 Validation and Evaluation

We evaluated SemTS using two complementary strategies: automated structural and quality assessment using dedicated validation tools, and a data-driven evaluation through competency questions executed against populated knowledge graphs.

Structure and Quality Assessment. Modeling pitfalls were detected using OOPS! [PGS14], which flags issues at various severity levels ranging from minor suggestions to critical structural flaws. All issues classified as critical or important were resolved, and the scanner results are linked from the published online documentation. Conformance with FAIR principles was verified through FOOPS! [GCP21], with identified issues subsequently addressed. The current FAIRness score is reproducible by submitting the ontology IRI to the publicly available FOOPS! endpoint.

Data-driven Ontology Validation The central evaluation instrument consisted of competency questions posed to knowledge graphs that had been populated with data from several industry and research projects (some concrete applications are discussed in Section 5.1). To provide full reproducibility on open data, we additionally constructed knowledge graphs by applying a selection of established analysis methods from sktime [Lön+19] to samples from the UCR Time Series

Competency Question:
Is it possible to retrieve segments representing particular knowledge?

Example:
Retrieve two segments having a positive linear trend with slope > 1

SPARQL Query

```
PREFIX semts: <https://w3id.org/semts/ontology#>
PREFIX semts-dv: <https://w3id.org/semts/vocabulary/data-knowledge#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?segment ?valueString
WHERE {
    ?segment semts:segmentKnowledge ?knowledge .
    ?knowledge semts:knowledgeConcept semts-dv:LinearSlope .
    ?knowledge semts:hasValue ?value .
    ?value semts:valueString ?valueString .
    FILTER(xsd:float(?valueString) > 1)
} LIMIT 2
```

Formatted Response

```
Segment: TimeSeriesSegment_026607a659, Slope: 1.304
Segment: TimeSeriesSegment_63126db1dc, Slope: 4.606
```

Visualization

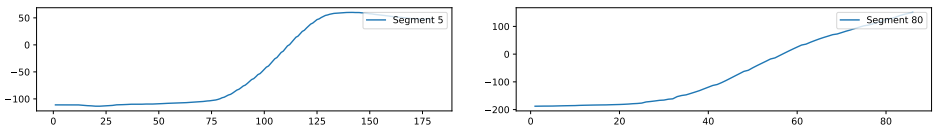


Table 4.2: Evaluation of SemTS through competency questions. Example taken from [Gra+26].

Archive [DKK+18]: specifically, the *Trace* dataset (simulated instrumentation faults in a plant setting) and the *GestureMidAirD1* dataset (one spatial dimension of hand trajectory recordings).

A catalogue of more than 20 competency questions (Table 4.3) was developed based on the requirements from Section 4.3.1. The questions are organized along the three functional areas of the ontology, namely data management, knowledge

classification, and knowledge generation, and cover tasks such as retrieving segment metadata, filtering by knowledge concept, and reconstructing provenance chains.

Execution against both the project-derived and the openly available datasets produced correct and complete results in the majority of cases, as reported in [Gra+26]. The remaining discrepancies that emerged led us to further iterations of discussions and subsequent ontology refinements. Table 4.2 illustrates one such question with its SPARQL implementation and output, while Table 4.3 lists questions from the catalogue. An executable script is available in the SemTS repository.

Table 4.3: Competency Questions and Examples

Competency Question	**Example**
Data Management	
Can the hierarchical structure of time series segments be retrieved?	Retrieve parent-child relationships between segments.
Can all dimensions of time series segments be retrieved?	Retrieve dimensions of segments with associated properties.
Is it possible to identify segment metadata?	Get title, dataset column index and time index of 5 segments.
Is it possible to retrieve details on a knowledge value associated with a segment?	Get the datatype of an arbitrary knowledge value.
Is it possible to identify all segments within a particular time window?	Retrieve all segments between ”2024-01-01T00:00:00” and ”2024-01-09T07:59:59”.
Knowledge Classification	
Can all knowledge categorized as ”Data Knowledge” be retrieved?	Retrieve all knowledge instances directly derived from data.
Can all knowledge categorized under ”Method Knowledge” be retrieved?	Retrieve knowledge about optimal method presets.
Can all knowledge categorized under ”Scenario Knowledge” be retrieved?	Retrieve all domain-specific knowledge instances.
Can quality measures for specific knowledge entities be retrieved?	Retrieve the quality measure of knowledge instances.
Can reasoning-based knowledge generation processes be identified?	Retrieve all reasoning processes used to validate knowledge.

Competency Question	**Example**
Can knowledge quality measures and their associated concepts be retrieved?	Retrieve all evaluation metrics for knowledge quality.
Is it possible to retrieve any segment representing particular data knowledge?	Retrieve two segments having a positive linear trend with slope greater than 1.
Is it possible to identify all knowledge concepts occurring in a certain time frame of a time series?	Retrieve all knowledge concepts associated with segments overlapping with time ”2024-01-02T12:00:00”.
Is it possible to filter the data based on scenario knowledge?	Retrieve all segments which are annotated by a label.
Knowledge Generation	
Can all generated knowledge and their provenance be retrieved?	Retrieve knowledge entities and their generating entities.
Can knowledge describing specifics of an algorithm be retrieved?	Retrieve knowledge defining algorithms.
Can all embedded values in the knowledge graph be retrieved?	Retrieve embedded values such as mean or variance.
Can referenced values and their description be retrieved?	Retrieve referenced values stored externally and their descriptions.
Is it possible to retrieve knowledge that was generated by a particular analysis method?	Retrieve all knowledge instances generated by an Autoencoder.
Is it possible to list all segments that contributed to the generation of knowledge?	List all segments which served as input to the generation of knowledge.
Is it possible to retrieve knowledge based on specifics from the generation process?	Retrieve any prior knowledge, which means any knowledge that was not derived from raw data or other knowledge.

Instance Validation

Concerning the definition of axioms (cf. Section 4.4.5), instance validation in SemTS is used for maintaining the semantic integrity and consistency of individuals. By ensuring that instances adhere to the defined ontology constraints, knowledge representations remain valid. In SemTS, instance validation on ABox data is performed using SHACL, to enforce the correctness of structural and syntactic constraints (e.g., cardinalities and datatypes). The corresponding collection of SHACL shapes has been made available in the online repository.

For the SemTS ontology, SHACL validation serves two main purposes. It ensures

that the data and knowledge instances comply with the structural requirements and semantic constraints defined through axioms. Furthermore, it may be used to enforce alignment between the ontology's knowledge groupings and the vocabulary hierarchies, which are structured using SKOS (cf. Section 4.4.4). The validation framework for SemTS is thus organized into two main categories, to capture both the structural and semantic aspects of the ontology:

Structural Validation Structural validation ensures that the properties, classes, and relationships are used correctly and that they conform to the cardinality constraints, data types, and other restrictions defined in the ontology. For example:

- **Node Shapes:** SHACL shapes are defined for each class in SemTS, such as `TimeSeriesSegment`, `Knowledge`, and `DataDimension`. These shapes validate the required properties, their cardinality, and their expected data types or associated classes.
- **Disjointness Constraints:** Certain classes in SemTS are defined to be disjoint. For instance, `DataKnowledge` cannot also be an instance of `MethodKnowledge`, `ScenarioKnowledge`, or `KnowledgeGroup`.

An example of a structural validation rule is the `SegmentPredictionGenerationMethodShape`, which ensures that every `Segment Prediction` was generated by exactly one `Knowledge Generation Method`. The associated SHACL shape in Listing 4.1 illustrates the validation of this constraint.

Semantic Validation Semantic validation ensures that instances of SemTS adhere to the hierarchical and domain-specific semantics defined by the ontology and its SKOS-based vocabularies. This type of optional validation ensures that assigned concepts for entities align with their respective hierarchies and that group-level relationships are semantically meaningful. Examples include:

- **Concept Alignment:** For instance, `DataKnowledge` instances must use a concept that is part of the `semts-dv:DataKnowledge` hierarchy. This ensures that

Listing 4.1: Example of a SHACL Node Shape to ensure that every `Segment Prediction` was generated by exactly one `Knowledge Generation Method`

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix semts: <https://w3id.org/semts/ontology#> .
semts:SegmentPredictionGenerationMethodShape
a sh:NodeShape ;
sh:targetClass semts:SegmentPrediction ;
sh:property [
    sh:path semts:knowledgeGenerationEntity ;
    sh:class semts:KnowledgeGenerationMethod ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:name "Unique Knowledge Generation Method" ;
    sh:description "Every SegmentPrediction must be generated by exactly one
        KnowledgeGenerationMethod" ;
    sh:message "SegmentPrediction must have exactly one
        knowledgeGenerationEntity
    of type KnowledgeGenerationMethod."
] .
```

the assigned concepts are semantically appropriate and consistent with the ontology's domain-specific vocabulary.

- **Group Knowledge Validation:** The SHACL shapes validate the structure and contents of knowledge groups, such as ensuring that a clustering group contains one or more clusters, each with at least one class ID and optionally representatives like shapelets.

4.7 Impact and Ongoing Adoption

Beyond the core ontology design and evaluation presented in the preceding sections, SemTS has been adopted in several applied contexts that demonstrate its broader relevance. In the Next Generation Dataspaces Initiative (NGDI) [Fra25], SemTS is planned to be adopted as a standardized semantic layer for describing and assessing time series knowledge across multiple dataspace applications, contributing to improved interoperability and quality assurance in cross-domain data infrastructures. Additionally, the integration of SemTS with Large Language Model-based data science agents [Gra+25c] using Graph-based Retrieval Augmented Generation [Col+25] has enabled semantically enriched reasoning over

time series insights within agentic frameworks. Other ongoing work investigates the mapping of SemTS knowledge instances to experiment tracking platforms such as MLflow [Zah+18], pursuing bidirectional integration strategies where external tools both consume and produce SemTS-compliant artifacts. Furthermore, the development of application profiles using the AP-first methodology [Fra22] is being explored to enable more precise domain-specific adaptations of SemTS across varying application areas.

Chapter Summary

This chapter presented *SemTS*, the Semantic Time Series Ontology, as a core contribution addressing RQ 2. SemTS organises time series knowledge across three interconnected layers: *Data Management* for segment-based indexing of time series data, *Knowledge Classification* for semantically typing insights into Data Knowledge, Scenario Knowledge, and Method Knowledge, and *Knowledge Generation* for representing how knowledge is produced and propagated with full provenance. Through integration with established ontologies, SemTS achieves interoperability while avoiding redundant modelling. The recursive definition of `KnowledgeGroup` aligned with SKOS hierarchies enables the representation of arbitrarily nested knowledge compositions. Axioms at different levels support reasoning and consistency via semantic and structural validation. The ontology was evaluated through competency questions applied to multiple datasets and validated using OOPS! and FOOPS! scanners, with all artefacts published under persistent W3ID identifiers and a CC BY 4.0 licence.

What follows. Chapter 5 transitions from the ontology as a schema to its practical instantiation as a knowledge graph, addressing how SemTS can be populated, queried, and used to propagate knowledge through concrete analytical workflows.

5 SemTS Knowledge Graph

In Chapter 4, we discussed how SemTS provides a semantic model capable of classifying and representing time series knowledge as well as its associated generation. In this chapter, we examine how such time series knowledge can be utilized to generate one common knowledge graph that serves as a repository for exploratory data analysis, machine learning and other AI-based technologies. This chapter includes approaches to an instantiation of concrete knowledge entities as well as their retrieval, propagation and validation, referencing objectives related to RQ 2.

Chapter Outline

Chapter 5 begins by highlighting real-world scenarios derived from research and industry projects modeled using SemTS in Section 5.1. Following the previous discussion of the ontology, this section aims to demonstrate how specific situations can be modeled in practice. Next, Section 5.2 introduces two distinct methods for populating the SemTS ontology, including an agentic approach for automated ontology population from code. In Section 5.3, we explore both query-based and embedding-based approaches for retrieving these entities from a knowledge graph. Finally, Section 5.4 delves into the use of SemTS for defining knowledge propagation through a formal algebraic framework. Additionally, we examine how scenario-specific information, combined with Data Knowledge, can support contextual reasoning.

5.1 Real-World Application Scenarios

In the following, we present a selection of real-world application scenarios that we used to derive requirements and to evaluate SemTS. Since these scenarios involve sensitive information from partners across research and industry projects, the corresponding scenario examples use artificial values. The primary purpose of selected scenarios is to demonstrate how SemTS can be effectively applied and utilized across various domain contexts and analytical tasks. Scenario descriptions are not intended to provide a comprehensive representation of each case, but to emphasize the functionality and structures of SemTS. Each scenario comprises a general overview of the project, an example scenario illustrating one of its use cases, and an explanation of how this use case is represented in SemTS, focusing on a particular aspect of the ontology. The complete examples are provided in Appendix D, while simplified, derived versions are shown in the corresponding figures below. Equivalent to the architectural diagram of SemTS in Figure 4.2, visualizations of this subsection are colored as follows: Data Management (green), Knowledge Classification (blue) and Knowledge Generation (red).

5.1.1 Computer Vision for Animal Welfare

The Tierwohl-KI project [Bee+24b] was funded by the German Ministry of Food, Rural Affairs and Consumer Protection Baden-Württemberg to improve animal welfare in pig slaughterhouses using artificial intelligence. Previously, video footage was manually reviewed by employees to identify potential issues, but this process was time-consuming and often missed some violations due to the large volume of data. The project developed an AI system capable of automatically detecting situations that posed risks to animal welfare, such as overcrowding near stunning systems, unusual animal behavior, improper tool use, and people present in areas where access is normally not permitted. By flagging relevant video scenes, the system reduced the need for manual reviews and improved detection efficiency.

Example Scenario. This scenario describes an automated animal welfare monitoring that records 10 seconds of multivariate data (video, audio, and paddle velocity) on September 15, 2025. During the recording, machine learning models automatically detect and classify two events: a person appearing in the video feed at 4.5 seconds and a paddle "hit" event with 3.8 m/s velocity at 5.1 seconds.

SemTS Representation. The example of video and paddle velocity monitoring for event detection primarily illustrates how SemTS defines the representation of complex annotations in a multimodal time series context. The YOLO-based object detection method [Bee+24a] identifies classified objects such as the presence of a person (`person-detection-knowledge`) in video frames, while an analysis of the paddle velocity determines intensive paddle hits (`paddle-hit-knowledge`). As the generation method does not only provide the coordinates of the recognized object but also the class assignment of the object, the result is defined as a knowledge group comprising both information. While the class label is comparatively simple information it is represented by an embedded value, whereas complex information such as the predicted bounding boxes are instantiated as referenced values inherently using DCAT pointing to the separately stored data (`dist-yolo-bboxes-person-04480-04520`). The bounding box annotations are stored as YOLO coordinates in a JSON file, where each box is represented by its normalized center coordinates, width, and height relative to the image dimensions. This format ensures compatibility with standard object detection workflows while maintaining a lightweight and interoperable data structure. The example also shows the explicit use of units (e.g. `unit:M-PER-SEC`) and data characteristics to ensure interoperability and increase data understanding. Figure 5.1 illustrates the definition of knowledge groups and the two different value classes.

5.1.2 Forecasting of Grinding Beads Wear

In the scope of an industry project, started in collaboration with the German initiative Mittelstand-Digital Zentrum[1], the goal was to enhance sustainability and process stability in the manufacturing industry through artificial intelligence. The

[1] https://www.mittelstand-digital-wertnetzwerke.de, last update: Dec' 2025

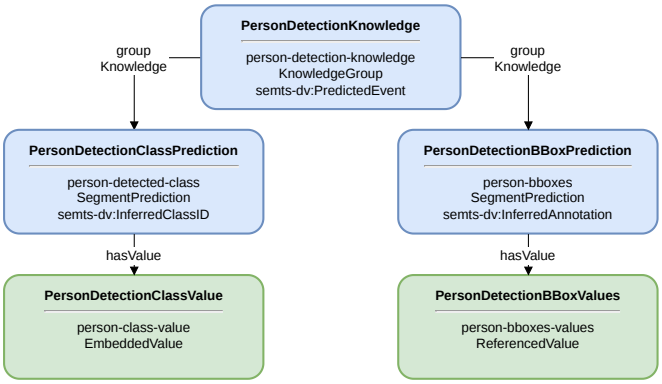


Figure 5.1: Scenario 1: Description of the use of knowledge groups and different value classes in SemTS.

project aimed to address the wear-and-tear of ceramic grinding beads, essential components in the production of various industrial and technical materials. Late replacement of these beads can lead to increased material consumption and incorrect machine behavior. To address this problem, the project developed an AI-driven wear prediction system. By analyzing real-world machine and wear data provided by a long-term customer, the team created a structured data foundation. Iterative analysis enabled the identification of relevant parameters and correlations influencing bead wear. This approach allowed predicting optimal replacement timings, minimizing costs and environmental impact while improving operational efficiency.

Example Scenario. This scenario describes an industrial grinding bead wear monitoring system that tracks and analyzes grinding bead degradation in a mill over January 2023. The system records hourly measurements of wear rates (mm/hour) and operating temperatures, capturing both routine operational data and significant events like a mid-month refill of 50kg new grinding beads, while generating statistical insights, correlation analyses between temperature and wear, LSTM-based forecasts for February and incorporating expert observations about non-linear wear patterns above 60°C.

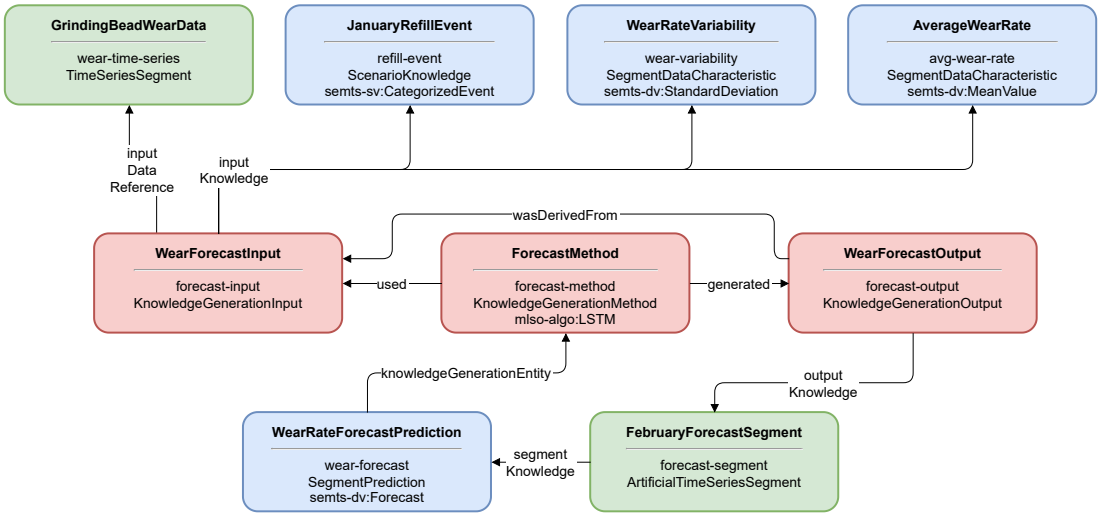


Figure 5.2: Scenario 2: SemTS representation focusing on the synergy of Data Knowledge and Scenario Knowledge. Furthermore it illustrates the use of artificial time series segments, describing generated time series data.

SemTS Representation. The grinding bead wear example, illustrated in Figure 5.2, demonstrates how SemTS effectively supports knowledge propagation. Statistical analysis produces quantitative metrics such as wear-rate and wear-variability, while domain experts provide qualitative insights, including temperature thresholds and operational events like refilling. These discrete events are linked to the raw measurements through segment information, leveraging the hierarchical structure of `isPartOf`. Furthermore, the ontology ensures methodological traceability by explicitly linking analytical outcomes to their corresponding methods including specific inputs and outputs. Knowledge propagation is performed by incorporating the previously generated knowledge `avg-wear-rate`, `wear-variability` and expert-provided insights including `refill-event`. Alongside the raw data this knowledge is used to create a more robust input for the LSTM-based forecast. Additionally, quantitative measures, such as `forecast-accuracy` enable an evaluation of the generated knowledge.

5.1.3 Anomaly Detection for Blow Molding

The knowlEdge project [Alv+21], funded by the European Union, was a collaborative effort of 11 European partners to develop a secure AI system aimed at improving industrial manufacturing processes. Many companies struggle with the analysis of large amounts of data generated by modern production systems, making it challenging to transition from monitoring to predictive capabilities. Despite the potential of AI technologies to enhance sustainability and quality in production, their complexity poses a significant barrier. To address these challenges, knowlEdge focused on creating innovative AI systems and data management infrastructures. The project aimed to lower the entry barriers by developing a secure AI-centric software architecture that supported agile manufacturing scenarios, allowing semi-automated data analysis to identify patterns and anomalies based on domain expertise.

Example Scenario. This scenario, derived from a corresponding publication on anomaly detection for manufacturing [Sch+23], describes a blow molding machine monitoring system that captures multivariate sensor data throughout 2024, including temperature (°C), pressure (bar), vibration (m/s^2), and two binary signals indicating cycle states. Domain experts provide knowledge defining how the combination of these binary signals marks the start and end of machine cycles. A segmentation method applies this cycle definition to derive individual machine cycle segments represented as an array of time tuples. The resulting cycle segments are then used to train an autoencoder, producing a trained model stored as a referenced artifact. Finally, the trained model serves as input to a threshold determination method that computes an optimal reconstruction error threshold (0.85) for subsequent anomaly detection.

SemTS Representation. The blow molding machine example demonstrates how SemTS effectively supports knowledge propagation across multiple analytical stages. A corresponding visualization is provided in Figure 5.3. Domain expert reasoning produces Scenario Knowledge, which formally encodes the binary signal conditions for cycle boundary detection. This expert-provided knowledge,

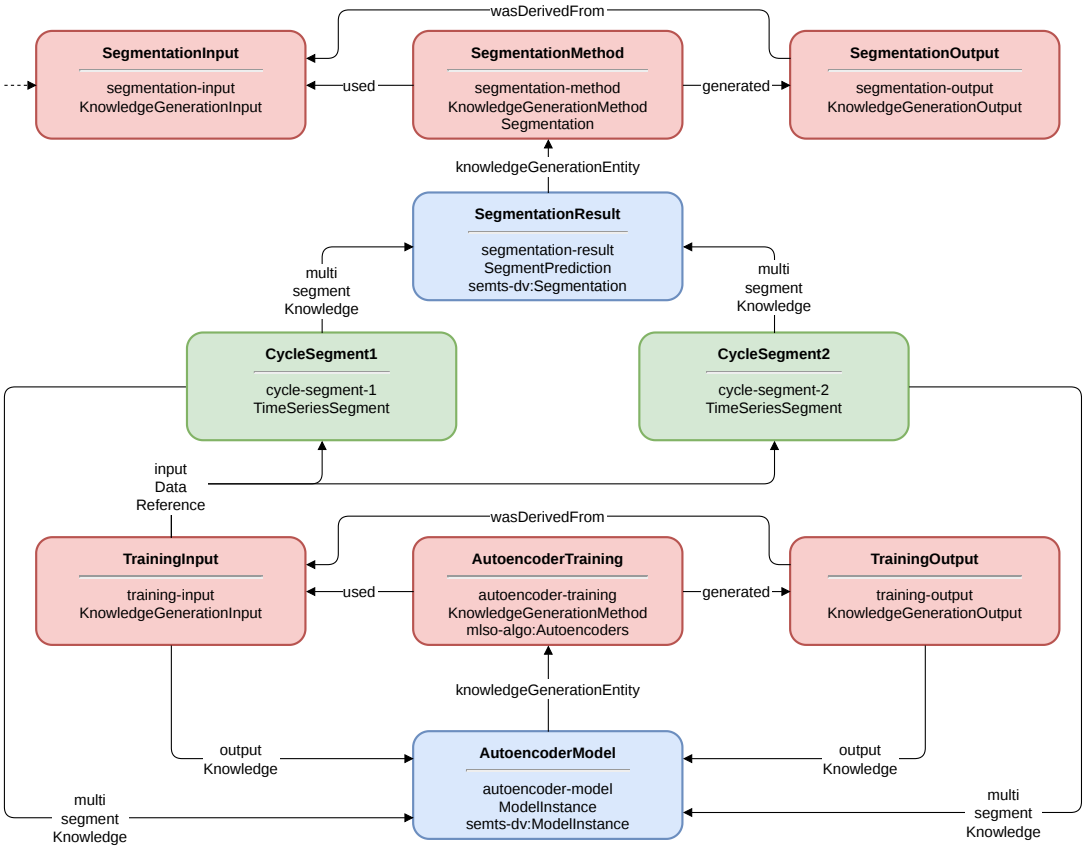


Figure 5.3: Scenario 3: Visualization of knowledge propagation via SemTS in a concrete use case.

together with the raw multivariate sensor data, forms the input for the segmentation method, which outputs `SegmentationResult` containing time tuple arrays. In this example, two of the inferred cycle segments `CycleSegment1` and `CycleSegment2` reference this segmentation knowledge via `multiSegmentKnowledge`, establishing their provenance while maintaining the hierarchical structure through `isPartOf` relationships to the parent time series. Knowledge propagation continues as the segmentation result and cycle segments feed into the transformer autoencoder training, producing `AutoencoderModel` as a `ModelInstance`. The trained model is subsequently propagated as `inputKnowledge` to the threshold determination method, which generates `AnomalyDetectionThreshold` as Scenario Knowledge for future anomaly detection. This chain of knowledge generation entities with explicit inputs, outputs, and `wasDerivedFrom` relationships ensures

complete methodological traceability from expert annotation through segmentation and model training to the final threshold determination.

5.2 Knowledge Graph Generation

So far, we discussed how time series knowledge can be represented by means of SemTS. In concrete applications such as the ones from Section 5.1, an essential step towards the creation of a knowledge graph is ontology population. Informally, ontology population [LNM19] is the process of mapping unstructured information to entities of one or more predefined ontologies and subsequently generating uniquely identifiable instances, also called individuals. To ease this population process, we developed two different solutions for the population of SemTS. The first approach described in Section 5.2.1 is a Python-based library, serving as practical asset for a programmatic generation and validation of SemTS-specific individuals. The second solution presented in Section 5.2.2, named Code2Onto, is a more general and innovative approach and corresponds to a multi-agent framework for automatically conducting ontology population from Python code. Descriptions as well as evaluations of Code2Onto are mainly reused from the associated publication [Gra+25b].

5.2.1 SemTS Ontology Population Library

The SemTS Ontology Population Library (SPL) is a Python-based solution to construct and validate knowledge graphs that follow the SemTS ontology and associated vocabularies. Instead of being a generic ontology population API, it offers a lightweight, specialized abstraction for semantic modeling of SemTS using RDFLib [Kre+25].

Conceptually, the library separates three building blocks. First, graph management: loading, storing, querying, and serializing RDF data. Second, simplified construction patterns: intuitive functions for creating and linking SemTS entities through Python classes instead of using low-level triple manipulations. Third, quality control and inference: optional SHACL validation, inference of tempo-

ral information and configurable RDFS/OWL-RL reasoning. The library thus serves as middleware, mapping analysis code to semantic representations of inferred insights, along with related generation process information and contextual data.

Related Work

Popular ontology editors such as Protégé [Mus15] and NeOn Toolkit [Haa+08] provide graphical interfaces for manually creating ontology individuals. However, these tools prioritize ontology modeling over population, require notable human effort and are not optimized for a programmatic, pipeline-based population of data. At the programmatic level, a variety of general-purpose Semantic Web libraries provide APIs for manipulating RDF graphs and OWL ontologies. Java-based examples include Apache Jena [Apa21] and the OWL API [HB11] for an OWL-centric manipulation. For Python, ontology-oriented programming libraries such as Owlready2 [Lam17] generate and manage OWL entities through Python objects. In the context of SPL, these related solutions influenced the design, but were not adopted as the primary population mechanism. Using Jena as the primary SemTS API involves working with low-level graph operations and requires cross-language integration, which increases complexity for Python-focused workflows commonly used in time series analysis. Alternatively, Owlready2 offers a convenient object-oriented mapping of OWL ontologies in Python and access to comprehensive reasoners such as HermiT or Pellet. Yet Owlready2 is less RDF- and SPARQL-centric, and integrating it with SHACL and external triple stores would introduce an additional abstraction layer on top of RDFLib, rather than simplifying the population task. In the context of SemTS, these technologies are more appropriately used as complementary infrastructure than a replacement for the SPL. Corresponding complements could be the usage of Jena as a scalable backend store and Owlready2 as an additional, comprehensive reasoning interface.

Another category of related work focuses on mapping structured data sources to RDF and ontologies using declarative mappings. R2RML [Wor12] and tools such as D2RQ [BS04] and Ontop [Cal+17] support the definition of mappings from

relational databases or tabular data to RDF graphs, but they typically target schema- and instance-level mappings. In contrast, with SPL we pursue a code-based approach where such constructs are expressed as reusable programmatic patterns, making it easier to embed ontology population into analytic workflows allowing a flexible instantiation of entities where appropriate. Moreover, much information is not yet available but can only be implicitly derived from the code runtime context, instead of structured documents or databases. A solution to this particular problem is discussed in the upcoming Section 5.2.2.

Positioning. Existing general-purpose libraries such as RDFLib and Owlready2 provide the building blocks for RDF manipulation and OWL reasoning, but they leave the burden of encoding domain-specific modeling patterns entirely to the developer. Declarative mapping approaches like R2RML target schema-level transformations from structured sources, yet they cannot capture knowledge that exists only implicitly in the runtime context of analytical code. SPL addresses the gap between these two paradigms by encoding SemTS-specific construction patterns such as provenance-aware knowledge generation chains, SKOS-aligned knowledge groupings, and Allen-style temporal inference as reusable, type-safe Python abstractions. This design decision is motivated by the observation that in time series analytics, much of the information needed for ontology population (e.g., segment boundaries, feature values, model hyperparameters) is available only at runtime and must be captured programmatically rather than declared in advance. SPL thus serves as a controlled, validated middleware between analytical code and the semantic layer, complementing the more flexible but non-deterministic Code2Onto approach presented in the next subsection.

Architecture and Programmatic Abstractions

The SPL reuses concepts of existing approaches, but adds a SemTS-specific layer. It adopts RDFLib as its core RDF representation and SPARQL engine and introduces class-based Python wrappers, mixins, and factory functions for SemTS entities. These abstractions partially encode SemTS modeling patterns (cf. 4.4) and integrate external dependencies like DCAT, PROV and OWL-Time in a preconfigured way, so that users only populate selected classes and properties rather than

manipulating low-level triples from multiple, complete ontologies. SHACL validation is used as the primary mechanism for enforcing structural and datatype constraints and optional RDFS/OWL-RL reasoning supports lightweight inference. Central component of the architecture is the `SemtsSession`. A session wraps an `rdflib.Graph`, a triple store backend, and the configuration objects that define validation and reasoning settings. It binds all relevant namespaces and exposes different methods including SPARQL querying and serialisation. Backends are represented by a simple `TripleStoreBackend` interface with `InMemoryBackend` being the default store to keep all information in process memory, while custom backends can be added to connect to further, external triple stores.

The session behavior is controlled by different configuration classes covered within `SemtsConfig` to define parameters for storage, SHACL validation and reasoning. For instance, `ReasoningConfig` controls whether formal reasoning is applied, which predefined profile should be used and which axioms are to be considered. In this context, a dedicated `Reasoner` class wraps functions of the owlrl [Her14] library and allows to compute a deductive closure using either RDFS or OWL-RL semantics. SHACL validation is available via a `ShaclValidator` wrapper around pySHACL [SC24]. The validator runs shapes loaded from a Turtle file against the graph and raises an error if constraints are violated. Both, validation and reasoning are part of the population pipeline and are performed during the commit phase to persist data in the backend store if enabled.

In addition to schema-based reasoning, the library includes explicit algorithmic support for temporal inference. Although time-related resources are expressed via OWL-Time constructs, the library offers a dedicated procedure for computing Allen-style interval relations between time-indexed segments. A corresponding method scans the graph for segments with temporal indices, maps the OWL-Time representation to comparable numeric or datetime bounds, and applies Allen's algebra to derive relations such as *before*, *meets*, and *overlaps*. The corresponding SemTS properties are then asserted in both directions for each segment pair.

All domain entities are represented by subclasses of `SemtsResource`. This base class encapsulates the session, the resource URI, and the assignment of RDF types. It also provides a minimal set of low-level methods for manipulating prop-

erties, making updates uniform across all resource types. Common modeling patterns are encoded as mixins. For instance, `HasTitleDescriptionMixin` bundles methods for setting titles and descriptions, while `HasUnitDatatypeMixin` covers units and datatypes. Any resource that includes these mixins automatically gains the corresponding methods and consistent triple patterns.

On top of the resource classes, the library provides factory functions for convenience that encode higher-level construction patterns, such as creating time-indexed segments, constructing knowledge items together with their values, or building provenance chains of knowledge generation inputs, outputs, and entities. Another example is the ML-Schema helper for creating hyperparameters and their settings and attaching them to a method implementation.

The Python snippet in Listing 5.1 illustrates a compact example of the library. It shows the creation of a session and a time series segment, the assignment of an embedded mean-value knowledge item and the serialization of the resulting graph. In practice, most of the here explicitly defined information would rather stem from concrete data sources or variables of the underlying program code.

Advantages and Limitations

The SPL offers features to simplify and validate SemTS ontology population. The object-oriented design derived from owlready2 provides a flexible, typed layer over RDFLib to avoid low-level RDF triple management. Configuration-based behavior for backends, validation, and reasoning makes the library adaptable to different scenarios. As external dependencies are already preselected and integrated, the library limits instantiation to the necessary entities. With respect to limitations, in the current version of the SPL, only an in-memory backend is provided, but external triple stores can be implemented by users through extending the provided interface.

The reasoning support is restricted to RDFS and OWL-RL profiles and operates as a global closure over the current graph. More expressive reasoning modules or application-specific inference rules are planned to be implemented as custom extensions. In contrast to recent approaches like LLM-based solutions that generate RDF or ontology content directly from natural language, the SemTS library

Listing 5.1: Example of using the SemTS Ontology Population Library

```
# Create a new session with settings being defined by the YAML configuration
    file
session = SemtsSession.from_yaml("config.yml")

# Time interval and segment
interval = create_time_interval_from_datetimes(session=session,
start_datetime="2025-09-15T08:00:05.100Z",
end_datetime="2025-09-15T08:00:05.150Z")

dim_velocity = create_data_dimension(session=session, label="Velocity", position
    =0, unit=QUDT_UNIT.M_PER_SEC, description="Velocity in meters per second")

segment = create_time_series_segment(session=session, title="Recorded velocity
    segment", temporal_entity=interval, dimensions=[dim_velocity])

# Embedded value and data knowledge
mean_value = create_embedded_value(session=session, value="3.8", unit=QUDT_UNIT.
    M_PER_SEC)

mean_knowledge = create_data_knowledge(session=session, concept=DK.MeanValue,
    value=mean_value, title="Mean of velocity")

# Attach knowledge to the segment
segment.add_segment_knowledge(mean_knowledge)

# Apply reasoning + SHACL validation (if enabled in config) and serialize graph
session.commit()
print(session.graph.serialize(format="turtle"))
```

requires more deliberate, manual engineering. LLMs can quickly generate RDF triples, reducing the amount of handwritten code needed to populate a graph. However, depending on the selected LLM and the degree of subsequent validation, their outputs are usually nondeterministic and inconsistent, whereas the SemTS library emphasizes controlled, type-safe construction and validation. One potential optimization is to combine LLM-based approaches and the SPL to deterministically create validated entities from suggested patterns. In the following subsection we present another custom approach to lay the foundation for this idea, especially with regard to a mapping from code-based information.

5.2.2 Code2Onto: AI-based Ontology Population

The work presented in this subsection is based on and extends our publication [Gra+25b], in which we introduced the Code2Onto framework and its initial evaluation. The following text expands on that contribution by contextualizing

the framework within the broader thesis, providing additional discussion on the validation methodology and elaborating on design decisions. The framework's source code is publicly available.[2]

Motivation and Problem Statement

As discussed in the preceding sections, the SemTS Population Library facilitates instantiation of SemTS through programmatic interfaces but still demands that developers manually specify the concrete mapping from their analysis code to ontological structures. This observation generalizes beyond SemTS: traditional ontology population relies on schema-driven mappings or dedicated ETL pipelines [OOO25], both of which presuppose that the developer possesses detailed ontological expertise and invests considerable effort to maintain custom transformation logic.

This challenge becomes particularly pronounced in data science workflows. A typical analytical script produces intermediate and final results that carry rich semantic information, yet this information lives exclusively within the program's runtime state. Consider a clustering algorithm whose output includes cluster centroids, silhouette scores, and assignment vectors: none of these values appear in the source code as literals, and they vanish once execution completes unless explicitly persisted. The same holds for hyperparameter configurations resolved at runtime, inferred data characteristics, and method provenance. Capturing such knowledge in a machine-readable, reusable form would enable downstream applications ranging from reproducibility tracking to intelligent recommendation of analytical methods.

Prior work on AI-driven ontology population [LB22; Nor+24] has demonstrated how machine learning and natural language processing can reduce the manual burden. These approaches, however, operate predominantly on unstructured text or semi-structured documents and assume that all relevant information is statically accessible. Program code does not fit this assumption because much of the valuable content emerges only during execution. To the best of our knowledge, no existing approach combines source code interpretation with runtime context

[2] https://github.com/semts-ontology/Code2Onto

extraction for the purpose of generating validated ontology instances. Code2Onto addresses this gap by automating the entire path from raw analytical code to formally validated RDF/Turtle representations, requiring only two additional lines of code from the developer. Although we demonstrate the framework using SemTS, its architecture is deliberately ontology-agnostic. All ontology-specific knowledge, including class hierarchies, property definitions, vocabularies, and validation shapes, is provided at runtime through dedicated service endpoints, which means that substituting or extending the target ontology requires no changes to the framework's core logic. This design property is realized through the modular separation of prompt templates, semantic resource retrieval, and validation services, as described in the following architectural discussion.

Related Work

Research on automated ontology population has progressed through several methodological phases. The earliest systems exploited surface-level linguistic patterns in text to discover taxonomic relations. Hearst [Hea92] demonstrated that recurring syntactic constructions such as "X including Y" reliably signal hyponymy, and subsequent systems extended this idea through bootstrapping from seed examples [Bri98; AG00], open information extraction [Etz+11], and linguistically informed extraction [APM15]. A parallel line of work targeted structured and semi-structured sources. Schema matching systems [DR02], virtual RDF mapping languages [BS04; Wor12], and record linkage techniques [Chr12] established declarative ways to project relational data into ontological form. These methods remain in active use but require predefined, manually maintained mappings.

The introduction of distributed word representations and neural sequence models shifted the field toward learned extraction. Distant supervision [Min+09] and multi-instance learning [Sur+12] enabled relation extraction without exhaustive annotation, while convolutional [Xu+15] and recurrent architectures [Lam+16] improved entity and relation recognition across domains by reducing the dependency on hand-crafted features.

Most recently, large language models have opened new possibilities. Their pre-

trained understanding of factual knowledge and linguistic structure allows them to extract triples from text with minimal task-specific training [Kum+20]. Recognizing that unconstrained generation introduces hallucination risks, recent systems pair LLM-based extraction with formal verification. Han et al. [Han+24b] propose iterative verification of generated graph structures, and Mustafa et al. [Mus+24] guide generation through ontological constraints. These contributions address text-to-knowledge pipelines but do not consider the scenario where the source material is executable code whose semantics partially reside in runtime state. Code2Onto builds upon the principle of combining generative flexibility with formal validation, while introducing the novel capability of runtime context integration for program code.

Architecture

The design of Code2Onto, presented in detail in [Gra+25b], follows from two observations. First, runtime objects such as large data frames, fitted model instances, or numerical arrays are frequently non-serializable, memory-intensive, or subject to confidentiality constraints, making it undesirable to transmit them to an external service. Second, ontology validation requires heavyweight dependencies including OWL reasoners and SHACL engines whose versions must remain consistent across invocations to ensure reproducible results. These competing requirements motivate a split architecture with clearly separated responsibilities.

The framework comprises core components whose interactions are depicted in Figure 5.4. At the center stands the *AI Pipeline Orchestrator*, which initializes and coordinates the entire population process. It maintains two prompt templates, a system prompt that defines the agent's structure and available tools, and a user prompt that injects the concrete source code and workflow instructions. The orchestrator builds on LangChain [Cha22] for LLM integration and pipeline management.

Communication between the language model and the available tools is mediated by the *Agent Client*, which presents a uniform interface regardless of where a given tool executes. Internally, the client distinguishes between two execution targets. The *Local Agent Server* runs within the same process as the user's code.

It hosts tools that inspect in-process variables through a configurable resolver mechanism and that write generated Turtle files to the local file system. Because inspection occurs in-process, large or sensitive data never leaves the developer's machine.

The *Remote Agent Server* hosts all ontology-related resources and validation capabilities. It exposes tools for retrieving ontology metadata, class and property definitions, vocabulary hierarchies in SKOS format, and usage examples in Turtle syntax. Crucially, it also provides a multi-stage validation endpoint. This server communicates via the Model Context Protocol (MCP) [Ray25], which standardizes tool discovery and invocation between the client and server processes. Although called "remote" to emphasize the conceptual separation of concerns, the current implementation spawns the server as a local subprocess communicating through standard input/output pipes. MCP's support for networked transports would, however, allow physical distribution across machines without changes to the tool interfaces.

The *LLM* itself acts as both a planner and a reasoner. Given the prompt templates and the roster of available tools, it decides which variables to inspect, how to map discovered information onto ontological structures, and when to invoke validation. Its pretrained knowledge of programming patterns and common analytical concepts supplements the explicit ontology information provided by the server.

Finally, a *Logging* subsystem records all interactions throughout the pipeline, capturing prompt content, tool invocations with their arguments and return values, and validation feedback. This produces a complete audit trail that supports debugging, reproducibility analysis, and post-hoc quality assessment. The logging integrates with LangChain's callback infrastructure through a dedicated handler class.

Tool Design and Population Workflow

The tools exposed through the two agent servers constitute the operational vocabulary available to the language model during planning and execution. On the local side, two capabilities are provided. A runtime context resolver accepts

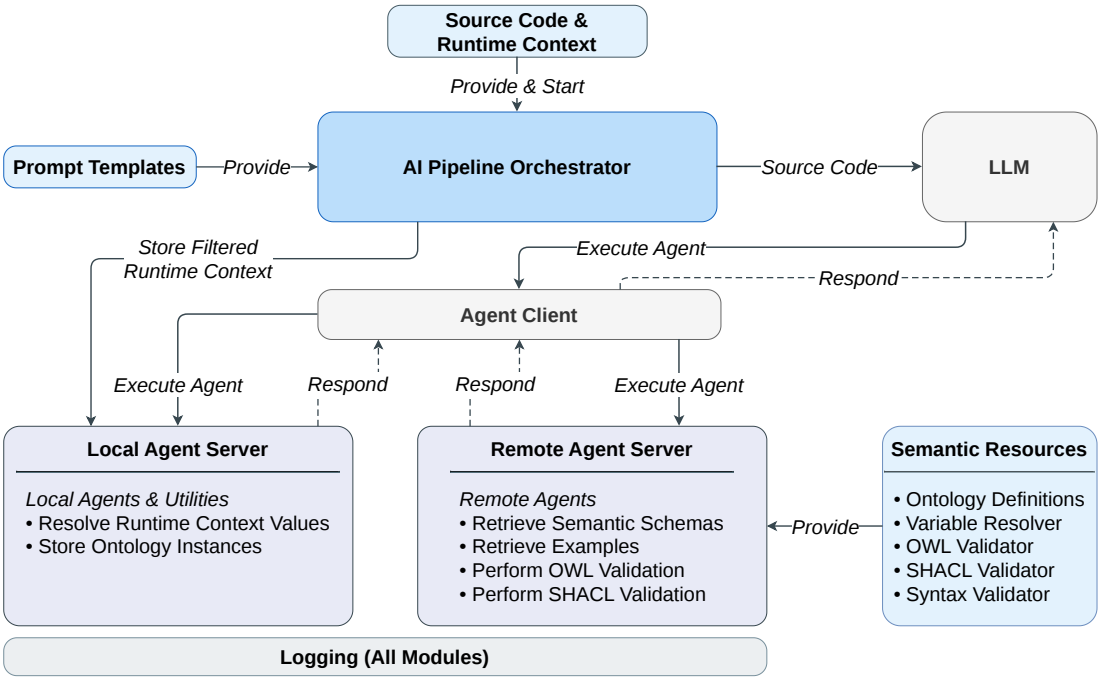


Figure 5.4: Simplified illustration of Code2Onto's architecture and interactions based on [Gra+25b].

a structured specification of variable names and desired attribute paths, navigates the live object graph, and returns a JSON representation containing types, shapes, metadata, and actual values. If a requested variable cannot be located or an attribute path fails, the tool returns a diagnostic message that the LLM can use to reformulate its query. A file creation tool accepts Turtle content, parses it into an RDF graph as an implicit syntax check, and writes it to a specified path, returning confirmation including the file size.

The remote side offers richer functionality organized around four concerns. Ontology retrieval encompasses both a high-level summary, including namespace prefixes, the ontology's abstract, and dependency declarations, and a detailed structural export that lists all classes with their inheritance hierarchies, object properties with domains and ranges, and datatype properties with their constraints. Vocabulary retrieval extracts SKOS concept schemes, preferred labels, definitions, and hierarchical broader/narrower relationships. Example retrieval returns curated Turtle snippets demonstrating correct instantiation patterns for

the target ontology. Validation, discussed in more detail in the next paragraph, performs a graduated sequence of checks and returns structured feedback.

The default workflow proceeds through four phases. Upon invocation, the orchestrator passes the user's source code and the system prompt to the LLM. In the first phase, the LLM fetches ontology metadata and structural information from the remote server to understand the target schema. In the second phase, it analyzes the source code, identifies variables that carry semantically relevant content, and constructs a resolution plan specifying which attributes and sub-attributes to inspect. It then invokes the local resolver, potentially multiple times, to gather the necessary runtime values. In the third phase, the LLM synthesizes the gathered information into Turtle code, mapping variables and their values to appropriate classes, properties, and vocabulary concepts. In the fourth and final phase, the generated Turtle is submitted for validation. If violations are detected, the LLM receives structured feedback and refines its output, repeating validation up to a configurable maximum number of iterations. Once validation succeeds, the local file tool persists the result. A simplified mapping example is shown in Figure 5.5.

Instance Validation

A distinguishing property of Code2Onto compared to purely generative approaches is its integration of formal validation as an iterative feedback mechanism. Validation is executed on the Remote Agent Server to maintain determinism and to centralize the necessary dependencies. It proceeds through three graduated stages.

The first stage is syntactic parsing. The submitted Turtle content is loaded into an RDF graph via RDFLib [Kre+25]. A failure at this point indicates malformed syntax and produces an error message identifying the offending line and expected token, which the LLM uses for immediate correction.

The second stage applies OWL reasoning through Owlready2 [Lam17]. This step detects logical inconsistencies arising from class axioms, property characteristics such as functionality or transitivity, and disjointness declarations. As discussed

```python
from sktime.datasets import load_airline
from sktime.transformations.panel.tsfresh import TSFreshFeatureExtractor

data = load_airline()
extractor = TSFreshFeatureExtractor(
    default_fc_parameters={'mean': None}
)
features = extractor.fit_transform(data)

print(f"Mean value: {features.iloc[0, 0]:.2f}")

# Additional code to execute ontology population
from ontology_population import populate_from_code
populate_from_code(output_path="instances/instances.ttl")
```

Ontology Population via Code2Onto

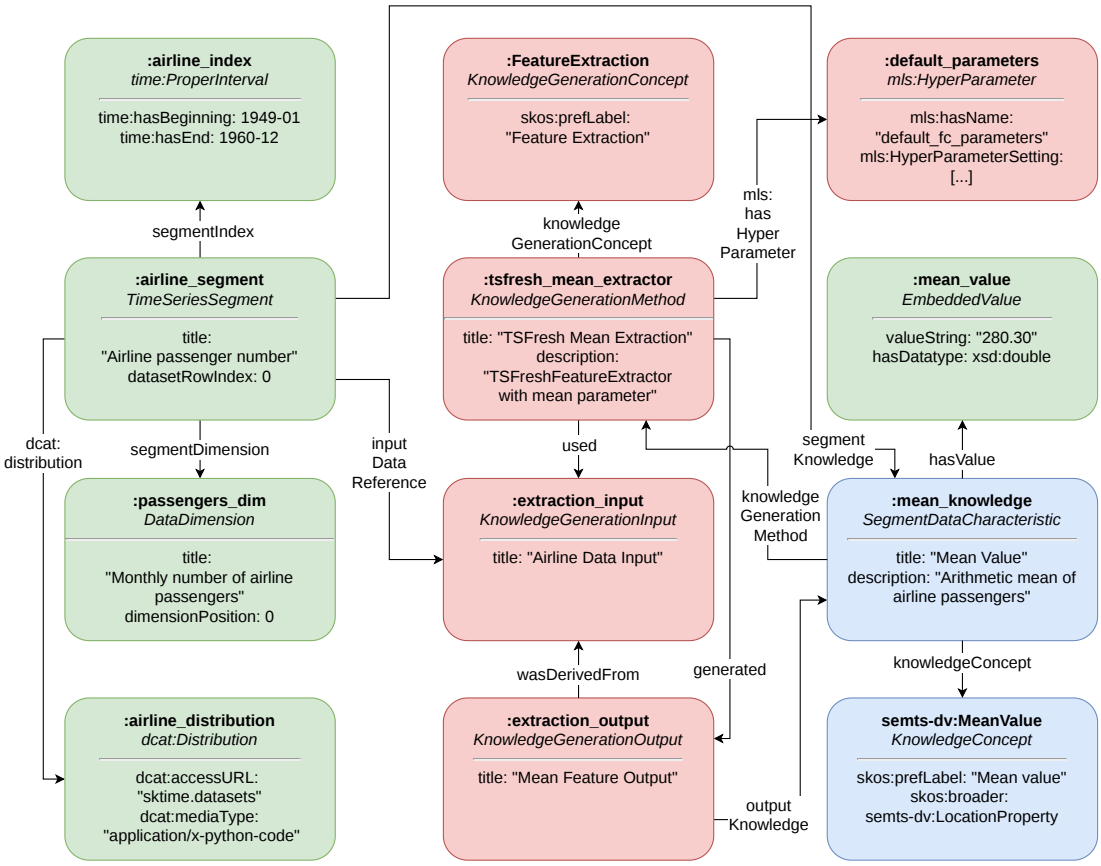


Figure 5.5: Simple code example of a mean computation through the tsfresh library. The code was mapped to SemTS via Code2Onto.

in Section 4.4.5, SemTS defines constraints such as minimum cardinality restrictions on grouping relations and mutual exclusivity among method-related classes. The reasoner identifies violations of these constraints and additionally exposes inferred class memberships that arise from the asserted properties, confirming or contradicting the intended classification of instances.

The third stage performs SHACL validation using pySHACL [SC24]. The shapes encode structural expectations including required properties, permitted node types, datatype constraints, and cardinality bounds. When a shape is violated, the validator returns a structured report identifying the focus node, the property path, the specific constraint that failed, and a human-readable message explaining the expected condition. This structured feedback is critical for enabling the LLM to make targeted corrections rather than regenerating entire graphs.

By combining parsing, reasoning, and shape validation, the framework provides guarantees at three complementary levels: well-formedness, logical consistency, and structural conformance. Executing all checks against the same ontology version with identical inference settings on the remote server eliminates discrepancies that could arise from differing library installations across developer machines.

Evaluation

The evaluation, originally presented in [Gra+25b], assesses Code2Onto along syntactic, structural, and semantic dimensions across ten analytical tasks implemented in Python. The tasks, listed in Table 5.1, span diverse analysis types and libraries to avoid concentrating on a narrow subset of the ontology. Syntactic and structural correctness was verified through the framework's own OWL and SHACL validation logs. Semantic correctness was evaluated in two complementary ways: a manual comparison of generated instance values against ground truth values recorded during code execution, and a set of 30 competency questions formulated as SPARQL ASK queries that probe specific expected content such as the presence of particular hyperparameters or correct data characterizations. An example competency question is shown in Listing 5.2.

Three models from the OpenAI GPT family [Ope24b] were selected to represent distinct capability profiles: `GPT-4.1`, which is specifically tuned for code under-

Task	Analysis Type	Library
1	Forecasting (Theta Forecaster)	sktime [Lön+19]
2	Changepoint Detection (Binary Segm.)	ruptures [TOV20]
3	Anomaly Detection (Z-score)	numpy [Har+20]
4	Out-of-Sample Anomaly Detection	numpy
5	Similarity (DTW Distance)	tslearn [Tav+20]
6	Series Correlation	numpy
7	Series Clustering (K-Means)	tslearn
8	Series kNN-Classification	tslearn
9	Matrix Profile	stumpy [Law19]
10	Mean Imputation	sktime

Table 5.1: Analytical tasks used for evaluation [Gra+25b], covering diverse analysis types and libraries to avoid concentrating on a narrow ontological subset.

standing and generation, `GPT-4o` as a general-purpose model, and `GPT-4o mini` as a resource-efficient alternative. Open-source models were not considered for this initial evaluation, as the goal of the experiments was to assess the effectiveness of the approach itself rather than the robustness and influence of model capacity. Since open-source models generally exhibit lower performance on complex reasoning tasks [Gu+24], their inclusion could obscure whether observed limitations originate from the approach or from the underlying model. The intention was not an exhaustive model comparison but rather an investigation of how different capability profiles affect population quality. All models were configured with a temperature of 0.2 to favor deterministic outputs while preserving enough flexibility for iterative refinement after validation failures.

Metrics. Let T denote the number of tasks and $R_{\max}$ the cap on validation runs. For each task and model pair, the following metrics were computed as defined in [Gra+25b]:

Valid CQ Ratio. The fraction of competency questions answered positively:

$$\text{cq_ratio} = \frac{\text{cq_pass}}{\text{cq_total}}.$$

Valid SHACL Shape Ratio. The fraction of matched SHACL shapes without violations:

Listing 5.2: Example competency question in SPARQL: Does the generated instance correctly capture the window size parameter from task 9?

```
PREFIX semts: <https://w3id.org/semts/ontology#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

ASK {
    ?mth a semts:KnowledgeGenerationMethod .
    ?hp  a semts:Hyperparameter ;
    semts:title "window" ;
    semts:knowledgeGenerationEntity ?mth ;
    semts:hasValue ?val .
    ?val a semts:EmbeddedValue ;
    semts:valueString "6" .
}
```

$$\text{valid_shacl_ratio} = \frac{\text{shacl_valid_shapes}}{\text{shacl_matched_shapes}}.$$

Equal-weight CQ/SHACL Score. A composite metric averaging CQ success and SHACL shape validity.

Inverse Percentage of Validation Runs. A linear, capped measure rewarding fewer validation iterations:

$$\text{inv_percentage_validation_runs} = 1 - \frac{\min(N, R_{\max})}{R_{\max}}.$$

SHACL Improvement Score. A bounded composite that captures relative reduction in violations, the absolute final violation level, and efficiency. Let I denote the initial SHACL violations, F the final violations, and N the number of validation runs. The improvement ratio is defined as:

$$\text{impr} = \begin{cases} 1, & I = 0 \wedge F = 0, \\ 0, & I = 0 \wedge F > 0, \\ \max\{0, (I - F)/I\}, & \text{otherwise.} \end{cases}$$

The overall score combines three weighted components:

$$\text{improvement_score} = \text{clip}_{[0,1]}\big(0.5\,\text{impr} + 0.3\,s(F) + 0.2\,r(N)\big),$$

with auxiliary terms:

$$s(F) = \frac{10}{F + 10}, \qquad r(N) = 1 - \frac{\min(N, R_{\max})}{R_{\max}}.$$

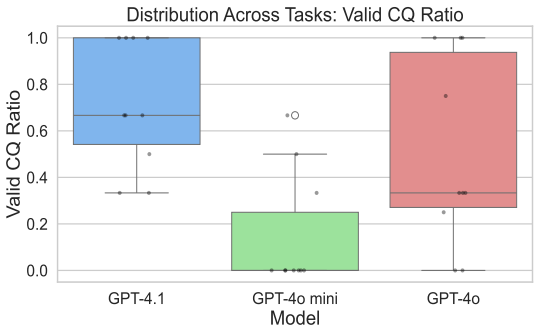
The term $s(F)$ provides a saturating reward for low final violation counts, with the constant 10 setting the penalty scale. The term $r(N)$ penalizes excessive validation iterations relative to the cap $R_{\max}$. The weighting scheme for the improvement score was chosen by design to prioritize actual improvement over absolute quality, and absolute quality over efficiency, yielding a single comparable score per task that reflects progress, final quality, and the effort required. When N is unavailable, the runs term is omitted and the remaining weights are renormalized.

Results. Across all experiments, every model produced syntactically valid Turtle that passed OWL consistency checks, and runtime context values were correctly resolved in all cases. The differences between models emerge in the semantic dimension, as visualized in Figure 5.6.

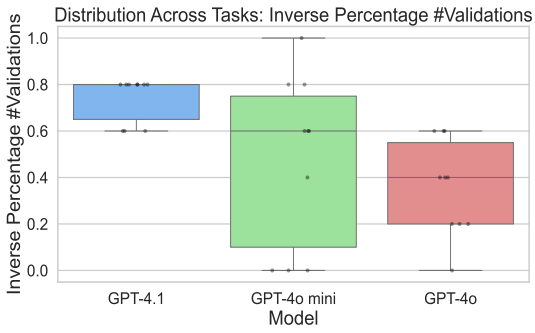
Figure 5.6a presents the distribution of valid competency question ratios. `GPT-4.1` achieved the highest median accuracy, reflecting its superior ability to interpret code semantics and translate them into appropriate ontological structures. `GPT-4o` showed considerable variance across tasks, sometimes matching `GPT-4.1` but often falling short. `GPT-4o mini` consistently underperformed, indicating that lightweight models lack the reasoning depth necessary for this task.

The distribution of validation iterations (Figure 5.6b) reveals a nuanced picture. While `GPT-4o mini` used fewer validation rounds on average, this did not stem from higher initial accuracy but rather from a failure to productively leverage the iterative feedback mechanism. In contrast, `GPT-4.1` also required few iterations, but in this case because its initial outputs already satisfied most constraints.

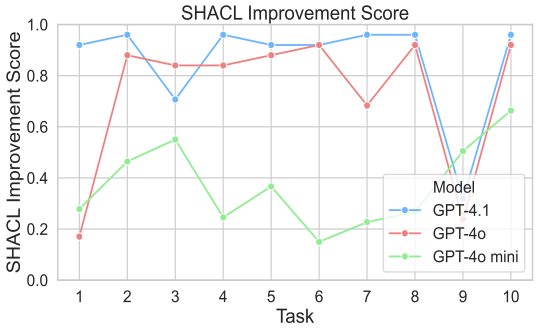
Figure 5.6c tracks improvement across validation iterations using the composite score. `GPT-4o mini` struggled to translate violation feedback into effective corrections. The other two models generally demonstrated strong improvement trajectories, with the exception of tasks 1 and 9, where complex provenance structures



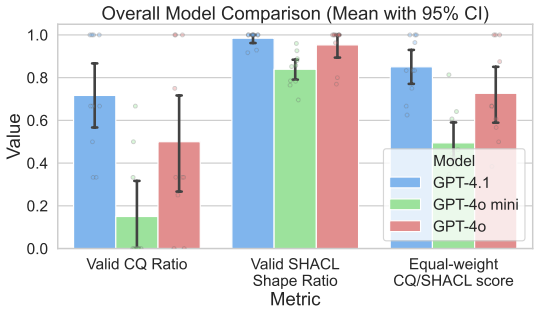
(a) Distribution of CQ accuracy across tasks per model.



(b) Distribution of inverse validation run counts. Higher values indicate fewer required iterations.



(c) SHACL improvement score per task and model, reflecting the effectiveness of iterative refinement.



(d) Aggregated comparison across CQ ratio, SHACL shape validity, and their equal-weight combination.

Figure 5.6: Evaluation results for Code2Onto across three LLMs [Gra+25b].

proved challenging. Notably, no model achieved a perfect score of 1 on any task, confirming that even well-performing models required at least some refinement.

The overall comparison in Figure 5.6d consolidates these findings. All models achieved high SHACL shape validity on average, but the competency question metric exposes substantial differences in how completely the available knowledge was captured.

Analysis of Failure Modes. An inspection of failed competency questions revealed two recurring patterns. The more frequent issue was incomplete knowledge assignment: the LLM identified relevant data but failed to associate it with the

correct ontological segment or knowledge category. Properties specific to knowledge propagation, which encode how inferred knowledge relates to its originating segment, proved particularly difficult for all models.

The second pattern concerns the depth of runtime context exploration. In several cases, the LLM constructed a variable resolution plan that was too shallow, requesting only top-level attributes when the semantically relevant information resided deeper in the object hierarchy. This finding suggests that adopting a more interactive, iterative communication pattern between the LLM and the resolver, analogous to the iterative validation already implemented, could yield substantial improvements. Rather than performing inspection as a single linear pass, the system would benefit from allowing the LLM to refine its resolution requests based on intermediate results.

A third contributing factor is the relative obscurity of the target ontology. Because SemTS is a specialized, recently developed ontology, the language models possess limited prior exposure compared to widely used schemas such as DCAT or Schema.org. Consequently, the models rely heavily on the examples and structural descriptions provided through the remote server. Enriching these resources with more diverse examples and detailed usage guidelines represents a straightforward path to improvement.

The evaluation confirmed that models specifically optimized for code comprehension outperform general-purpose alternatives in this task. This aligns with the expectation that accurate population requires both code interpretation skills and the ability to reason about semantic structures, a combination that code-focused models handle more naturally.

Limitations and Future Directions

Code2Onto's current design entails limitations that suggest directions for future development. The orchestrator operates on a truncated snapshot of the user's code augmented by an inventory of available variables. This design enables processing of unfiltered source code without requiring developers to annotate their programs, but it can lead to incomplete context in scenarios involving complex control flow, conditional variable definitions, or dynamically constructed objects.

Since the population process is guided by an LLM operating at a non-zero temperature, outputs remain inherently probabilistic even when validation constrains the solution space.

The local inspection mechanism traverses attribute paths through straightforward property access, which covers standard Python objects but does not account for properties computed on demand, descriptors with side effects, or lazily evaluated structures. The framework's support for custom code resolvers partially addresses this limitation by allowing developers to register specialized inspection logic for specific libraries or idioms, but this reintroduces a degree of manual effort.

The configurable cap on validation iterations serves as a practical safeguard against unbounded tool invocation loops, but it may terminate the refinement process prematurely for complex graphs that require multiple repair cycles. A more adaptive termination criterion, for instance based on the rate of violation reduction, could balance safety against completeness.

Looking ahead, we identified several enhancements that could strengthen the framework. The runtime resolution workflow could be restructured to support incremental, multi-turn exploration of the variable space, allowing the LLM to investigate deeper into object hierarchies based on what it discovers in initial passes. Operationally, the orchestrator could incorporate richer fallback strategies such as partial validation when the remote server is degraded, or human-in-the-loop confirmation for low-confidence mappings. Furthermore, while the current evaluation relies exclusively on OpenAI models, further comprehensive evaluations are planned that also include open-source models, assessing generalizability across different LLM families. An associated line of ongoing research explores the use of Code2Onto for incrementally constructing knowledge repositories that serve as memory for Data Science Agents [Gra+25c; Col+25], extending the framework's utility toward continuously evolving knowledge bases (cf. Section 6.3).

5.3 Knowledge Retrieval

In this section, we concentrate on enabling access to structured information. To support semantic retrieval from the SemTS knowledge graph, we first outline

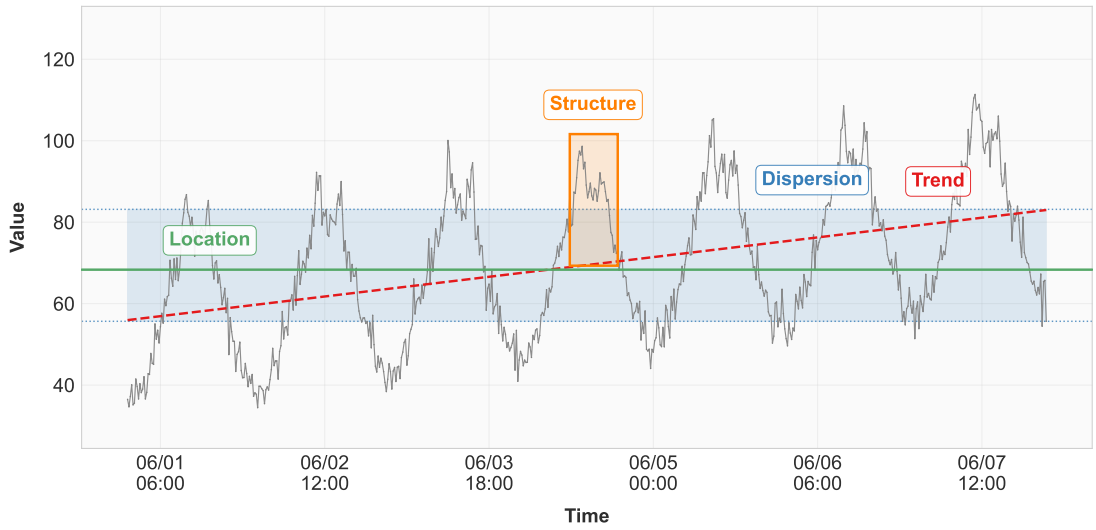


Figure 5.7: Examples of categories of time series characteristics aligned with the Data Knowledge vocabulary. Such categories can, for instance, be used to semantically group individual characteristics for feature engineering, enabling the construction of feature vectors with a consistent, predefined meaning.

in Section 5.3.1 how to leverage the ontology structure and related vocabularies to execute SPARQL queries that exploit the classification of knowledge and associated information. This is particularly effective when questions and tasks are precisely defined. As a more approximate, similarity-based approach, in Section 5.3.2 we also evaluate different embeddings for SemTS to determine their suitability and how ontology information can be retrieved efficiently.

5.3.1 Semantic Indexing

In a populated SemTS graph, each time series segment serves as a semantic pointer into concrete time series data stored in files or time series databases. On the one hand it specifies which segments of a dataset it refers to, on the other hand it manages any knowledge that has been derived from or defined for the corresponding segments, including features, predictions and scenario annotations. The graph thus functions as a semantic index structure that sits between raw data and knowledge that can be navigated in both directions. Starting from high-level

Listing 5.3: SemTS-based Retrieval: Feature Engineering

```
PREFIX semts:     <https://w3id.org/semts/ontology#>
PREFIX semts-dv: <https://w3id.org/semts/vocabulary/data-knowledge#>
PREFIX skos:      <http://www.w3.org/2004/02/skos/core#>

SELECT ?segment ?featureConcept ?featureLabel ?value
WHERE {
    ?segment a semts:TimeSeriesSegment ;
    semts:segmentKnowledge ?k .

    ?k a semts:SegmentDataCharacteristic ;
    semts:knowledgeConcept ?featureConcept ;
    semts:hasValue/semts:valueString ?value .

    # Restrict to trend- and dispersion-type features
    ?featureConcept (skos:broader)* ?group ;
    skos:prefLabel ?featureLabel .
    VALUES ?group { semts-dv:TrendProperty semts-dv:DispersionProperty }
}
ORDER BY ?segment ?featureLabel
```

concepts like particular features types one can locate the relevant segments in the graph and then follow their dataset references back to the underlying values. Conversely, starting from a concrete dataset, one can systematically access all knowledge currently indexed for its segments. While not presenting a contribution on its own, in the following we will explain in which ways SemTS graphs can be queried to answer specific questions or retrieve information for concrete tasks.

Feature Engineering. For feature engineering, the SemTS graph serves as a catalogue of all features computed for each segment, across tools and pipelines. Because segment-level data characteristics are classified using the Data Knowledge vocabulary (see Figure 5.7), feature selection can be expressed in terms of feature families. An analyst can, for example, decide to extract a representation that focuses on trend-specific characteristics, turning the result into semantically grouped feature vectors per segment. This decouples the feature engineering logic from individual extraction implementations, where new, extracted features become part of the graph and thus the shared feature space. As an example, the query in Listing 5.3 retrieves all trend- and dispersion-related features and their values for every segment.

Listing 5.4: SemTS-based Retrieval: Informed Analytics

```
PREFIX semts:    <https://w3id.org/semts/ontology#>
PREFIX semts-dv: <https://w3id.org/semts/vocabulary/data-knowledge#>
PREFIX semts-sv: <https://w3id.org/semts/vocabulary/scenario-knowledge#>
PREFIX skos:     <http://www.w3.org/2004/02/skos/core#>

SELECT DISTINCT ?anomalySeg ?maintenanceSeg
WHERE {
    # Data-driven anomaly segment
    ?anomalySeg a semts:TimeSeriesSegment ;
    semts:segmentKnowledge ?kAnom .
    ?kAnom a semts:SegmentPrediction ;
    semts:knowledgeConcept ?anomConcept .
    ?anomConcept (skos:broader)* semts-dv:Anomaly .

    # Scenario-driven maintenance segment
    ?maintenanceSeg a semts:TimeSeriesSegment ;
    semts:segmentKnowledge ?kMaint .
    ?kMaint a semts:ScenarioKnowledge ;
    semts:knowledgeConcept ?maintConcept .
    ?maintConcept (skos:broader)* semts-sv:MaintenanceEvent .

    # Temporal relation between the two segments
    ?anomalySeg semts:overlaps ?maintenanceSeg .
}
```

Informed Analytics. Informed analytics combines data-derived knowledge with context information so that analyses become aware of scenario knowledge including events, operating modes, or business conditions. Since both kinds of knowledge are attached to the same segments and temporal relations between segments are available directly in the graph, questions such as *"where do anomalies coincide with maintenance?"*, *"which changepoints follow setpoint changes?"* or *"which regimes occur mainly during a particular shift?"* can be expressed without manually aligning timestamps. The semantic index can be queried by asking for segments that satisfy certain behavioral conditions (e.g. anomaly, changepoint, regime) and certain scenario conditions (e.g. maintenance event, lifecycle event, calendar entry) given a temporal relation. The query in Listing 5.4 demonstrates this by retrieving all anomaly segments that overlap maintenance-related segments, which can serve as a starting point for linking anomalies to interventions or for defining scenario-aware training sets.

Experimental Tracking and Methodical Provenance. For experimental tracking and methodical provenance, the SemTS graph connects segments and their

Listing 5.5: SemTS-based Retrieval: Experimental Tracking

```
PREFIX semts: <https://w3id.org/semts/ontology#>
PREFIX mls:   <http://www.w3.org/ns/mls#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?segment ?method
WHERE {
    # Knowledge generation method / implementation
    ?method a semts:KnowledgeGenerationMethod ;
    semts:used ?input ;
    mls:hasHyperParameter ?hp ;
    mls:hasHyperParameterSetting ?setting .

    # Input references a time series segment
    ?input a semts:KnowledgeGenerationInput ;
    semts:inputDataReference ?segment .

    # Hyperparameter identified by name
    ?hp a mls:HyperParameter ;
    mls:hasName "eps" .

    # Concrete hyperparameter setting with value
    ?setting a mls:HyperParameterSetting ;
    mls:specifiedBy ?hp ;
    mls:hasValue ?v .
    FILTER (?v = "2.5"^^xsd:decimal)
}
ORDER BY ?segment ?method
```

attached knowledge to the analysis runs and models that processed them, including algorithm choices, hyperparameter settings and evaluation measures. Each run is represented as a knowledge-generating method that takes segments as inputs and outputs new knowledge such as models, forecasts, labels or additional features. By also following ML-Schema's pattern for implementations and hyperparameter settings, experiments can be queried in terms of both their data and their configuration. This enables questions such as "which segments were used in experiments with a particular hyperparameter setting?", "which models share the same hyperparameters but were trained on different datasets?" or "what hyperparameter ranges tend to work well on segments with specific characteristics?". The example in Listing 5.5 shows how to find all segments that were used in knowledge generation methods whose implementation had a hyperparameter named `"eps"` with value `2.5`, combining SemTS' notion of input segments with ML-Schema's hyperparameter structures.

By querying this provenance-aware index, analysts can reconstruct which data

slices and contextual annotations each model has seen, identify which configurations were actually evaluated on which types of segments, and derive method recommendations grounded in prior experiments, all while maintaining a clear link back to the concrete time series stored in external systems.

The three tasks are only intended as illustrative examples of possible uses. Similar combinations of segment-level knowledge, temporal relations and method provenance can, for instance, be extended to assembling scenario-aware training sets, where segments are selected according to both behavioural and contextual criteria. Likewise, when using knowledge concepts across datasets, the index can support transferring and comparing annotations between them, and the provenance links that describe model configurations can be reused to trace where particular settings have been applied. In a more operational direction, analogous queries can be formulated to inspect the coverage of annotations over time, to compare between data-derived and scenario-provided labels, or to detect specifics in analysis runs.

5.3.2 Embedding-based Retrieval

Many practical retrieval tasks on SemTS graphs rely on similarity and approximation rather than exact graph pattern matches. To address such use cases, we investigated whether embeddings could be effectively learned from entities and relationships encoded in the SemTS ontology. We compared four embedding families, namely relation-specific KGE, random-walk structural embeddings, text-based encoders, and score-level hybrids, across 13 retrieval tasks that probed distinct ontology properties at varying levels of structural complexity.

Related Work

Embedding-based retrieval [Cao+24] on knowledge graphs has been explored extensively in contexts such as link prediction [Ros+21], entity recommendation [Hui+22], and semantic search [Jai+21]. More recently, Graph RAG approaches [Pen+24] have adopted embeddings to retrieve relevant subgraphs as context for language model generation. As noted in surveys [Wan+23; GF18;

CZC18], different KGE models differ in how they score triples (see Definition 7) and in the relational patterns they can express. In general, these models are categorized into geometric and algebraic paradigms. Early translational approaches like TransE [Bor+13] map relations as simple vector offsets to capture linear hierarchies or chains. Conversely, semantic matching models such as DistMult [Yan+15] rely on tensor factorization, making them highly effective for symmetric patterns but poorly suited for directed structures. To handle more intricate relational properties, newer frameworks shift representational geometries: RotatE [Sun+19] models relations as complex-space rotations to capture inversion and composition, while region-based models like BoxE [Abb+20] use axis-aligned hyperrectangles to establish spatial boundaries. This allows BoxE to express complex topological constraints such as containment and overlap without relation-specific scoring functions. Beyond purely structural embeddings, text-enhanced representations [Mor+23; Li+23a] exploit entity labels and descriptions to enrich embeddings with semantic content. Sentence-BERT [RG19] maps texts into dense vectors using Transformer encoders trained for semantic similarity, providing a general-purpose text embedding mechanism. Work on graph-to-text generation [Kon+19] has shown that encoding graph structure directly into attention via Graph Transformers yields substantial improvements over simple verbalization. Contemporary frameworks such as GFM-RAG [Luo+25] reason over graph structures during retrieval-augmented generation, capturing complex interactions between queries and knowledge without domain-specific fine-tuning.

Experimental Design

For the evaluation we generated $M = 3$ independent SemTS instance graphs and performed $N = 2$ training runs per graph with distinct random seeds, yielding 6 evaluation points per task and embedding combination.

Graph Generation. For controlled experiments, each instance graph was produced by a generator that created varied structures while maintaining consistent ontological patterns across all graphs. An LLM generated domain-specific definitions covering algorithms, data dimensions, quality measures, generation con-

cepts, and propagation patterns. These definitions were validated against SemTS vocabularies before being instantiated into a knowledge graph. Structural parameters varied systematically using deterministic functions seeded by graph index, with the number of data dimensions around a base of 14, parent segments around a base of 32, child segments per parent between 25 and 45, and propagation chain depth from 4 to 9 steps, with up to 100 propagation chains per graph. Numeric encodings discretized values into magnitude bins and value quartiles as dedicated entities (`numenc:MagnitudeBin`, `numenc:ValueQuartile`) with 10 bins per concept, which let embedding models learn value-based similarity without having to process numeric literals directly.

Graph Materialization. Following KGE research [Nic+16; Wan+23], we materialized frequently queried multi-hop relationships as explicit triples in an enrichment namespace (`enrich:`). These materialized relations included shared MLS runs, shared concepts, shared knowledge groups, provenance links, causal links, and shared multi-segment relations, capped at a bounded number of shortcuts per group to avoid degenerate fully connected cliques. Each encoded a pattern that would otherwise require traversing multiple hops in the base ontology. This materialization provided explicit training signal for relation-specific models and defined the relation vocabulary used for retrieval scoring.

Segment titles were kept generic (e.g., "S0012-0034") so that text embeddings could not reconstruct structural relationships from lexical hints alone. Statistical features including mean, standard deviation, skewness, and kurtosis were generated per segment, with mean values offset by parent index to create meaningful between-group variation.

To isolate the contribution of materialization, we evaluated three graph variants. The *full* variant contained all enrichment shortcuts and was the default condition. The *no-shortcuts* (stripped) variant removed every `enrich:` relation, so any pattern had to be recovered from the base ontology alone. The *raw* variant was a targeted ablation used for the controlled Deep Provenance experiment (Task 9), removing only `enrich:sharedRun`. When this relation was absent from training, it was absent from the model's vocabulary entirely, which produced random

rankings for relation-specific models.

Embedding Methods. We computed a panel of embedding variants per graph, spanning four complementary families. The *KGE* family comprised four relation-specific models evaluated on their geometric properties:

TransE [Bor+13] served as our structural baseline, projecting relations as rigid vector shifts ($\mathbf{h} + \mathbf{r} \approx \mathbf{t}$). This assumption performs well in steep, tree-like hierarchies where levels map cleanly to constant offsets.

DistMult [Yan+15] scores triples via element-wise multiplication ($\langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle$). Because head and tail entities contribute identically, the scoring is inherently symmetric. Thus it functioned largely as a negative control on the mostly asymmetric SemTS graph.

RotatE [Sun+19] treats relations as complex-space rotations ($\mathbf{h} \circ \mathbf{r} \approx \mathbf{t}$), allowing the model to naturally infer non-symmetric, inverted, and compositional pathways.

BoxE [Abb+20] discards point-to-point mapping in favor of geometric region learning, scoring candidate entities based on whether their vectors fall inside axis-aligned hyperrectangles. This makes it structurally optimized for intersecting or conjunctive constraints.

All KGE models were trained with PyKEEN [Ali+21] using dimension 256, up to 200 epochs with early stopping (patience 2, evaluation frequency 20, minimum delta 0.005), 64 negative samples per positive, and the Adam optimizer with learning rate 5×10^{-4} on the full triple set including enrichment relations. For retrieval, given a query entity h and a task-specific relation r_{task}, scores were computed as $f(h, r_{\text{task}}, t)$ for all candidates t, where f denotes the model-specific scoring function.

The *structural* family added Node2Vec [GL16], a single relation-agnostic embedding learned from biased random walks (dimension 128, walk length 40, 10 walks per node, return and in-out parameters $p = q = 1$, window 5). Unlike the KGE models, Node2Vec produces one vector per node that was scored by cosine similarity rather than through a relation-specific function, which gives a structural representation that does not depend on any individual materialized relation.

The *text* family comprised three task-agnostic encoders. Text encoded only the `dct:title` and `semts:title` labels using Sentence-BERT (`all-MiniLM-L6-v2`) and served as a lower-bound control. Text+ serialized graph neighborhoods into natural language via deterministic templates covering concept, algorithm, dimension, quality, context, provenance depth, generation method, and temporal extent, encoded with `all-MiniLM-L6-v2`. Text+_mpnet encoded the identical serialization with the higher-capacity `all-mpnet-base-v2` encoder, which isolated the effect of encoder capacity. Flat encoded the same fields in a key-value format using semicolon separation, and a Flat_shuffled control randomly permuted the field order to verify that retrieval reflected content rather than ordering artifacts. All text methods used cosine similarity between L2-normalized embeddings.

The *hybrid* family fused each KGE model with Text+ at score level. For a query, the KGE relation scores and the Text+ cosine similarities were each min-max normalized over the finite-valued candidates, excluding the query itself, and combined as $\alpha \cdot s_{\text{KGE}} + (1-\alpha) \cdot s_{\text{text}}$ with $\alpha = 0.5$. This yielded the variants TransE+Text+, DistMult+Text+, RotatE+Text+, and BoxE+Text+, which tested whether equal-weight fusion combined the strengths of both signal types.

A key distinction across families is that KGE scoring is *relation-specific*, computing $f(h, r_{\text{task}}, t)$ for a particular relation, whereas Node2Vec and the text methods use *task-agnostic* similarity independent of any relation type. The text methods are largely independent of the materialization infrastructure.

Metrics. We evaluated our framework using two experimental paradigms, semantic segment retrieval and global link prediction.

For the *semantic retrieval tasks*, 40 to 60 query segments were sampled deterministically, and their relevant targets were defined by ground-truth relationships. The primary evaluation metric is Adjusted Precision@k, formulated as:

$$\text{Adjusted Precision@}k = \frac{\text{Precision@}k - \text{baseline}}{1 - \text{baseline}}$$

where baseline $= |\mathcal{R}|/(|\mathcal{C}| - 1)$ represents the expected precision under random ranking, given the set of true ground-truth relevant segments $\mathcal{R}$ and the complete

set of candidate segments $\mathcal{C}$. Precision@k is defined as the fraction of the top-k retrieved results that are relevant. For the retrieval tasks, we additionally reported Recall@k, defined as the fraction of relevant items appearing in the top-k results, and Mean Reciprocal Rank (MRR), the average reciprocal of the rank at which the first relevant result appears, focusing on $k \in \{5, 10, 20\}$.

For the *global link prediction evaluation*, we measured performance using Mean Reciprocal Rank (MRR) and Hits@k:

$$\text{Hits@}k = \frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{(h,r,t)\in\mathcal{T}_{\text{test}}} \mathbf{1}\big[\text{rank}(t) \leq k\big]$$

where $\mathcal{T}_{\text{test}}$ is the test triple split. The term rank(t) denotes the filtered rank of the true tail entity among all candidates when sorted by the scoring function $f(h, r_{\text{task}}, t)$, excluding all other known true entities in the graph during ranking. We reported Hits@k for $k \in \{1, 3, 10\}$.

Finally, to compare structured and flat representations directly, we computed the structure advantage over the text-only baseline (Flat) as the absolute difference:

$$\Delta = s_{\text{struct}} - s_{\text{flat}}$$

where s_{struct} is the Adjusted Precision@10 of the best performing structured method (KGE, Node2Vec, or Text+) and s_{flat} is the one for the Flat baseline. Throughout our analysis, Adjusted Precision@10 served as the primary benchmark metric for comparison.

Retrieval Tasks

We defined 13 tasks organized into five categories (apart from the baseline) that tested progressively complex structural properties, as summarized in Table 5.2. Each task defined relevance based on ontological relationships of varying complexity, and relation-specific models accessed these relationships through the corresponding materialized enrichment relations.

The *value-based tasks* tested whether embeddings could identify numeric similarity across dimensions. Numeric Range required finding segments with the

Table 5.2: Overview of retrieval tasks organized by structural category. Each task tests whether embeddings can recover a specific ontology-encoded relationship. The "Ont. Dist." column indicates the minimum path length between query and relevant segments in the base ontology (without enrichment shortcuts). Relation-specific KGE scoring operates via the corresponding materialized enrichment relation.

Task	Ont. Dist.	Enriched Relation	Relevance Criterion
Value			
Numeric Range	3+	`sharedMagnitudeBin`	Same concept with similar value (±20%) from different dimension
Cross-Dim. Transfer	3+	`sharedTaskType`	Same concept, task, and quality (±0.25) from different dimension
Statistical Profile	–	`hasMeanBin`	Similar mean and std. dev. within 20th percentile distance
Hierarchy			
Sibling Concept	2	`sharedConcept`	Same parent and same knowledge concept
KG Membership	4	`sharedKnowledgeGroup`	Same KnowledgeGroup via indirect path
KG Hierarchy	6+	`sharedKnowledgeGroup`	Sibling groups under same parent group
Provenance			
Provenance Chain	3–9	`provenanceLinked`	Same propagation chain
Deep Prov. (Mat.)	5+	`sharedRun`	Shared MLS run with materialized shortcuts
Deep Prov. (Raw)	5+	`sharedRun`*	Shared MLS run without shortcuts
Causal			
Temporal Causal	2+	`causallyLinked`	Causal predecessors in temporal chain
Multi-Seg. Relation	2	`sharedMultiSegRel.`	Same n-ary SegmentRelation
Cross-Knowledge			
Context + Concept	3+	`sharedContext`	Same scenario context and knowledge concept
Baseline			
Text Baseline	–	`sharedConcept`	Same concept as title-only lower bound

*Trained on graph with `enrich:sharedRun` triples removed (controlled experiment).

same concept and similar value from a different dimension, which prevents trivial structural-proximity solutions. Cross-Dimension Transfer additionally demanded matching algorithm task type and quality bracket, while Statistical Profile identified segments with similar distributions based on normalized Euclidean distance using a 20th percentile threshold, operating independently of concept. Together, these tasks established whether value-aware retrieval was feasible from graph structure alone.

The *hierarchy tasks* formed a progression of increasing structural depth. Sibling Concept (distance 2) required both a shared parent and the same knowledge concept, which tests conjunctive pattern learning because the enrichment captures only concept co-occurrence. KG Membership (distance 4) tested recovery of indirect group affiliation, while KG Hierarchy (distance 6+) required distinguishing sibling groups under the same parent group from mere co-group membership. This progression shows how enrichment precision interacts with task complexity, because the same enrichment relation (`sharedKnowledgeGroup`) serves both KG Membership, where it captures the criterion exactly, and KG Hierarchy, where it provides only a superset signal.

The *provenance tasks* tested the ability to learn derivation chains. Provenance Chain asked models to find segments connected through multi-step `semts:inputKnowledge` paths spanning 4 to 9 hops. The Deep Provenance pair formed a controlled experiment. Both variants used identical ground truth based on shared MLS runs, but the Materialized variant included `enrich:sharedRun` shortcuts while the Raw variant excluded them entirely, which directly measures the contribution of materialization to retrieval performance.

The *causal and relational tasks* tested the recovery of multi-segment patterns. Temporal Causal required finding predecessors that were both temporally before and causally linked to the query segment. Multi-Segment Relation asked models to identify all segments participating in the same n-ary `semts:SegmentRelation` entity. Finally, the *Context + Concept* task required matching across disjoint knowledge subclasses, and the *Text Baseline* verified that generic segment titles did not leak structural information. This task is trivially solvable from structure but should not be solvable from titles alone.

Table 5.3: Link prediction results (mean ± std across all runs). Best values per column in bold.

Model	MRR	Hits@1	Hits@3	Hits@10
RotatE	$\mathbf{0.390 \pm 0.022}$	**0.269**	**0.455**	0.605
BoxE	0.290 ± 0.015	0.111	0.392	**0.609**
DistMult	0.100 ± 0.035	0.035	0.096	0.227
TransE	0.097 ± 0.008	< 0.001	0.114	0.286
RelFreqBaseline	0.017 ± 0.001	0.004	0.014	0.036

Results and Analysis

Link Prediction. Table 5.3 and Figure 5.8 report link prediction performance. RotatE achieved the highest MRR (0.390) and Hits@1 (0.269), followed by BoxE with MRR 0.290. Both TransE and DistMult trailed with MRR at or below 0.10, although all models clearly beat the relation-frequency baseline (MRR 0.017). BoxE reached Hits@10 of 0.609, just ahead of RotatE's 0.605, even though its MRR was much lower. This pattern of reliable top-10 placement together with weaker top-1 ranking matches region-based scoring, which assigns similar scores to all entities inside a box boundary and therefore favors precision@10 retrieval over MRR. The same gap between link-prediction rank and retrieval quality reappears throughout the retrieval results below.

Retrieval Performance. Table 5.4 summarizes adjusted precision@10 across all runs, with complementary views in Figures 5.9 and 5.10. Across the full task set, the structural methods (KGE or Node2Vec) achieved the best result on 10 of 13 tasks. Among individual methods, RotatE and BoxE each won or tied on 5 tasks, Node2Vec won 2 (Sibling Concept and Context + Concept), and the text family won 3, namely Flat on Cross-Dimension Transfer and Deep Provenance Raw and Text on Statistical Profile. In aggregate, BoxE attained the highest mean (0.381), ahead of RotatE (0.346), TransE (0.326), and Node2Vec (0.240). The text encoders clustered near 0.11 to 0.12, and Text at 0.055 confirmed that titles alone carry little signal.

Both RotatE and BoxE reached near-perfect performance whenever an enrichment relation captured the ground-truth criterion exactly, for example about 0.91 on

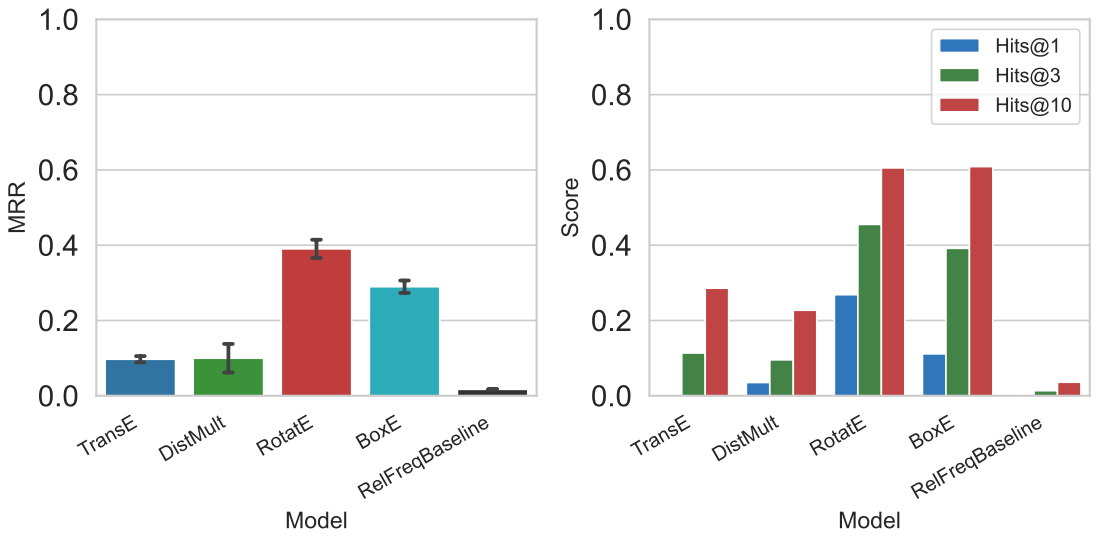


Figure 5.8: Link prediction performance across KGE models. Left: MRR with standard deviation bars. Right: Hits@k for $k \in \{1, 3, 10\}$.

KG Membership and above 0.94 on Deep Provenance Materialized. Performance dropped when the enrichment provided only a partial or superset signal. On Provenance Chain it settled around 0.57 because `enrich:provenanceLinked` encodes chain co-membership but spans larger and more diffuse relevance sets. The hierarchy tasks showed the pattern most clearly. KG Membership at distance 4 reached 0.91 because the enrichment captures co-membership exactly, Sibling Concept at distance 2 reached at most 0.168 for KGE because the enrichment misses the shared-parent constraint, and KG Hierarchy at distance 6+ reached an intermediate 0.643 because the enrichment is only a superset signal. Performance therefore tracks how well the enrichment relation aligns with the task criterion rather than the ontological distance. Text+ reached only 0.097 on KG Membership because group membership is relational rather than attributional.

The relation-agnostic Node2Vec embedding narrowly led on Sibling Concept (0.198 against 0.168 for the best KGE) and Context + Concept (0.133 against 0.116). Both criteria are conjunctive and local, combining a shared parent with a shared concept in the first case and a shared scenario context with a shared concept in the second, and the matching enrichment relations encode only one of the two conditions. Node2Vec was therefore strongest exactly where the

Table 5.4: Retrieval evaluation results (adjusted precision@10, mean across all runs) on the *full* graph variant. Best result per task in bold. KGE, Node2Vec, and text families are shown, with hybrid fusion reported separately in Table 5.6.

Task	TransE	DistMult	RotatE	BoxE	Node2Vec	Text	Text+	Flat
Value-based								
Numeric Range	0.152	0.010	**0.264**	0.262	0.052	0.007	0.067	0.054
Cross-Dim. Transfer	0.065	0.032	0.043	0.064	0.061	0.001	0.130	**0.144**
Statistical Profile	0.191	0.077	0.066	0.309	0.264	**0.316**	0.094	0.087
Hierarchy								
Sibling Concept	0.168	0.013	0.064	0.156	**0.198**	0.051	0.091	0.068
KG Membership	0.845	0.256	0.908	**0.911**	0.570	0.033	0.097	0.101
KG Hierarchy	0.552	0.225	**0.643**	0.607	0.303	0.036	0.041	0.039
Provenance								
Provenance Chain	0.549	0.269	**0.571**	0.566	0.477	0.148	0.136	0.092
Deep Prov. (Mat.)	0.802	0.294	0.945	**0.953**	0.524	0.012	0.277	0.303
Deep Prov. (Raw)	0.015	0.017	0.014	0.016	0.081	0.012	0.277	**0.303**
Causal & Relational								
Temporal Causal	0.164	0.154	**0.166**	**0.166**	0.147	0.027	0.025	0.018
Multi-Seg. Relation	0.186	0.132	**0.197**	**0.197**	0.155	0.031	0.021	0.020
Cross-Knowledge & Baseline								
Context + Concept	0.114	0.022	0.031	0.116	**0.133**	0.026	0.101	0.048
Text Baseline	0.436	0.028	0.588	**0.629**	0.156	0.020	0.215	0.205
Mean	0.326	0.118	0.346	**0.381**	0.240	0.055	0.121	0.114

materialization was too coarse for the criterion, and it ranked fourth overall (mean 0.240), behind BoxE and RotatE but above DistMult and every text variant. It still trailed the enrichment-aligned KGE models where a single relation captured the criterion exactly, for example 0.570 against 0.911 on KG Membership.

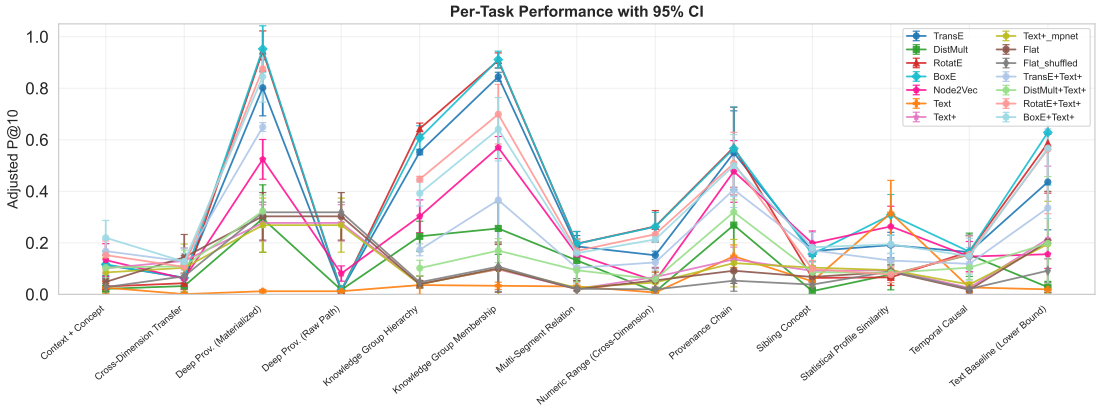


Figure 5.9: Retrieval performance (adjusted precision@10) across all tasks with 95% CI (mean $\sigma = 0.024$).

Among the KGE models, BoxE achieved the highest mean (0.381 against 0.346 for RotatE), despite RotatE's higher link-prediction MRR. BoxE led RotatE on tasks whose relevance needs several simultaneous criteria while the enrichment captures only a subset, namely Statistical Profile (0.309 against 0.066), Sibling Concept (0.156 against 0.064), and Context + Concept (0.116 against 0.031), where box geometry represents constraint intersections directly. RotatE in turn led on KG Hierarchy (0.643 against 0.607), where chaining rotations along hierarchical paths helps. On Temporal Causal and Multi-Segment Relation the two models scored identically. BoxE was thus the most robust single KGE method, winning or tying on 5 of 13 tasks and never collapsing on a structural task.

Text methods won on Cross-Dimension Transfer (Flat: 0.144), Deep Provenance Raw (Flat: 0.303), and Statistical Profile (Text: 0.316). On Cross-Dimension Transfer, relevance depends on matching concept, task type, and quality bracket, which are local properties that map well to string similarity. KGE scored lower here because the enrichment relation `sharedTaskType` captures only one dimension of the criterion. On Statistical Profile, where relevance is a purely numeric distribution-similarity criterion independent of concept, the title-only Text encoder edged out even BoxE (0.309), which suggests that the discretized statistical features are recoverable from the encoded numeric bins without any relational scoring. Deep Provenance Raw simply reflects the designed absence of the required enrichment relation. Both Flat and Text+ did poorly

on topology-dependent tasks because local property descriptions cannot express structural relationships. Flat and Text+ performed about equally on most tasks, and Flat_shuffled (mean 0.094) only slightly underperformed Flat (0.114), so Sentence-BERT is largely insensitive to field order and the Flat signal reflects content rather than a serialization artifact.

Impact of Materialization. The Deep Provenance comparison gave the most direct single-task evidence. With `enrich:sharedRun` present, BoxE reached 0.953. Without it, in the Raw variant, all KGE models fell to about 0.014 to 0.017, a roughly 60× drop caused by removing one relation type, because that relation was then absent from the model vocabulary and the rankings became effectively random. Node2Vec fell far less, from 0.524 to 0.081, because it embeds raw topology rather than the removed scoring axis. Text+ (0.277) and Flat (0.303) stayed unchanged because they encode run information directly in the segment descriptions.

Generalizing the Deep Provenance experiment to all tasks, Table 5.5 reports performance with and without the enrichment shortcuts. Removing every `enrich:` relation dropped all four KGE models to near-random levels (0.016 to 0.018 on average across the 12 standard tasks), which confirms that materialization is a precondition for relation-specific retrieval in general and not only in the deep-provenance setting. Node2Vec kept about 45% of its performance (from 0.253 to 0.114) because raw topology stays informative without explicit shortcuts, and the text methods were unaffected by construction. The practical reading is that any structural pattern KGE should retrieve needs a matching materialized relation, while the topology-based and text-based methods give a usable fallback when the materialization layer is incomplete. In this sense, the `enrich:` namespace acts as a declarative index that fixes which structural patterns the system can retrieve.

Depth Invariance of Retrieval. Looking at provenance retrieval as a function of chain hop depth showed that materialization removes the usual depth penalty. Instead of decaying with distance, KGE retrieval quality stayed flat or rose with depth. RotatE climbed from about 0.46 at the shallowest depth to about 0.80

Table 5.5: Shortcut ablation: adjusted precision@10 on the *full* graph (with `enrich:` shortcuts) versus the *no-shortcuts* variant (all enrichment relations removed), averaged across the 12 standard retrieval tasks. KGE models collapse to near-random when shortcuts are removed, whereas the text methods are robust by construction.

	TransE	**DistMult**	**RotatE**	**BoxE**	**Node2Vec**	**Text+**	**Flat**
Full (with shortcuts)	0.352	0.126	0.374	0.411	0.253	0.108	0.098
No shortcuts	0.016	0.016	0.017	0.018	0.114	0.108	0.098
Absolute change	−0.336	−0.110	−0.357	−0.393	−0.139	0.000	0.000

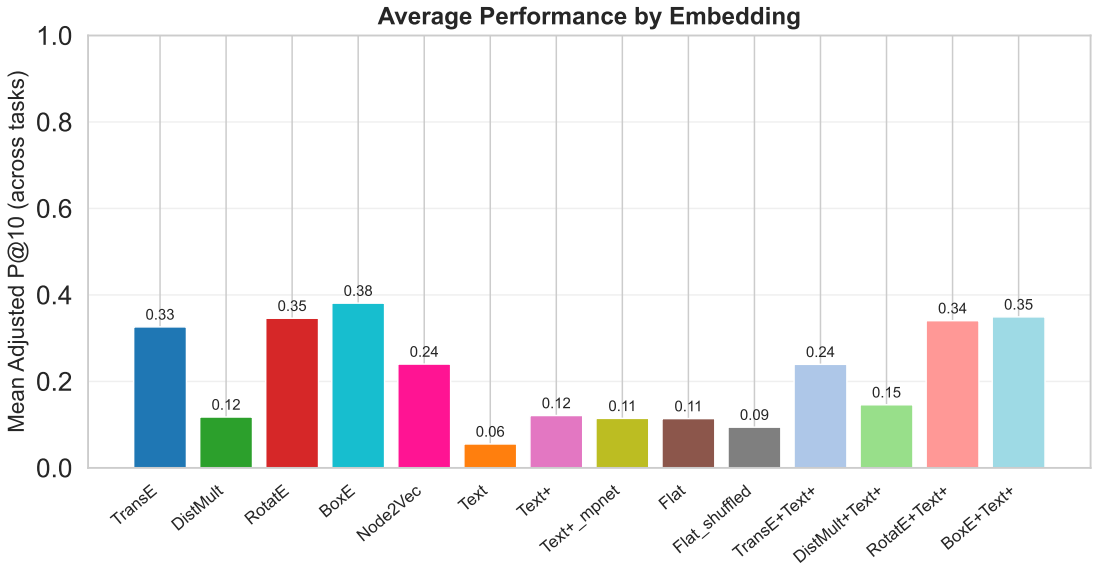


Figure 5.10: Average adjusted precision@10 across all 13 tasks per embedding method.

at the deepest, BoxE tracked it closely, and Node2Vec followed the same upward trend at a lower level. The text methods stayed low and almost flat, between about 0.06 and 0.24 across all depths. Each materialized shortcut collapses a multi-hop chain into a single edge, so deep relationships are no harder to retrieve than shallow ones, and often easier because deeper chains produce larger and more distinctive relevant sets. The value of materialization therefore grows with the structural distance it bridges.

Table 5.6: Score-level hybrid fusion (KGE + Text+, $\alpha = 0.5$) versus the corresponding KGE base model, mean adjusted precision@10 across the 12 standard tasks, plus selected mixed-criteria tasks. Equal-weight fusion helps only the weakest base model and on multi-criteria tasks, while degrading the strongest models overall.

Model	Type	Mean AP@10	Context+ Concept	Numeric Range	KG Membership
TransE	Base	**0.352**	0.114	**0.152**	**0.845**
	Hybrid	0.240	**0.168**	0.124	0.366
DistMult	Base	0.126	0.022	0.010	**0.256**
	Hybrid	**0.146**	**0.104**	**0.059**	0.170
RotatE	Base	**0.374**	0.031	**0.264**	**0.908**
	Hybrid	0.340	**0.152**	0.233	0.699
BoxE	Base	**0.411**	0.116	**0.262**	**0.911**
	Hybrid	0.349	**0.220**	0.213	0.641

Encoder Capacity. Upgrading the Text+ encoder from `all-MiniLM-L6-v2` to the higher-capacity `all-mpnet-base-v2` did not improve retrieval. Text+_mpnet reached a mean of 0.115, indistinguishable from Text+ at 0.121 and slightly below it on most tasks. The limiting factor for text-based graph retrieval is therefore representational rather than encoder capacity, because relational structure that is absent from the serialized description cannot be recovered by a stronger language encoder.

Equal-Weight Fusion. Table 5.6 evaluates the score-level hybrids. Equal-weight ($\alpha = 0.5$) fusion redistributed performance rather than improving it uniformly. It helped the weakest base model (DistMult: 0.126 to 0.146) and lifted models on genuinely mixed-criteria tasks, most clearly Context + Concept, where BoxE+Text+ reached 0.220 against 0.116 for BoxE alone. It nonetheless lowered the strong models overall, with BoxE going from 0.411 to 0.349, RotatE from 0.374 to 0.340, and TransE from 0.352 to 0.240, because the task-agnostic text component dilutes the sharp relation-aligned signal on pure-structural tasks such as KG Membership (BoxE 0.911 to 0.641). Fusion should therefore be applied selectively, weighted toward text only on attribute-driven or multi-criteria queries, instead of as a blanket combination.

Table 5.7: Structure advantage comparing the best structured embedding (KGE, Node2Vec, or Text+) against the Flat key-value embedding. Positive Δ values indicate that leveraging ontological structure improves retrieval. Tasks sorted by absolute advantage.

Task	Best Structured	Value	Flat	Δ
Knowledge Group Membership	BoxE	**0.911**	0.101	+0.810
Deep Prov. (Materialized)	BoxE	**0.953**	0.303	+0.650
Knowledge Group Hierarchy	RotatE	**0.643**	0.039	+0.604
Provenance Chain	RotatE	**0.571**	0.092	+0.479
Text Baseline	BoxE	**0.629**	0.205	+0.424
Statistical Profile	BoxE	**0.309**	0.087	+0.222
Numeric Range	RotatE	**0.264**	0.054	+0.210
Multi-Segment Relation	RotatE	**0.197**	0.020	+0.177
Temporal Causal	RotatE	**0.166**	0.018	+0.148
Sibling Concept	Node2Vec	**0.198**	0.068	+0.130
Context + Concept	Node2Vec	**0.133**	0.048	+0.085
Cross-Dimension Transfer	Text+	0.130	**0.144**	−0.014
Deep Prov. (Raw)	Text+	0.277	**0.303**	−0.026

Structure Advantage. Table 5.7 quantifies the benefit of encoding ontological structure as the absolute difference in adjusted precision@10 between the best structured method and Flat. On 11 of 13 tasks, structure gave a mean absolute improvement of +0.358. The largest gains appeared where the enrichment captures the criterion precisely, most notably Knowledge Group Membership (+0.810) and Deep Provenance Materialized (+0.650), which mainly reflects the contribution of materialization. The near-zero Flat baselines there confirm that key-value serialization carries essentially no signal for topology-dependent retrieval. On the two conjunctive-local tasks the best structured representation was Node2Vec rather than a KGE model. The two negative cases were the purely property-based Cross-Dimension Transfer and the controlled absence of enrichment in Deep Provenance Raw.

Feature Ablation. Table 5.8 shows task-specific dependencies in Text+ performance. Numeric Range benefited from knowledge features alone (0.218 against 0.058 for the full configuration) because concept is the main discriminator and

Table 5.8: Text+ feature ablation (mean adjusted precision@10 across the three instance graphs). Each row removes or isolates specific feature groups from the natural language description. Best value per column in bold.

Configuration	Numeric Range	Provenance Chain	Cross-Dimension
Full (all features)	0.058	0.112	0.130
Knowledge only	**0.218**	0.050	0.065
Knowledge + Quality	0.107	0.034	0.110
Knowledge + Provenance	0.192	0.047	0.074
No Knowledge	0.023	0.079	0.047
No Provenance	0.053	0.119	0.125
No Quality	0.072	**0.133**	**0.149**

Configuration	Context + Concept	Deep Provenance	KG Membership
Full (all features)	0.097	0.281	0.094
Knowledge only	**0.138**	0.024	0.113
Knowledge + Quality	0.074	0.156	0.085
Knowledge + Provenance	0.116	0.022	**0.125**
No Knowledge	0.020	**0.305**	0.098
No Provenance	0.084	0.297	0.095
No Quality	0.102	0.270	0.098

extra features dilute the signal. Deep Provenance instead benefited from removing knowledge features (0.305 against 0.281) because shared runs correlate with algorithm metadata rather than concept identity. Context + Concept peaked with knowledge features alone (0.138), and knowledge combined with provenance stayed competitive (0.116), which confirms that concept identity is the dominant cross-knowledge signal. Removing quality features was best for Provenance Chain (0.133) and Cross-Dimension (0.149) because quality is orthogonal to these criteria. The full configuration was reasonable but not optimal for any single task, so task-specific feature selection could further improve text-based retrieval.

Summary, Limitations, and Future Directions

Comparing the different families, relation-specific KGE models were best at recovering indirect relational patterns such as group membership via `semts:groupKnowledge`, provenance chains via `semts:inputKnowledge`, and

n-ary relations via `semts:multiSegmentKnowledge`, given materialization. Conjunctive criteria that need several simultaneous local conditions, like in the tasks Sibling Concept and Context + Concept, were instead handled best by the relation-agnostic Node2Vec embedding. Property-based similarity over attribute values was in turn served best by text serialization, as in Cross-Dimension Transfer and Statistical Profile, where relevance maps naturally to string or numeric-bin similarity but poorly to any single graph relation. With six evaluation points per task and embedding combination (three graphs $\times$ two seeds), between-graph variability was comparable to within-graph variability (mean $\sigma \approx 0.024$), so the per-task results were stable across instantiations. Given this sample size, we treat the per-task means as descriptive and base our conclusions on the consistent effects, in particular the materialization ablations and the gap between structured and text methods on relational tasks.

Implications for Graph RAG Systems. These empirical findings yield several practical guidelines for Graph RAG deployment. Critically, hybrid retrieval should be applied selectively rather than as a default ensemble across all queries. Because arbitrary score fusion dilutes structural signals on purely relational queries, routing mechanisms should isolate text embeddings to attribute-heavy tasks while reserving KGE for topological traversal. Materialization planning must similarly balance hop-depth against shortcut precision, ensuring precomputed shortcuts do not introduce spurious false positives. Standard link-prediction metrics such as MRR do not accurately reflect retrieval quality: RotatE is strongest at recovering the single top result, whereas BoxE is more effective when retrieving multiple relevant results. Given this mismatch, a phased rollout is appropriate, in which simpler methods such as Node2Vec or text-based serializations provide initial results while more complex embedding models are integrated as they mature. Encoder capacity likewise does not scale proportionally, and simple key-value text formats perform comparably to complex graph-to-text models, so text pipelines can remain lightweight and inexpensive to run.

Conclusion and Outlook. The evaluation showed how different embedding families coped with the distinct aspects of the SemTS ontology. It established a clear aspect-to-geometry mapping in which no single family dominates every category, which provides evidence that ontological aspects are heterogeneous enough to require different representational geometries. It also showed that this mapping is enabled rather than intrinsic, because the relation-specific models recovered deep multi-hop patterns only once materialization flattened them into direct edges. The absolute scores should thus not be interpreted as proof that KGE captures the ontology. Instead, they showed that KGE retrieves accurately along the relations it is provided, and materialization is the necessary precondition that supplies those relations. These conclusions hold within a controlled scope. Since the instance graphs are synthetically generated, some results may partly reflect the generator rather than the ontology itself. On Statistical Profile, for instance, the title-only Text encoder slightly exceeded BoxE (0.316 against 0.309), consistent with the discretized statistical features being recoverable directly from the encoded numeric bins. A real-world SemTS deployment would help separate these effects. The most informative next steps follow directly from the findings. Inductive, path-based reasoners such as NBFNet [Zhu+21] and A*Net [Zhu+23] can be run on the no-shortcuts variant to test whether materialization is intrinsic to the ontology or an artifact of shallow scoring. Knowledge-graph foundation models such as ULTRA [Gal+24] and GFM-RAG [Luo+25] can show whether a single pretrained reasoner subsumes the per-aspect specialization seen here. Extending the protocol end-to-end through Graph RAG [Edg+24; Pen+24] would check whether the retrieval advantages persist once a generator consumes the context. The fixed-α fusion can be replaced by a learned, query-conditioned router. Finally, literal-aware models such as LiteralE [Kri+19] can target the value-based aspects that remain hardest, ideally on a real SemTS deployment where materialization becomes a cost-aware planning decision rather than a precondition.

5.4 Knowledge Propagation and Contextual Reasoning

Apart from common knowledge generation patterns (see Section 4.4.3), time series analysis does not always constitute a single, isolated task. Instead, analytical workflows often involve incremental, multi-stage processes at different hierarchical levels such as whole time series collections and single segments. For instance, a common pattern involves first applying changepoint detection to partition a time series into multiple segments, then extracting statistical features from each segment, and finally clustering these segments based on their feature profiles. Similarly, for the same segmentation one might perform anomaly detection for a specific segment of interest rather than the entire time series. Figure 5.11 visualizes the analytical pipelines of these examples.

An even richer form of knowledge propagation emerges when data-derived insights are combined with domain-specific contextual information. This combination enables not merely propagation chains but full reasoning chains, where analytical outputs are contextualized and validated through expert knowledge or established domain rules.

The following subsections discuss the implementation of such propagation pipelines and reasoning processes derived from or based on SemTS. For a more general formalization and validation of knowledge propagation pipelines, Section 5.4.1 introduces a many-sorted algebra. Afterwards Section 5.4.2 demonstrates contextual inference over time series data by combining analytical insights with domain knowledge to identify causal relationships and to derive semantically enriched conclusions.

5.4.1 Knowledge Propagation Algebra

As in SemTS segments can be nested, activities can consume and produce multiple inputs and outputs, and knowledge can be grouped hierarchically, SemTS can represent arbitrarily complex propagation pipelines. In practice, there is no single set of “typical” pipelines.

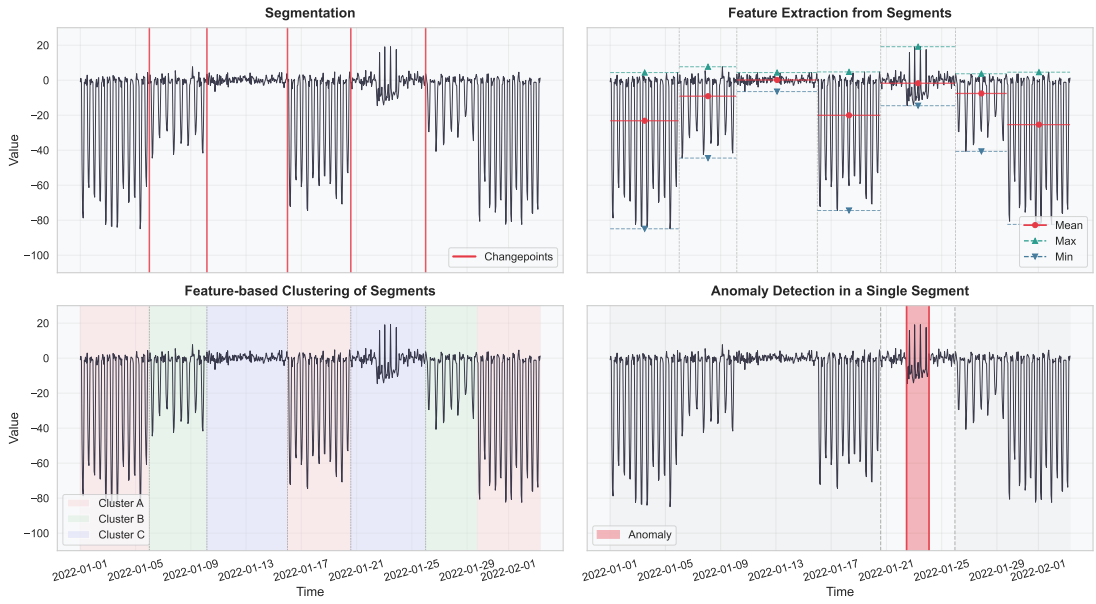


Figure 5.11: Example of incremental knowledge propagation [Gra+23]: (top left) segmentation, (top right) feature extraction from segments, (bottom left) feature-based clustering of segments, (bottom right) anomaly detection in a single segment.

To validate and reason about such rich structures, we introduce a many-sorted algebraic specification [ST12] as an abstract representation of the SemTS ontology. This algebra does not try to encode every aspect of the ontology. Instead, it concentrates on core notions that are central for knowledge propagation: segments, knowledge, inputs and outputs, and knowledge-generation activities with PROV-based provenance.

Hence, the algebra specifies the essential elements of knowledge propagation in a mathematical form. More detailed and SemTS-specific constraints, such as the distinction between single-segment and multi-segment knowledge, the exact class hierarchies, and cardinality restrictions are expressed in the ontology itself and can be checked with OWL reasoners and SHACL shapes. The algebra rather equals a formalization, helping to trace, validate and design consistent patterns of knowledge propagation.

The formalization conceptually intersects with several established theoretical frameworks. While algebraic provenance tracking has been explored in relational

database theory [GKT07], our approach adapts these concepts for hierarchical knowledge graphs and dynamic semantic pipelines represented by SemTS. Furthermore, unlike traditional belief revision models such as the AGM postulates [AGM85] which manage the retraction of contradicting facts, our algebra is strictly monotonic. It treats knowledge accumulation as an append-only process, aligning with reproducible scientific workflow models [Lud+06] where historical data states are preserved rather than overwritten. The core concepts of our algebra include:

- **Traceable propagation**: Every evaluation creates a *fresh* activity instance with PROV-style links to the inputs it used and outputs it generated. This makes pipelines auditable and reproducible.
- **Monotone knowledge growth**: Feeding never removes knowledge. For any fixed set of outputs, commutativity, idempotence, and batch decomposition of feeding ensure you can merge them in any order or batch without changing the resulting knowledge state.
- **Hierarchical knowledge with optional flattening**: Knowledge groups preserve structure or can be used for flat propagation of individual knowledge items. Group-preserving propagation favors structural coherence, while flat propagation favors scope precision at the atomic level.
- **Segment-safe distribution**: Scope definitions ensure that only knowledge relevant to a segment is fed back to that segment.
- **Multi-input/multi-output**: Activities can consume many inputs jointly and produce many outputs in one run, while provenance is preserved.

In the following we specify a many-sorted algebraic specification treating involved SemTS entities as different sorts, and all operations on them in a signature. In addition, defined axioms formalize the intended behavior of the operators, to reason about knowledge propagation in a consistent manner. Supplementary material is available in Appendix E.

Sorts

Notation. $\Phi(X)$ denotes all finite subsets of a sort X. We write $\Phi_+(X)$ for the set of all finite, nonempty subsets of X, regarded as a subsort: $\Phi_+(X) \subseteq \Phi(X)$. We further use $\subseteq$ for subset, $\cup$ for union, $\varnothing$ for the empty set, and $\dot\cup$ for disjoint union. Operators with tuple-valued codomains are shorthand for operators into auxiliary product sorts.

Sorts Σ are defined as follows:

$$\mathbf{\Sigma} \;=\; \big\{\,\mathrm{Seg},\ \mathrm{Know},\ \mathrm{KAtom},\ \mathrm{KGroup},\ \mathrm{KGI},\ \mathrm{KGO},\ \mathrm{KGE},\ \mathrm{N}\,\big\}.$$

Seg: A *time series segment*. References data as well as associated knowledge.

KAtom: An *atomic* knowledge item that itself is not a group.

KGroup: A *knowledge group*: a hierarchical container that may hold atoms and/or other groups. It is itself defined as knowledge.

Know: Anything we consider *knowledge*. This is the category that includes both single, atomic knowledge items and knowledge groups. It holds that

$$\mathrm{Know} \;=\; \mathrm{KAtom} \,\dot\cup\, \mathrm{KGroup}.$$

KGI: A *knowledge-generation input*: points to exactly one segment.

KGO: A *knowledge-generation output*: carries newly produced knowledge (atoms and/or groups) *together with a reference to the activity run that generated it*. Outputs are thus provenance-bearing entities with identity: two runs that produce identical knowledge yield two *distinct* outputs.

KGE: A *knowledge-generation entity*, which is an activity, such as a method run, reasoning step, or comparison, that consumes inputs and produces outputs. Each run of an activity is represented by a fresh KGE instance, built explicitly by the run constructor mkRun below, to support precise provenance.

N: Natural numbers, used as an explicit name supply (counter) for run instances.

Signature

For the signature S we define the subsequent operators on Σ. A consolidated reference of these operators, grouped by their operational roles, is provided in Table 5.9.

In the following, we detail the specific semantics and design motivations for each of these mathematical categories:

Input and Output Operators

Constructors and accessors:

$$\begin{aligned}
\textbf{mkIn} &: \mathrm{Seg} \times \Phi(\mathrm{Know}) \ \to \ \mathrm{KGI}, \\
\textbf{inSeg} &: \mathrm{KGI} \ \to \ \mathrm{Seg}, \\
\textbf{inKnow} &: \mathrm{KGI} \ \to \ \Phi(\mathrm{Know}), \\
\textbf{mkOut} &: \mathrm{KGE} \times \Phi_+(\mathrm{Know}) \ \to \ \mathrm{KGO}, \\
\textbf{outKnow} &: \mathrm{KGO} \ \to \ \Phi_+(\mathrm{Know}), \\
\textbf{runOf} &: \mathrm{KGO} \ \to \ \mathrm{KGE}.
\end{aligned}$$

Semantics:

- $\mathrm{mkIn}(s, K)$ creates an input tied to segment s that currently carries knowledge K.
- $\mathrm{inSeg}(i)$ returns which segment input i refers to.
- $\mathrm{inKnow}(i)$ returns all knowledge currently attached to input i.
- $\mathrm{mkOut}(r, K)$ creates an output generated by run r that carries the produced knowledge K (nonempty).
- $\mathrm{outKnow}(o)$ returns the produced knowledge carried by output o.

Table 5.9: Quick Reference Guide for the Knowledge Propagation Signature

Operational Role	**Operator & Algebraic Type**	**Description**
Input / Output Carriers	**mkIn** : Seg × Φ(Know) → KGI	Creates a segment-tied input state
	inSeg : KGI → Seg	Extracts the segment from an input
	inKnow : KGI → Φ(Know)	Retrieves current knowledge payload
	mkOut : KGE × Φ_+(Know) → KGO	Creates a provenance-bearing output
	outKnow : KGO → Φ_+(Know)	Retrieves newly produced knowledge
	runOf : KGO → KGE	Extracts the generating activity run
Hierarchy & Scope	**mkGroup** : Φ_+(Know) → KGroup	Bundles knowledge hierarchically
	members : KGroup → Φ_+(Know)	Lists direct group components
	atomsOf : Know → Φ(KAtom)	Resolves items down to basic facts
	flatten : Φ(Know) → Φ(KAtom)	Unpacks all nested groups to atoms
	scopeOf : Know → Φ(Seg)	Identifies relevant segments
Provenance Trackers	**mkRun** : KGE × Φ_+(KGI) × N → KGE	Instantiates a fresh execution trace
	used : KGE → Φ(KGI)	Returns consumed inputs of a run
	generated : KGE → Φ(KGO)	Returns produced outputs of a run
	derivedFrom : KGO → Φ(KGI)	Traces output lineage back to inputs
Dynamic State Transformers	**feed** : Φ(KGO) × KGI → KGI	Scope-safe, structure-preserving merge
	feedFlat : Φ(KGO) × KGI → KGI	Scope-safe, atom-flattening merge
	eval : KGE × Φ_+(KGI) × N → KGE × Φ_+(KGO) × N	Executes a dynamic pipeline step

- runOf(o) returns the run that generated output o.

These operations are the basic "data carriers" for propagation: inputs bundle a segment with known knowledge, outputs bundle newly produced knowledge with the run that produced it. An output is therefore *not* reducible to its knowledge content, mirroring the identity that PROV entities possess on the RDF level (IRIs).

Knowledge Hierarchy Operators

Group constructor and accessors:

$$\begin{aligned} \textbf{mkGroup} &: \Phi_+(\text{Know}) \ \rightarrow \ \text{KGroup}, \\ \textbf{members} &: \text{KGroup} \ \rightarrow \ \Phi_+(\text{Know}), \\ \textbf{atomsOf} &: \text{Know} \ \rightarrow \ \Phi(\text{KAtom}), \\ \textbf{flatten} &: \Phi(\text{Know}) \ \rightarrow \ \Phi(\text{KAtom}). \end{aligned}$$

Semantics:

- mkGroup(K) builds a knowledge group from a *nonempty* set K of knowledge.
- members(g) lists the direct, non-transitive members of group g.
- atomsOf(x) gives all *atomic* facts inside x: if x is atomic, it returns $\{x\}$; if x is a group, it recursively gathers all atoms from its possibly nested members.
- flatten(K) unions atomsOf over all items in K. It is the set of all atomic facts reachable from K.

Scope Operator

$$\textbf{scopeOf} : \text{Know} \ \rightarrow \ \Phi(\text{Seg}).$$

Semantics: scopeOf(k) returns the set of segments that knowledge item k is about. It controls where k is allowed to be attached during feeding.

Remark. scopeOf has a two-level status. On *atoms* it is *interpretation-given* (ground): in any concrete SemTS instance the scope of an atomic knowledge item is fixed at creation time via the ontology's object properties and is not subject to rewriting. On *groups* it is a *defined* operation, determined by the hierarchy equation of Axiom A3 and computed by the rewrite system (rule R13 in Appendix E). Since the group-scope equation is not expressible as an OWL axiom, conforming SemTS instances should additionally validate it on the RDF level, e.g., via a SHACL shape.

Provenance Operators

$$\begin{aligned} \textbf{mkRun} &: \mathrm{KGE} \times \Phi_+(\mathrm{KGI}) \times \mathrm{N} \;\to\; \mathrm{KGE}, \\ \textbf{used} &: \mathrm{KGE} \;\to\; \Phi(\mathrm{KGI}), \\ \textbf{generated} &: \mathrm{KGE} \;\to\; \Phi(\mathrm{KGO}), \\ \textbf{derivedFrom} &: \mathrm{KGO} \;\to\; \Phi(\mathrm{KGI}). \end{aligned}$$

Semantics:

- $\mathrm{mkRun}(e, I, n)$ is the run instance of activity e on input batch I under counter n. It is a *constructor* and injective (A1). A run therefore uniquely determines its activity, its consumed inputs, and its counter.
- $\mathrm{used}(e')$ is "which inputs e' actually consumed". On run instances it is *derivable*: $\mathrm{used}(\mathrm{mkRun}(e, I, n)) = I$.
- $\mathrm{generated}(e')$ is "which outputs e' produced". It is loosely specified together with eval and pinned down by Axiom A4.
- $\mathrm{derivedFrom}(o)$ lists the nonempty inputs that o was derived from. Unless an interpretation supplies a finer, conforming assignment, it defaults to the full input batch of the generating run, $\mathrm{used}(\mathrm{runOf}(o))$.

Remark. On base activities (KGE elements that are not run instances), used and generated are unconstrained. Provenance is meaningful for runs only. In well-formed pipeline states, outputs are created exclusively by eval, so the run

argument of every output is indeed a run instance.

Core Propagation Operators

$$\begin{aligned}
\textbf{feed} &: \Phi(\mathrm{KGO}) \times \mathrm{KGI} \;\to\; \mathrm{KGI}, \\
\textbf{feedFlat} &: \Phi(\mathrm{KGO}) \times \mathrm{KGI} \;\to\; \mathrm{KGI}, \\
\textbf{eval} &: \mathrm{KGE} \times \Phi_+(\mathrm{KGI}) \times \mathrm{N} \;\to\; \mathrm{KGE} \times \Phi_+(\mathrm{KGO}) \times \mathrm{N}.
\end{aligned}$$

Semantics:

- $\mathrm{feed}(O, i)$ merges the knowledge from outputs O into input i, adding exactly those knowledge items whose scope contains $\mathrm{inSeg}(i)$, and preserving group structure.
- $\mathrm{feedFlat}(O, i)$ is like feed but first flattens groups in the outputs to their atomic members, so it adds only atomic knowledge relevant to $\mathrm{inSeg}(i)$.
- $\mathrm{eval}(e, I, n)$ runs activity e once on the batch of inputs I, consuming name-supply index n, and returns a new run instance e'_n together with all outputs O that this run produced, and the advanced counter $n{+}1$.

The algebra adds dynamic operators like *feed* and *eval* that describe how one consistent graph state can be transformed into another. The feed operation returns a new KGI rather than a KGO because a KGI represents the evolving knowledge state of a segment. Feeding advances this state, producing an updated KGI that can serve as input to subsequent activities.

Derived Utility Operators

$$\begin{aligned}
\textbf{propagate}(e, i, n) &\stackrel{\text{def}}{=} \text{let } (e', O, n') = \mathrm{eval}(e, \{i\}, n) \text{ in } \big(\mathrm{feed}(O, i),\, n'\big), \\
\textbf{propagateFlat}(e, i, n) &\stackrel{\text{def}}{=} \text{let } (e', O, n') = \mathrm{eval}(e, \{i\}, n) \text{ in } \big(\mathrm{feedFlat}(O, i),\, n'\big).
\end{aligned}$$

For $O \subseteq \mathrm{KGO}$, $I \subseteq \mathrm{KGI}$, $s \in \mathrm{Seg}$:

$$\begin{aligned} \mathbf{outK}(O) &\stackrel{\text{def}}{=} \bigcup_{o \in O} \mathrm{outKnow}(o), \\ \mathbf{segs}(I) &\stackrel{\text{def}}{=} \{\, \mathrm{inSeg}(i) \mid i \in I \,\}, \\ \mathbf{filterScope}(s, K) &\stackrel{\text{def}}{=} \{\, k \in K \mid s \in \mathrm{scopeOf}(k) \,\}. \end{aligned}$$

Semantics:

- propagate(e, i, n) is a convenience operation: run e on the single input i (consuming counter n) and then construct a new input for the same segment that includes all scope-compatible knowledge from the outputs, using feed.
- propagateFlat(e, i, n) is the flat variant: run e on i and then merge all outputs back using feedFlat, so that only atomic knowledge is added.
- outK(O) is "all produced knowledge in O" (the union of the knowledge carried by each output in O).
- segs(I) is "all segments referenced by I" (the set of segments that the inputs in I refer to via inSeg).
- filterScope(s, K) is "the knowledge from K relevant to segment s" (exactly those items in K whose scope contains s).

Axioms

All axioms hold for any $e \in \mathrm{KGE}$, $r \in \mathrm{KGE}$, $I \subseteq \mathrm{KGI}$, $i \in \mathrm{KGI}$, $O \subseteq \mathrm{KGO}$, $s \in \mathrm{Seg}$, $K, M \subseteq \mathrm{Know}$, $a \in \mathrm{KAtom}$, $g \in \mathrm{KGroup}$, $n \in \mathrm{N}$.

(A1) Constructor freeness and accessor recovery.

$$\begin{aligned}
\text{inSeg}(\text{mkIn}(s, K)) &= s, \\
\text{inKnow}(\text{mkIn}(s, K)) &= K, \\
\text{outKnow}(\text{mkOut}(r, K)) &= K, \\
\text{runOf}(\text{mkOut}(r, K)) &= r, \\
\text{members}(\text{mkGroup}(M)) &= M, \\
\text{used}(\text{mkRun}(e, I, n)) &= I.
\end{aligned}$$

Additionally, all constructors are *injective* (no confusion) and every element of a constructed sort is reachable by its constructor (no junk):

$$\begin{aligned}
\text{mkIn}(s_1, K_1) = \text{mkIn}(s_2, K_2) &\implies s_1 = s_2 \ \wedge \ K_1 = K_2, \\
\text{mkOut}(r_1, K_1) = \text{mkOut}(r_2, K_2) &\implies r_1 = r_2 \ \wedge \ K_1 = K_2, \\
\text{mkGroup}(M_1) = \text{mkGroup}(M_2) &\implies M_1 = M_2, \\
\text{mkRun}(e_1, I_1, n_1) = \text{mkRun}(e_2, I_2, n_2) &\implies e_1 = e_2 \ \wedge \ I_1 = I_2 \ \wedge \ n_1 = n_2.
\end{aligned}$$

$$\begin{aligned}
\forall\, i \in \text{KGI}\ \exists\, s, K :\ & i = \text{mkIn}(s, K), \\
\forall\, o \in \text{KGO}\ \exists\, r, K :\ & o = \text{mkOut}(r, K), \\
\forall\, g \in \text{KGroup}\ \exists\, M :\ & g = \text{mkGroup}(M).
\end{aligned}$$

Moreover, $\text{mkRun}(e, I, n)$ is distinct from every base activity. The sort KGE is intentionally *not* freely generated: its carrier contains interpretation-given base activities in addition to the run instances built by mkRun; only the no-confusion properties above are required.

Explanation: Inputs, outputs, groups, and runs are exact containers of what is put in. Injectivity ensures that distinct contents produce distinct entities. Surjectivity (no junk) ensures that every entity of a fully constructed sort has a canonical constructor representation. For KGO this means in particular that an output is determined by its generating run *and* its knowledge. Knowledge content alone does *not* identify an output. Together, these guarantee a free (initial) algebra for

the constructor sorts.

(A2) Groups are acyclic.

Let $\in^+$ be the transitive closure of direct membership via members. Then:

$$\neg\big(g \in^+ g\big).$$

Explanation: Groups cannot, directly or indirectly, contain themselves, which prevents infinite regress. Extensionality of groups, $\mathrm{mkGroup}(M) = \mathrm{mkGroup}(N) \Leftrightarrow M = N$, is exactly constructor injectivity together with congruence and is therefore already covered by A1. A2 records only the genuinely additional constraint.

(A3) Scope and flattening respect the hierarchy.

Scope of groups:

$$\mathrm{scopeOf}(g) \;=\; \bigcup_{x \in \mathrm{members}(g)} \mathrm{scopeOf}(x).$$

Flattening:

$$\begin{aligned} \mathrm{atomsOf}(a) &= \{a\}, \\ \mathrm{atomsOf}(g) &= \bigcup_{x \in \mathrm{members}(g)} \mathrm{atomsOf}(x), \\ \mathrm{flatten}(K) &= \bigcup_{x \in K} \mathrm{atomsOf}(x). \end{aligned}$$

Algebraic laws:

$$\begin{aligned} \mathrm{flatten}(K \cup K') &= \mathrm{flatten}(K) \cup \mathrm{flatten}(K'), \\ \mathrm{flatten}(\varnothing) &= \varnothing, \\ \mathrm{flatten}(\{x\}) &= \mathrm{atomsOf}(x). \end{aligned}$$

Explanation: Group scope is the sum of member scopes. Flattening distributes over unions and returns only atoms.

Remark. Since $\mathrm{scopeOf}(g) = \bigcup_{x \in \mathrm{members}(g)} \mathrm{scopeOf}(x)$, a group's scope is the union of its members' scopes. Consequently, feed may attach an entire group to

a segment even if only some of the group's members are individually relevant to that segment. When finer-grained scope precision is required at the cost of losing structural grouping, feedFlat should be used, as it evaluates scope at the atomic level. This design offers a deliberate trade-off between structural coherence and scope precision.

(A4) Evaluation yields a fresh, self-describing run instance and coherent provenance.

The interpretation provides a ground *knowledge-output function*

$$\text{kout} \;:\; \text{KGE} \times \Phi_+(\text{KGI}) \;\to\; \Phi_+\big(\Phi_+(\text{Know})\big),$$

assigning to every activity e and input batch I a nonempty set of nonempty knowledge bundles, one bundle per output. Like scopeOf on atoms, kout is interpretation-given and not subject to rewriting.

Let $\text{eval}(e, I, n) = (e', O, n')$. Then:

$$\begin{aligned}
&e' \;=\; \text{mkRun}(e, I, n), && \text{(run naming)}\\
&n' \;=\; n + 1, && \text{(counter advancement)}\\
&O \;=\; \big\{\, \text{mkOut}(e', K) \;\big|\; K \in \text{kout}(e, I) \,\big\}, && \text{(output identity)}\\
&\text{used}(e') \;=\; I, && \text{(provenance: used; derivable from A1)}\\
&\text{generated}(e') \;=\; O, && \text{(provenance: generated)}\\
&\forall\, o \in O :\; \varnothing \neq \text{derivedFrom}(o) \subseteq I. && \text{(derivation traceability)}
\end{aligned}$$

The following properties are *derived* rather than postulated:

$$(e_1, I_1, n_1) \neq (e_2, I_2, n_2) \implies \text{mkRun}(e_1, I_1, n_1) \neq \text{mkRun}(e_2, I_2, n_2) \quad \text{(freshness, by A1)}$$

$$\text{outK}(O) \;=\; \bigcup \text{kout}(e, I) \;\text{ depends only on } e \text{ and } I. \quad \text{(knowledge-level determinism)}$$

Non-triviality ($I \neq \varnothing$, $O \neq \varnothing$, and nonempty knowledge per output) is enforced by the sorts $\Phi_+(\cdot)$.

Explanation: Axiom A4 ensures that every evaluation yields a globally unique,

self-describing activity instance. Because pure term rewrite systems lack side effects (e.g., random UUIDs), freshness is achieved deterministically: the run constructor mkRun records the activity, the consumed input batch, and an explicit counter $n \in \mathbb{N}$ threaded through evaluation. Injectivity of mkRun (A1) then guarantees that distinct evaluations yield distinct run instances and distinct output entities, *even when the produced knowledge coincides*. The knowledge content of the outputs remains functionally deterministic in the activity and its inputs (via kout), cleanly separating *what* is produced from *when* and *by which run* it was produced. This algebraic treatment decouples formal reasoning about data provenance from the underlying execution environment's identity generation mechanism (counters, UUIDs, timestamps, etc.).

Remark (well-definedness of provenance). A run $\mathrm{mkRun}(e, I, n)$ uniquely determines (e, I, n), and an output $\mathrm{mkOut}(e', K)$ uniquely determines its run (A1). Hence used and generated are well-defined functions, and the traceability constraint on derivedFrom is satisfiable, e.g., by the coarsest conforming choice $\mathrm{derivedFrom}(o) = \mathrm{used}(\mathrm{runOf}(o))$, which the rewrite system adopts (rule R16, Appendix E). Without the run argument on mkOut, outputs would be identified by their knowledge content alone. Two evaluations producing the same knowledge from disjoint input batches would then yield the *same* output term, forcing $\mathrm{derivedFrom}(o) \subseteq I_1 \cap I_2 = \varnothing$ and contradicting traceability. Output identity is thus not an implementation convenience but a consistency requirement of the specification, and it mirrors PROV, where entities are individuated by IRIs rather than by content.

(A5) Segment-safe feeding preserves the segment and unions knowledge.
Let $s = \mathrm{inSeg}(i)$. Then:

$$\begin{aligned} \mathrm{inSeg}\big(\mathrm{feed}(O, i)\big) &= \mathrm{inSeg}(i), \\ \mathrm{inKnow}\big(\mathrm{feed}(O, i)\big) &= \mathrm{inKnow}(i) \cup \mathrm{filterScope}\big(s,\ \mathrm{outK}(O)\big). \end{aligned}$$

Explanation: Feeding never changes which segment an input refers to. It only adds produced knowledge items whose scope includes that segment.

(A6) Segment-safe flat feeding adds only atoms (groups expanded).
Let $s = \mathrm{inSeg}(i)$. Then:

$$\mathrm{inSeg}\big(\mathrm{feedFlat}(O, i)\big) \;=\; \mathrm{inSeg}(i),$$

$$\mathrm{inKnow}\big(\mathrm{feedFlat}(O, i)\big) \;=\; \mathrm{inKnow}(i) \;\cup\; \big\{\, a \in \mathrm{flatten}(\mathrm{outK}(O)) \mid s \in \mathrm{scopeOf}(a) \,\big\}.$$

Explanation: Flat feeding replaces groups by their atomic members, again only adding knowledge relevant to the segment.

Proposition (P1). Feeding is monotone, extensive, idempotent, commutative, and batch-compositional.
The following properties are *derivable* from axioms A5 and A6 together with the ACI properties (associative, commutative, and idempotent) of set union. They are stated explicitly for reference and to make the algebraic guarantees of the propagation pipeline immediately visible. For $\star \in \{\mathrm{feed}, \mathrm{feedFlat}\}$:

$$O_1 \subseteq O_2 \implies \mathrm{inKnow}\big(\star(O_1, i)\big) \subseteq \mathrm{inKnow}\big(\star(O_2, i)\big) \quad \text{(output monotonicity)},$$

$$\big(\mathrm{inSeg}(i_1) = \mathrm{inSeg}(i_2) \;\wedge\; \mathrm{inKnow}(i_1) \subseteq \mathrm{inKnow}(i_2)\big) \implies \mathrm{inKnow}\big(\star(O, i_1)\big) \subseteq \mathrm{inKnow}\big(\star(O, i_2)\big) \quad \text{(input monotonicity)},$$

$$\mathrm{inKnow}(i) \;\subseteq\; \mathrm{inKnow}\big(\star(O, i)\big) \quad \text{(extensivity)},$$

$$\star\,(\varnothing, i) \;=\; i \quad \text{(identity)},$$

$$\star\,(O_1 \cup O_2,\, i) \;=\; \star\big(O_1,\, \star(O_2, i)\big) \quad \text{(batch decomposition)},$$

$$\star\,\big(O_1,\, \star(O_2, i)\big) \;=\; \star\big(O_2,\, \star(O_1, i)\big) \quad \text{(commutativity)},$$

$$\star\,\big(O,\, \star(O, i)\big) \;=\; \star(O, i) \quad \text{(idempotence)}.$$

Derivation sketch. All properties follow from expanding $\star$ via A5 (respectively A6) and applying standard set-algebraic identities:

- *Identity:* $\mathrm{outK}(\varnothing) = \varnothing$, so $\mathrm{filterScope}(s, \varnothing) = \varnothing$, hence $K \cup \varnothing = K$.
- *Extensivity:* The result is $K \cup F$ where $F = \mathrm{filterScope}(s, \mathrm{outK}(O))$, since

$K \subseteq K \cup F$, extensivity holds.

- *Batch decomposition:* $\star(O_1 \cup O_2, i)$ yields $K \cup F_1 \cup F_2$ where $F_j = \text{filterScope}(s, \text{outK}(O_j))$, because outK and filterScope distribute over union. Nested application $\star(O_1, \star(O_2, i))$ gives $(K \cup F_2) \cup F_1 = K \cup F_1 \cup F_2$ by ACI of $\cup$.
- *Commutativity:* Immediate from commutativity of $\cup$ in $K \cup F_1 \cup F_2 = K \cup F_2 \cup F_1$.
- *Idempotence:* $(K \cup F) \cup F = K \cup F$ by idempotence of $\cup$.
- *Output monotonicity:* $O_1 \subseteq O_2$ implies $\text{outK}(O_1) = \bigcup_{o \in O_1} \text{outKnow}(o) \subseteq \bigcup_{o \in O_2} \text{outKnow}(o) = \text{outK}(O_2)$ (by monotonicity of indexed union over a larger index set), which in turn implies $\text{filterScope}(s, \text{outK}(O_1)) \subseteq \text{filterScope}(s, \text{outK}(O_2))$ (since filtering a superset yields a superset), hence $K \cup F_1 \subseteq K \cup F_2$.
- *Input monotonicity:* $K_1 \subseteq K_2$ implies $K_1 \cup F \subseteq K_2 \cup F$. □

Explanation: Knowledge cannot be reduced. Order and batching do not matter, and re-feeding the same outputs has no further effect. These rules make pipelines consistent and predictable.

Examples

Example 1: Single input, single output; groups not used.

Purpose: Show basic evaluate-and-feed for one segment and one atomic fact.

Definitions:

- Segment $s \in \text{Seg}$; initial atomic fact $k_0 \in \text{KAtom}$ with $s \in \text{scopeOf}(k_0)$
- Input $i_0 = \text{mkIn}(s, \{k_0\})$; activity $e_A \in \text{KGE}$; counter $n_0 \in \text{N}$

Run and feed:

$$\text{let } (e'_A, O_1, n_1) = \text{eval}(e_A, \{i_0\}, n_0), \qquad e'_A = \text{mkRun}(e_A, \{i_0\}, n_0),$$
$$O_1 = \big\{\text{mkOut}(e'_A, \{k_1\})\big\}, \quad k_1 \in \text{KAtom}, \quad s \in \text{scopeOf}(k_1),$$
$$i_1 = \text{feed}(O_1, i_0).$$

Result (by A5):

$$\text{inSeg}(i_1) = s, \qquad \text{inKnow}(i_1) = \{k_0, k_1\}.$$

Description: The run e'_A produces a new atomic fact k_1 about s. Feeding merges it into the input's knowledge via set union. Provenance is fully computable: $\text{derivedFrom}(\text{mkOut}(e'_A, \{k_1\})) = \text{used}(e'_A) = \{i_0\}$ by rules R14–R16.

Example 2: Pipeline (run two activities in sequence).

Purpose: Demonstrate composability and monotone accumulation across steps.

Definitions: Continue from Example 1 with i_1 and counter n_1; take another activity $e_B \in \text{KGE}$.

Run and feed:

$$\text{let } (e'_B, O_2, n_2) = \text{eval}(e_B, \{i_1\}, n_1),$$
$$O_2 = \big\{\text{mkOut}(e'_B, \{k_2\})\big\}, \quad s \in \text{scopeOf}(k_2),$$
$$i_2 = \text{feed}(O_2, i_1).$$

Result:

$$\text{inKnow}(i_2) = \{k_0, k_1, k_2\}, \qquad \text{inSeg}(i_2) = s.$$

Description: The second run e'_B adds k_2; batch decomposition and idempotence (P1) ensure result independence of batching/order.

Example 3: Multi-input comparison producing a shared relation.

Purpose: Show multi-input evaluation and segment-safe distribution of a relation spanning multiple segments.

Definitions:

- Two segments $s_A, s_B \in \text{Seg}$
- Inputs $i_A = \text{mkIn}(s_A, \{k_A\})$, $i_B = \text{mkIn}(s_B, \{k_B\})$ with $k_A, k_B \in \text{KAtom}$
- Comparison activity $e_C \in \text{KGE}$; counter n

Run and feed to both:

$$\begin{aligned}&\text{let } (e'_C, O, n') = \text{eval}(e_C, \{i_A, i_B\}, n),\\ &\quad O = \big\{\text{mkOut}(e'_C, \{k_{rel}\})\big\}, \quad \text{scopeOf}(k_{rel}) = \{s_A, s_B\},\\ &i'_A = \text{feed}(O, i_A), \qquad i'_B = \text{feed}(O, i_B).\end{aligned}$$

Result:

$$\text{inKnow}(i'_A) = \{k_A, k_{rel}\}, \qquad \text{inKnow}(i'_B) = \{k_B, k_{rel}\}.$$

Description: The run e'_C uses both inputs ($\text{used}(e'_C) = \{i_A, i_B\}$), produces a relation k_{rel} whose scope includes both segments, and feeding copies it back to each input where it is in scope.

Example 4: Multi-output activity.

Purpose: Show that a single run can produce multiple outputs merged in one feed.

Definitions: Segment s, input $i = \text{mkIn}(s, \varnothing)$, activity e_D, counter n; atoms k_1, k_2 with $s \in \text{scopeOf}(k_1) \cap \text{scopeOf}(k_2)$.

Run and feed:

$$\begin{aligned}&\text{let } (e'_D, O, n') = \text{eval}(e_D, \{i\}, n),\\ &\quad O = \big\{o_1 = \text{mkOut}(e'_D, \{k_1\}),\ o_2 = \text{mkOut}(e'_D, \{k_2\})\big\},\\ &i' = \text{feed}(\{o_1, o_2\}, i).\end{aligned}$$

Result:

$$\text{inKnow}(i') = \{k_1, k_2\}.$$

Description: Both outputs share the run e'_D (a single multi-output run) while remaining distinct entities. Outputs are simply unioned on feed. No ordering constraints due to batch decomposition and commutativity (P1).

Example 5: Producing a knowledge group and keeping it as a group.

Purpose: Demonstrate propagation of hierarchical knowledge without flattening.

Definitions:

- Atoms $k_1, k_2 \in \mathrm{KAtom}$, group $g_1 = \mathrm{mkGroup}(\{k_1, k_2\})$, with $s \in \mathrm{scopeOf}(k_1)$ (hence $s \in \mathrm{scopeOf}(g_1)$ by A3)
- $i_0 = \mathrm{mkIn}(s, \{k_0\})$; activity e_G that produces g_1; counter n

Run and feed:

$$\begin{aligned} &\text{let } (e'_G, O, n') = \mathrm{eval}(e_G, \{i_0\}, n), \\ &\quad O = \big\{\mathrm{mkOut}(e'_G, \{g_1\})\big\}, \\ &i_1 = \mathrm{feed}(O, i_0). \end{aligned}$$

Result:

$$\mathrm{inKnow}(i_1) = \{k_0, g_1\}, \qquad \mathrm{members}(g_1) = \{k_1, k_2\}.$$

Description: The group is treated as a single knowledge item with internal structure where feeding preserves that structure.

Example 6: Nested groups, flat propagation.

Purpose: Show how to extract pure facts from nested groups when hierarchy is not needed.

Definitions: Extend Example 5 with atom $k_3 \in \mathrm{KAtom}$, nested group $g_2 = \mathrm{mkGroup}(\{g_1, k_3\})$, and an activity $e_H \in \mathrm{KGE}$ that produces g_2; assume $s \in \mathrm{scopeOf}(k_j)$ for $j = 1, 2, 3$.

Run and feedFlat:

$$\begin{aligned}&\text{let } (e'_H, O', n') = \text{eval}(e_H, \{i_0\}, n),\\ &\quad O' = \big\{\text{mkOut}(e'_H, \{g_2\})\big\},\\ &i_2 = \text{feedFlat}(O', i_0).\end{aligned}$$

Computation:

$$\text{atomsOf}(g_2) \;=\; \text{atomsOf}(g_1) \cup \{k_3\} \;=\; \{k_1, k_2, k_3\}.$$

Result:

$$\text{inKnow}(i_2) = \{k_0, k_1, k_2, k_3\}.$$

Description: Flat feeding expands groups to their atomic members and merges them into the input (subject to scope).

Example 7: Combining multi-output with groups.

Purpose: Illustrate mixed outputs (atoms and groups) and the effect of the two feeding modes.

Definitions: Activity e_M outputs both an atom and a group; let $i = \text{mkIn}(s, \varnothing)$ and assume $s \in \text{scopeOf}(k_4)$ and $s \in \text{scopeOf}(k_5) \cap \text{scopeOf}(k_6)$:

$$\begin{aligned}&\text{let } (e'_M, O, n') = \text{eval}(e_M, \{i\}, n),\\ &\quad O = \big\{\text{mkOut}(e'_M, \{k_4\}),\ \text{mkOut}(e'_M, \{g_3\})\big\},\\ &\quad g_3 = \text{mkGroup}(\{k_5, k_6\}).\end{aligned}$$

Feed (keep groups):

$$i' = \text{feed}(O, i) \quad\Longrightarrow\quad \text{inKnow}(i') = \{k_4, g_3\}.$$

Feed (flat):

$$i'' = \text{feedFlat}(O, i) \quad\Longrightarrow\quad \text{inKnow}(i'') = \{k_4, k_5, k_6\}.$$

Description: With feed, the group remains a single container item (its scope contains s by A3). With feedFlat, its contents are exposed as individual atoms and merged.

Example 8: Scope filtering (selective knowledge distribution).

Purpose: Demonstrate that feeding only adds knowledge whose scope includes the target segment.

Setup:

- Segments $s_A, s_B \in \mathrm{Seg}$
- Inputs $i_A = \mathrm{mkIn}(s_A, \varnothing)$, $i_B = \mathrm{mkIn}(s_B, \varnothing)$
- Activity $e_F \in \mathrm{KGE}$ producing knowledge with different scopes; counter n

Run and feed:

$$\begin{aligned}&\text{let } (e'_F, O, n') = \mathrm{eval}(e_F, \{i_A, i_B\}, n),\\ &\quad O = \big\{\mathrm{mkOut}(e'_F, \{k_A, k_B, k_{AB}\})\big\},\\ &\quad \mathrm{scopeOf}(k_A) = \{s_A\}, \quad \mathrm{scopeOf}(k_B) = \{s_B\}, \quad \mathrm{scopeOf}(k_{AB}) = \{s_A, s_B\},\\ &i'_A = \mathrm{feed}(O, i_A), \qquad i'_B = \mathrm{feed}(O, i_B).\end{aligned}$$

Result:

$$\mathrm{inKnow}(i'_A) = \{k_A, k_{AB}\}, \qquad \mathrm{inKnow}(i'_B) = \{k_B, k_{AB}\}.$$

Description: Although the output contains $\{k_A, k_B, k_{AB}\}$, feeding applies filterScope: segment s_A receives k_A and k_{AB} but not k_B (since $s_A \notin \mathrm{scopeOf}(k_B)$), and symmetrically for s_B.

Formal Proof Sketch and Mechanical Validation

The axioms defined so far (A1–A6) together with the derived P1 specify the *declarative* behavior of the knowledge propagation algebra. They describe conceptually what a valid knowledge pipeline looks like. However, to execute this

algebra on a computer, as for instance, using an automated reasoning engine or a semantic rule processor, we must translate these static rules into an *operational* framework like a term rewrite system. Essentially, a term rewrite system is a computational tool that continuously simplifies expressions using strict replacement rules until a final, stable state is reached [BN98].

In addition to a formal validation sketch provided as supplementary material in Appendix E, complementing the formal proofs of Theorems E.2 and E.3, we mechanically validated the rewrite system $\mathcal{R}$ using the Maude engine [Cla+20].

Setup. The executable specification encodes the full signature Σ with all rewrite rules R0–R16 as unconditional Maude equations. Set union is declared with `assoc` and `comm` attributes together with `id:` $\varnothing$ while idempotence (R0b) is added as an explicit equation. Together these implement the canonical finite-set semantics of Appendix E: rewriting modulo associativity and commutativity, with empty summands and duplicates actively normalised away. Axiom A4 is realised via explicit state threading.

Runs and outputs therefore carry their full construction data, so that the injectivity of the constructors discharges freshness, and the provenance accessors used, runOf, and derivedFrom become computable by rewriting (rules R14–R16). A test module defines concrete ground terms: two segments s_A, s_B; four atoms k_1, k_2, k_3, k_4 with distinct scope assignments (single-segment, shared, and complementary) and a group $g_1 = \text{mkGroup}(\{k_1, k_2\})$. This configuration exercises all scope-filtering outcomes, both hierarchical and flat propagation, multi-input evaluation with provenance, and output identity under repeated evaluation (identical knowledge content, distinct run instances). The environment is containerised via Docker [Doc26] to ensure reproducibility.

Coverage and Results.

All reductions terminated and produced the expected normal forms. Every defined operator of the computational fragment was fully eliminated during reduction, leaving only constructor terms, which is consistent with Theorem E.2. Both reduction paths of each critical pair converged to the same normal form modulo

AC, consistent with Theorem E.3. The algebraic properties of P1 hold as derived consequences of the rewrite rules, the AC theory, and the set-normalisation rules on union, without requiring additional equations on feed or feedFlat themselves. The state-threaded evaluation confirms that identical activity-input pairs produce identical output *knowledge* while yielding provenance-distinct run instances and output entities.

Beyond validating the algebra itself, the Maude specification can also serve as a reference engine for concrete SemTS instances. Translating a pipeline into algebraic ground terms and comparing reduced normal forms against actual knowledge states can reveal scope violations, provenance gaps, or monotonicity breaches. For structural constraints, the axioms can additionally be expressed as SHACL shapes to enable direct validation on the RDF level.

5.4.2 Contextual Knowledge Reasoning

One of SemTS's features is to enable the propagation and refinement of knowledge through scenario-specific reasoning processes, which are essential for validating approximated knowledge or generating new insights from existing data. One related, important aspect is the definition of the quality of Data Knowledge through Scenario Knowledge (cf. Section 3.1). In the following, we first review related approaches, then describe applications of contextual inference, present a declarative solution using SPARQL CONSTRUCT queries, and discuss alternative machine learning-based approaches.

Related Work

Continuous query languages such as C-SPARQL [Bar+10] and RSP-QL [Del+14] enable real-time reasoning over RDF streams by joining streaming data with static background knowledge. EP-SPARQL [Ani+11] extends this with complex event processing capabilities, while Della Valle et al. [Del+09] provide a broader perspective on reasoning over rapidly changing information. These approaches are optimized for real-time queries rather than persistent time series segments,

and query results are not treated as knowledge entities that can be iteratively refined or propagated.

Rule-based reasoning offers declarative mechanisms for deriving new knowledge. SWRL [Hor+04] combines OWL ontologies with Horn-like rules for deductive inference. However, such rules operate at the ontology level and do not natively support temporal reasoning over indexed segments. As an alternative to explicit rules, machine learning approaches can learn inference patterns automatically. Knowledge graph embedding methods (see Section 5.3.2) learn continuous vector representations that capture latent associations between entities, enabling link prediction without manual rule specification. Nickel et al. [Nic+16] review relational machine learning techniques for such scenarios. Knowledge graph refinement [Pau17] addresses completion and correction through statistical methods, while neuro-symbolic approaches [GL23] combine neural learning with symbolic reasoning to discover patterns while maintaining interpretability.

At the schema-level, the SSN/SOSA ontology [JHC+19] provides a framework for describing sensors and observations with features of interest and provenance via PROV-O alignment. Domain extensions such as SAREF [DHR15] build on these foundations. However, as mentioned in Section 4.2, SSN/SOSA models individual observations rather than time series segments and does not distinguish between data and validated domain knowledge. Contextual inference requires manual query design without standardized patterns for knowledge categorization. SemTS addresses this gap by treating time series segments with knowledge being explicitly typed into data-derived and scenario-specific categories using extensible SKOS vocabularies. In addition contextual inference produces new knowledge entities with PROV-based provenance that can be propagated back to relevant segments for iterative refinement.

Applications of Contextual Inference

The combination of Data Knowledge and Scenario Knowledge enables contextual reasoning across diverse analytical scenarios. The subsequent application examples illustrate how derived insights can be interpreted through domain context:

- **Anomaly Explanation during Events:** Anomalies detected in sensor data may coincide with scheduled maintenance periods or similar events. While these anomalies might initially be flagged as significant, the context provided by Scenario Knowledge (e.g., a `MaintenanceEvent`) can suggest that anomalies are rather expected behaviors. Reasoning can generate an explicit assumption that the anomaly is explained by maintenance, or downgrade the confidence in the detected anomaly.

- **Operational Mode Correlation:** Detected changepoints or regime shifts (`semts-dv:Changepoint`, `semts-dv:RegimeChange`) can be correlated with known operational mode transitions (`semts-sv:OperatingMode`). A changepoint coinciding with a scheduled shift from automatic to manual control mode may represent expected behavior rather than an anomaly, and reasoning can generate an explanatory assumption linking the two.

- **Environmental and External Factors:** In domains such as energy or agriculture, time series patterns may correlate with external events recorded as Scenario Knowledge like weather conditions, calendar entries (`semts-sv:CalendarEntry`), or business events (`semts-sv:BusinessEvent`). For example, an unusual consumption pattern detected in energy data might be explained by a public holiday, and reasoning can attach this contextual explanation to the affected segments.

- **Cross-Segment Pattern Validation:** When reoccurring patterns (`semts-dv:Pattern`) are detected across multiple segments, Scenario Knowledge can provide explanations for why certain segments exhibit similar behavior. If segments sharing a detected motif also share a common operational context, reasoning can generate hypotheses about causal factors that can later be validated by domain experts.

- **Alert Prioritization and Suppression:** In operational settings, automated alerts (`semts-sv:Alert`, `semts-sv:ThresholdBreach`) may flood operators during known exceptional conditions. Contextual inference can automatically associate alerts with overlapping planned downtime (`semts-sv:PlannedDowntime`) or known error states (`semts-sv:Error`), generating information that certain

alerts are expected.

With respect to SemTS, the reasoning workflow typically follows the same structure: (i) identify temporal or semantic relationships between Data Knowledge and Scenario Knowledge, (ii) generate new knowledge entities with full provenance, (iii) optionally refine these entities through manual verification or subsequent automated reasoning, and (iv) propagate the refined knowledge back to the relevant segments. This systematic approach ensures that contextual interpretations are integrated continuously, leading to an evolving, semantically enriched knowledge graph.

Contextual Reasoning using SPARQL CONSTRUCT

Contextual reasoning in SemTS is based on the synergy between Data Knowledge and Scenario Knowledge. When reasoning is applied, SemTS combines these two types of knowledge to produce new insights, often in the form of semantically refined or validated knowledge.

SPARQL CONSTRUCT [KRU15] queries provide a flexible and declarative mechanism for implementing reasoning in SemTS. These queries enable the generation of new triples in the knowledge graph based on logical patterns and relationships. The SPARQL CONSTRUCT query in Listing 5.6 demonstrates how reasoning can be applied to identify and explain anomalies that overlap with maintenance sessions according to the first previously listed application example.

As illustrated in Figure 5.12, this query identifies time series segments `?dataSeg` containing anomalies that temporally overlap with segments `?scSeg` carrying Scenario Knowledge of type `MaintenanceEvent`. For each overlapping pair, the CONSTRUCT clause creates a `semts:KnowledgeReasoning` activity with two input nodes and one output node. The output produces a new `semts:ScenarioKnowledge` instance with the concept `Assumption`, representing the hypothesis that the anomaly is explained by maintenance. Provenance is captured through `semts:used`, `semts:generated`, and `semts:wasDerivedFrom` relations. The hypothesis is attached to both segments as `semts:multiSegmentKnowledge`. Generated assumptions represent

Listing 5.6: SPARQL CONSTRUCT query for reasoning in SemTS.

```
PREFIX semts: <https://w3id.org/semts/ontology#>
PREFIX semts-dv: <https://w3id.org/semts/vocabulary/data-knowledge#>
PREFIX semts-sv: <https://w3id.org/semts/vocabulary/scenario-knowledge#>

CONSTRUCT {
    _:gen a semts:KnowledgeReasoning ;
    semts:used _:inData, _:inScenario ;
    semts:generated _:out .

    _:inData a semts:KnowledgeGenerationInput ;
    semts:inputDataReference ?dataSeg ;
    semts:inputKnowledge ?anomaly .

    _:inScenario a semts:KnowledgeGenerationInput ;
    semts:inputDataReference ?scSeg ;
    semts:inputKnowledge ?statusK .

    _:out a semts:KnowledgeGenerationOutput ;
    semts:outputKnowledge _:hyp ;
    semts:wasDerivedFrom _:inData, _:inScenario .

    _:hyp a semts:ScenarioKnowledge ;
    semts:knowledgeConcept semts-sv:Assumption ;
    semts:knowledgeGenerationEntity _:gen ;
    semts:title "Context assumption: anomaly explained by maintenance" ;
    semts:hasValue [ a semts:EmbeddedValue ; semts:valueString "False Positive
        Assumption" ] .

    ?dataSeg semts:multiSegmentKnowledge _:hyp .
    ?scSeg semts:multiSegmentKnowledge _:hyp .
}
WHERE {
    ?dataSeg a semts:TimeSeriesSegment ;
    semts:segmentKnowledge ?anomaly .
    ?anomaly a semts:SegmentPrediction ;
    semts:knowledgeConcept semts-dv:Anomaly .

    ?scSeg a semts:TimeSeriesSegment ;
    semts:segmentKnowledge ?statusK .
    ?statusK a semts:ScenarioKnowledge ;
    semts:knowledgeConcept semts-sv:MaintenanceEvent .

    FILTER(?dataSeg != ?scSeg
    && NOT EXISTS { ?dataSeg semts:before ?scSeg }
    && NOT EXISTS { ?dataSeg semts:after ?scSeg })
}
```

tentative knowledge requiring validation. A domain expert can inspect the assumption, examine the underlying data, and confirm or reject the explanation. Upon confirmation, a subsequent reasoning step can for instance transform the assumption into a validated fact by creating a new `semts:ScenarioKnowledge` instance with the concept `semts-sv:Fact`, linking it to the original assumption

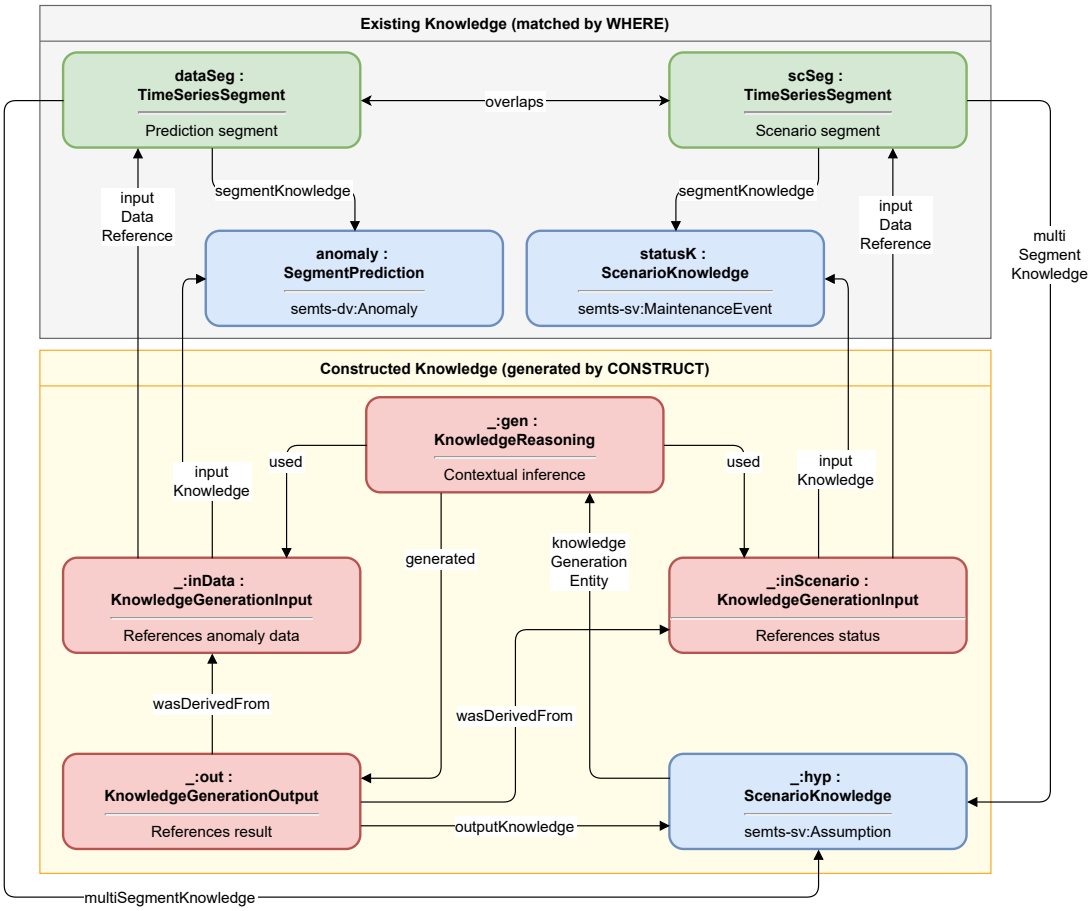


Figure 5.12: Contextual reasoning in SemTS via SPARQL CONSTRUCT

via provenance. The associated `semts:KnowledgeQualityMeasure` can be updated to reflect increased confidence.

Alternative Approaches for Contextual Reasoning

While SPARQL CONSTRUCT provides a declarative and interpretable mechanism for contextual reasoning, machine learning-based approaches offer alternatives that can learn inference patterns automatically without explicit rule specification. This subsection discusses how such approaches could be integrated with SemTS.

Knowledge Graph Embeddings. Knowledge graph embedding methods (see Section 5.3.2) learn continuous vector representations of entities and relations that capture latent patterns in the graph structure. In the context of SemTS, an embedding model is trained on the knowledge graph containing time series segments, their associated Data Knowledge, and Scenario Knowledge. For the maintenance example, rather than specifying explicit rules, the embedding model learns from historical data that anomaly entities co-occurring temporally with maintenance entities tend to have a particular relational pattern. Given a new anomaly, the model predicts a link to a contextual explanation based on learned associations. Methodically, this involves using link prediction to suggest contextual relationships for new Data Knowledge entities, and generating high-confidence predictions as new `semts:ScenarioKnowledge` instances with appropriate provenance indicating the inference method. The advantage is automatic pattern discovery across potentially complex relationships that may be difficult to specify manually. The disadvantage is reduced interpretability by learning statistical associations rather than explicit logical rules.

Knowledge Graph Refinement. Knowledge graph refinement approaches [Pau17] use statistical and machine learning methods to complete missing links and correct errors. Applied to SemTS, refinement methods analyze the existing knowledge graph to identify segments where contextual explanations are likely missing. In the maintenance example, a refinement system observes that anomaly segments temporally overlapping with maintenance segments often have associated `Assumption` knowledge, while some do not. The system then suggests adding the missing contextual links. This approach is particularly useful for ensuring consistency across large knowledge graphs where manual rule execution may miss cases due to data quality issues or incomplete temporal annotations.

Neuro-Symbolic Reasoning. Neuro-symbolic approaches [GL23] combine neural network learning with symbolic reasoning. In the SemTS context, a neuro-symbolic system learns to map combinations of Data Knowledge types and Scenario Knowledge types to inferred conclusions while respecting ontological constraints. For the maintenance example, such a system is trained on examples

Table 5.10: Comparison of contextual reasoning approaches for SemTS.

Criterion	**SPARQL**	**Embeddings**	**Graph Refinement**	**Neuro-Symbolic**
Interpretability	High	Low	Medium	Medium
Manual effort	High	Low	Low	Medium
Training data required	None	Yes	Yes	Yes
Ontology compliance	Guaranteed	Not guaranteed	Not guaranteed	Can be enforced
Pattern discovery	No	Yes	Yes	Yes
Provenance support	Native	Requires mapping	Requires mapping	Can be integrated

where experts have confirmed or rejected contextual explanations. The neural component learns patterns from feature representations of the involved segments and knowledge entities, while the symbolic component ensures that generated knowledge conforms to the SemTS ontology structure. The solution could be realized using a neural encoder that processes segment features (temporal extent, associated knowledge types, data characteristics) and produces latent representations. A reasoning module then combines representations from Data Knowledge and Scenario Knowledge to predict contextual relationships, and a symbolic decoder generates valid SemTS knowledge structures from these predictions.

Comparison. SPARQL CONSTRUCT offers full interpretability and guaranteed ontology compliance with native provenance support, but requires manual rule specification. Embedding and refinement methods reduce manual effort and can discover patterns automatically, but require training data and additional effort to ensure generated knowledge conforms to the SemTS structure. Neuro-symbolic approaches offer a middle ground, potentially combining pattern discovery with ontological constraints, but involve greater architectural complexity.

For applications requiring transparency, such as industrial quality control or regulatory compliance, SPARQL-based reasoning may be preferred. For exploratory scenarios with large knowledge graphs where manual rule engineering is impractical, learning-based methods provide a viable alternative. Hybrid architectures are also possible: machine learning methods could suggest candidate inferences

that are then validated through SPARQL queries checking ontological consistency before materialization in the knowledge graph.

Table 5.10 summarizes the trade-offs between these approaches.

Chapter Summary

This chapter addressed the practical realisation of the SemTS knowledge graph, contributing to both RQ 2 and RQ 3. We demonstrated the applicability of SemTS through three real-world scenarios - computer vision for animal welfare, grinding bead wear forecasting, and anomaly detection in blow moulding - each highlighting different ontology capabilities including multimodal knowledge groups, informed knowledge propagation, and multi-stage provenance chains. For knowledge graph generation, we introduced two complementary approaches: the SemTS Population Library (SPL) for programmatic, type-safe ontology population, and *Code2Onto*, a novel multi-agent framework that automates ontology population from program code and its runtime context. For knowledge retrieval, we explored both SPARQL-based semantic indexing for precise, schema-aware queries and embedding-based retrieval. Finally, we formalized knowledge propagation through a many-sorted algebraic specification that guarantees traceable provenance and monotone knowledge growth and demonstrated contextual reasoning through SPARQL `CONSTRUCT` queries that combine Data Knowledge with Scenario Knowledge to derive validated insights.

What follows. While this chapter established the mechanisms for populating, retrieving, and propagating knowledge, Chapter 6 focuses on exploiting the resulting knowledge graph within concrete analytical workflows, i.e. from interactive exploration and grammar-based method selection to fully automated, LLM-driven data science agents.

6 SemTS-based Data Analytics

The previous chapters mainly established an understanding of the SemTS ontology, associated techniques for populating the knowledge graph and approaches for knowledge retrieval and contextual reasoning. We demonstrated how the semantic organization of time series within the SemTS knowledge graph directly supports analysis workflows. Particularly, how users can query relationships between segments, filter by knowledge types, and trace how insights were derived, thereby enabling systematic exploratory analysis. This chapter addresses RQ 3 by exploring how these contributions can be exploited, presenting several approaches that leverage structured knowledge representation and associated workflows for retrieval and preservation. We discuss the visual exploration of time series segments and related knowledge, match analysis methods to graph-encoded data characteristics, and a utilization of the knowledge graph to support agentic, LLM-based data science workflows.

Chapter Outline

Chapter 6 starts by introducing SemTS Explorer in Section 6.1, a web-based application that utilizes SemTS to visually represent knowledge-enriched time series segments for knowledge exploration and contextual reasoning. In Section 6.2 we present a grammar- and SPARQL-based solution to the automated selection of analytical approaches in the context of automated knowledge discovery based on our work [GBD24]. Finally, in Section 6.3 we discuss the exploitation of a SemTS graph serving as information repository and incrementally enriched graph memory for LLM-driven data science agents referencing our previous work [Gra+25c].

6.1 SemTS Explorer

SemTS Explorer (STE) is an interactive application for exploring and validating instances of SemTS. It integrates time series data with semantic annotations, allowing users to inspect captured knowledge visually. Beyond exploration, STE enables the tracking of knowledge propagation, SPARQL-based inspection, and LLM-assisted reporting.

STE was primarily developed to make the semantic modeling of SemTS visually explainable. It aims to provide an integrated environment in which users can explore time series segments together with the knowledge attached to them, inspect how that knowledge was produced and propagated, and author new Scenario Knowledge describing annotations or reasoning steps. By processing knowledge graphs populated from the SemTS ontology and vocabularies, it becomes possible to move from simple data-driven plots towards a visual, provenance-aware presentation and reporting of time series knowledge.

6.1.1 Motivation and Related Work

In many practical settings, visual time series analysis [FXJ20] still corresponds to conventional dashboards that show plots of sensor readings, thresholds, and custom comments. These views help with basic monitoring and simple descriptive diagnosis, but do not account for a presentation of inferred insights from conducted analyses by default. While domain- [JDA+25] and task-specific approaches [Ali+19; Lek+20] exist to analyze time series visually, these approaches do not generalize to scenario-agnostic time series analytics or integrate associated contextual information. Knowledge such as the result of an anomaly detection model, a domain expert's confirmation or a change in operating mode can be semantically correlated or scattered across different sources. Moreover, the connection between determined time series knowledge and the processes that produced this knowledge is often implicit and not always traceable. What is missing in many practical setups is an interactive environment that combines and exposes the previous information to analysts and domain experts in a way that is useful for labeling, exploration and validation.

Several related tools exist which partially align with the idea of STE in different aspects. For monitoring, open-source dashboard solutions such as Grafana [CK21], Kibana [Ela11] or Chronograf [Inf15] provide comprehensive time series visualizations with the ability to add annotations. Users can mark individual points or intervals of time series data on a chart, attach descriptive text and correlate information across multiple panels. One problem is however, that time series insights or associated information are typically free-form metadata which are not linked via explicit provenance relations to the analyses or processes that produced them. Annotations are not classified as standardized, reusable semantic entities to explicitly define data-driven insights and scenario context.

Another category of related approaches concerns time-aligned annotation and labeling tools. Platforms like Label Studio [Tka+20] allow users to perform complex human-in-the-loop labeling. Examples include class labels, notes on maintenance events, or alerts defined for time series intervals. This is conceptually close to SemTS's use of controlled vocabularies (cf. Section 3.6) and the definition of individual data structures via the extension of DCAT. Yet, in contrast to the first category, these tools are not designed for visual analysis.

Yet another category of associated solutions comprises the ones based on Semantic Web technologies. Knowledge-graph dashboards like GraphDB Workbench [Ont24], NeoDash [Neo20], or Stardog Studio [Sta18] can display graphs and charts derived from queries, including those that reference temporal information. These approaches allow to utilize structured semantics to quantitative data analysis, but they rather operate at the level of individual observations or entire datasets rather than explicit time series segments. Additionally compared to STE, they do not represent self-contained, preconfigured solutions to human-in-the-loop interaction for captured time series knowledge. This also holds for provenance and lineage visualization tools. Frameworks such as Marquez UI [Mar19], Apache Atlas [Apa15], or YesWorkflow [McP+15] make it possible to represent and inspect how results or intermediate information are derived from inputs through workflows and data transformations. This idea equals the PROV-based specification within SemTS, using the classes `KnowledgeGenerationEntity`, `KnowledgeGen-`

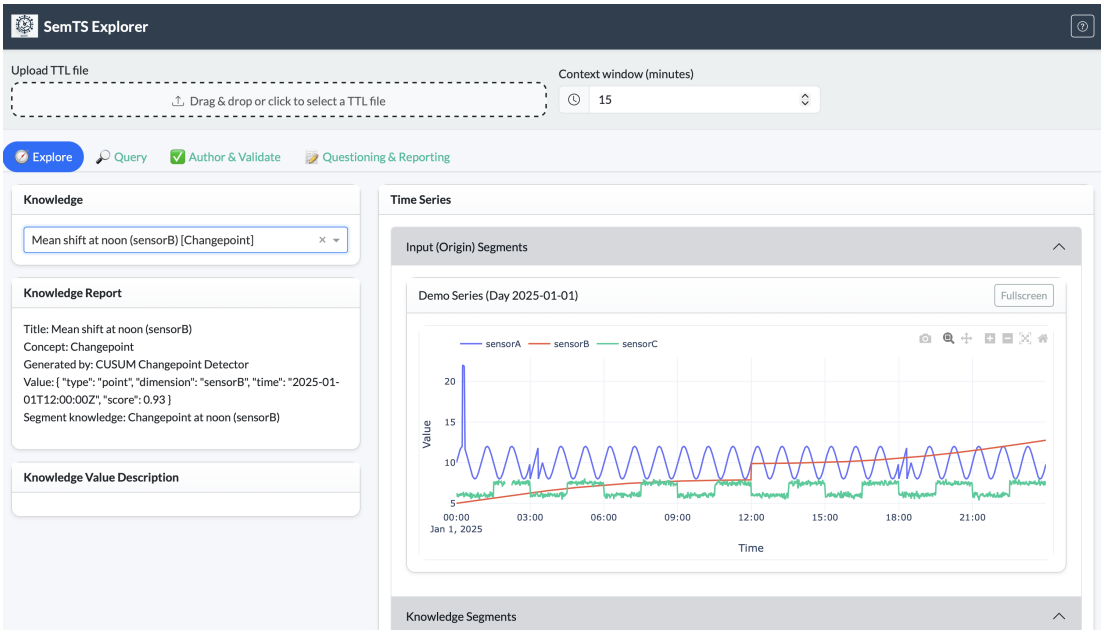


Figure 6.1: SemTS Explorer - *Explore* page

`erationInput`, and `KnowledgeGenerationOutput` combined with `used`, `generated`, and `wasDerivedFrom`. The former solutions provide graphical depictions of propagation chains and make information traceable. However, they are concretely designed to illustrate provenance-aware data pipelines and not integrated with interactive time series visualizations.

Given this comparison, identified open-source tools either provide time series monitoring capabilities with simple annotations, time-aligned labeling interfaces, or semantic modeling and provenance graphs for data pipelines. To the best of our knowledge, none of the existing approaches combine these features in a single solution for time series data. In particular, they do not offer a segment-centric, ontology-based visualization of time series along with a PROV-compliant description of classified knowledge instances. To integrate these elements, in the following section we discuss the SemTS Explorer as a complementary approach to the SemTS ontology and vocabularies.

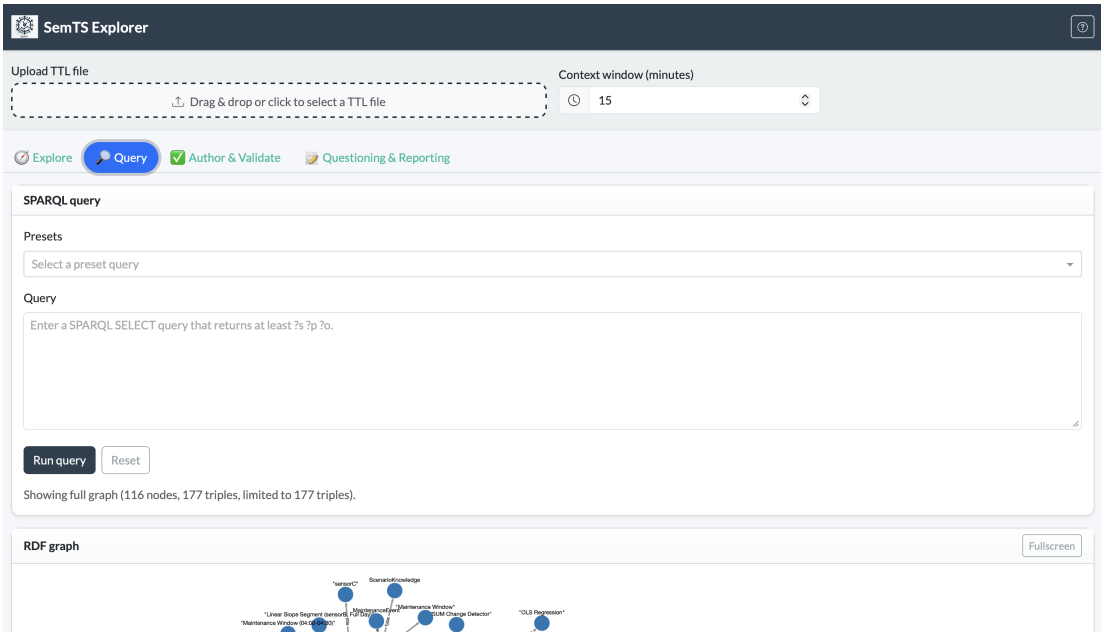


Figure 6.2: SemTS Explorer - *Query* page

6.1.2 Approach

SemTS Explorer is a Dash-based [Plo17] web application that operates directly on graphs conforming to the SemTS ontology and associated vocabularies. Conceptually, it assumes that in provided graphs, time series have been segmented into knowledge-enhanced `TimeSeriesSegment` instances defined by temporal index information and affiliated dimensions. Each segment additionally has at least one DCAT distribution pointing to the concrete time series data, for example a CSV file. Knowledge about these segments is modeled through `Knowledge` and its subclasses, with semantic details being provided via `KnowledgeConcept` instances from the vocabularies.

Knowledge Exploration and Querying

The user interaction begins on the *Explore* page illustrated in Figure 6.1, where users can upload SemTS graphs formatted in the RDF/TTL format. The application parses the graph and lists all knowledge instances for the user to select one. Once a knowledge instance is selected, SemTS Explorer determines all time series

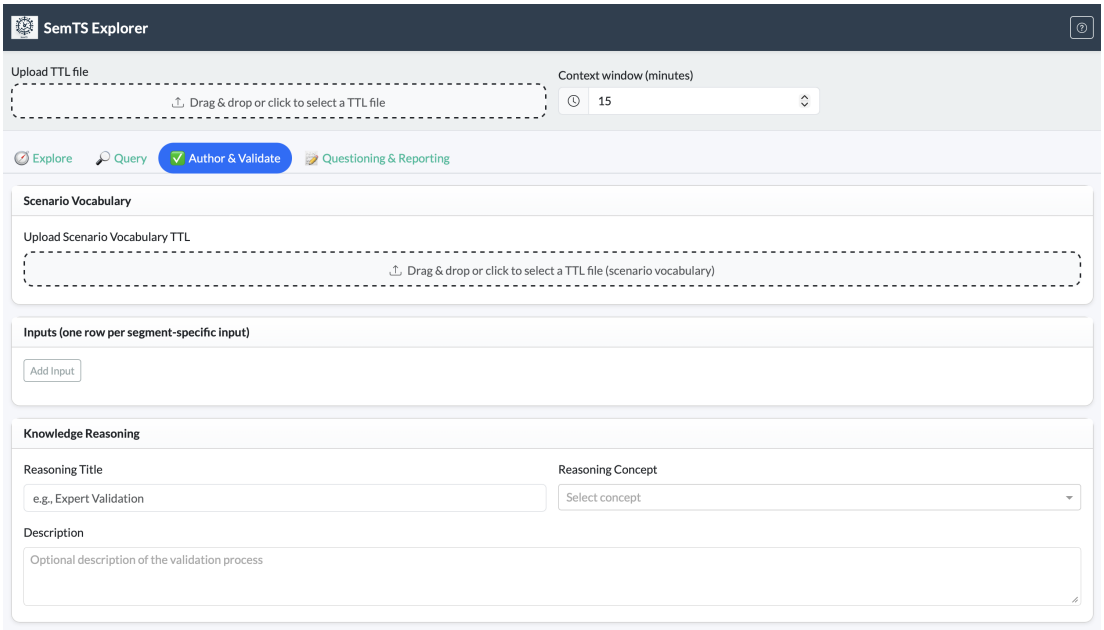


Figure 6.3: SemTS Explorer - *Author & Validate* page

segments that are involved in presenting this knowledge. It includes segments that directly reference the knowledge via `segmentKnowledge` or `multiSegmentKnowledge`, as well as segments that serve as input data to the `KnowledgeGenerationEntity` that produced the knowledge. For each of these segments, the corresponding DCAT distribution is resolved, the CSV is read into a pandas `DataFrame`, the timestamp column is interpreted as a time index, and the segment's time window is reconstructed from the OWL-Time index entity. The application distinguishes between origin segments, which correspond to the inputs of the knowledge-generation step and output segments representing knowledge such as anomalies or derived features. Origin segments are plotted on a global time axis derived from their combined ranges, whereas knowledge segments are scaled based on their own temporal windows, optionally extended by a user-supplied context offset to allow for a visualization of surrounding observations. The resulting time series plots show one trace per relevant data dimension. At the same time, SemTS Explorer inspects the selected knowledge's concept (or otherwise the RDF class if no concept was assigned) and uses a configurable mapping to the type of visualization. Depending on this mapping, the app may highlight

intervals, draw vertical lines for points in time, annotate scalar values, or overlay trends. For concepts that correspond to textual or purely categorical values such as scenario annotations, the information is rendered as text in an alternative panel, rather than being visualized. Technically, a modular architecture with registries for *knowledge type to visualization* enables a flexible addition of new graph types and concept-to-visualization mappings. In addition to the time series and knowledge plots, the Explore tab renders a knowledge propagation graph using Dash Cytoscape [Fra+16]. The graph makes the derivation chain behind the selected knowledge interpretable and complements the time series visualizations by provenance information about the knowledge generation and assignment processes.

In a second *Query* page shown in Figure 6.2, STE allows users to run SPARQL queries against the currently loaded graph data. Users can either select from a list of common, predefined queries or conduct custom queries. Corresponding results are visualized as an interactive network graph that shows subjects and objects as nodes and predicates as labeled edges.

Contextual Validation by Scenario Knowledge Ingestion

While the Explore page emphasizes inspection, the *Author & Validate* page, illustrated in Figure 6.3, supports the creation of new Scenario Knowledge and the explicit definition of expert-driven reasoning steps. Initially, the user can upload a vocabulary defining contextual concepts. As a default vocabulary we suggest the Scenario Knowledge vocabulary defined in Section 3.6, which provides several `KnowledgeConcept` individuals like `MaintenanceEvent`, `Annotation`, `Alert` or `CausalFactor`. SemTS Explorer parses these concepts and exposes them in dropdown menus across the page. At first, the user defines one or more inputs. Each input row consists of a selected time series segment and, optionally, one or more existing knowledge items. Afterwards, the user provides details about the reasoning step. As output, the user can either instantiate a single `ScenarioKnowledge` entity or a group of several entities utilizing `KnowledgeGroup`. When submitting, STE generates a delta graph with all required SemTS entities including reasoning-specific IO and provenance information. This delta is shown

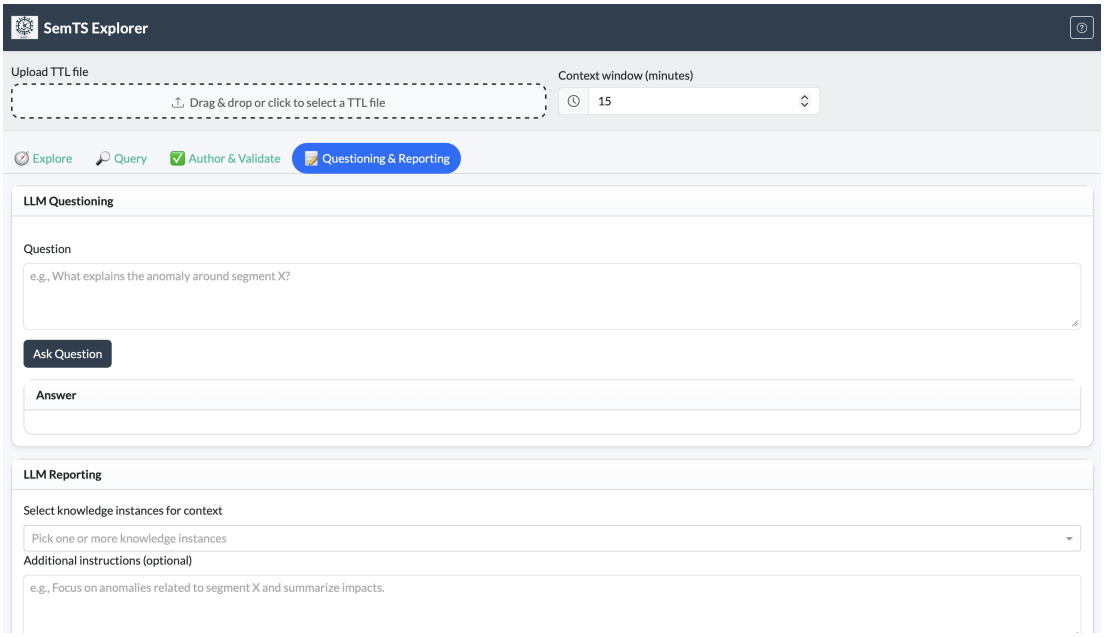


Figure 6.4: SemTS Explorer - *LLM Report* page

as Turtle for inspection and then merged into the base graph, so that the newly authored Scenario Knowledge immediately appears in the Explore tab for visualization alongside existing data-driven knowledge.

LLM-based Questioning and Reporting

As another page, STE provides a *LLM Questioning & Reporting* page to facilitate questioning and narrative reporting of the knowledge graph, depicted by Figure 6.4. For reporting, the user either selects one or more knowledge instances of interest, or ignores the selection to consider all knowledge available in the graph. Moreover, the user may optionally add instructions to guide the style and focus of the report. To draft prompts, SemTS Explorer uses the same functionality created to generate summaries in the Explore page, collapsing the titles, concepts, generating entities and further information into a single context. This context, together with the user's instructions, is provided as a prompt to a large language model endpoint, that can be configured in the configuration of STE. As a response, the model returns a coherent text report. Information available in the

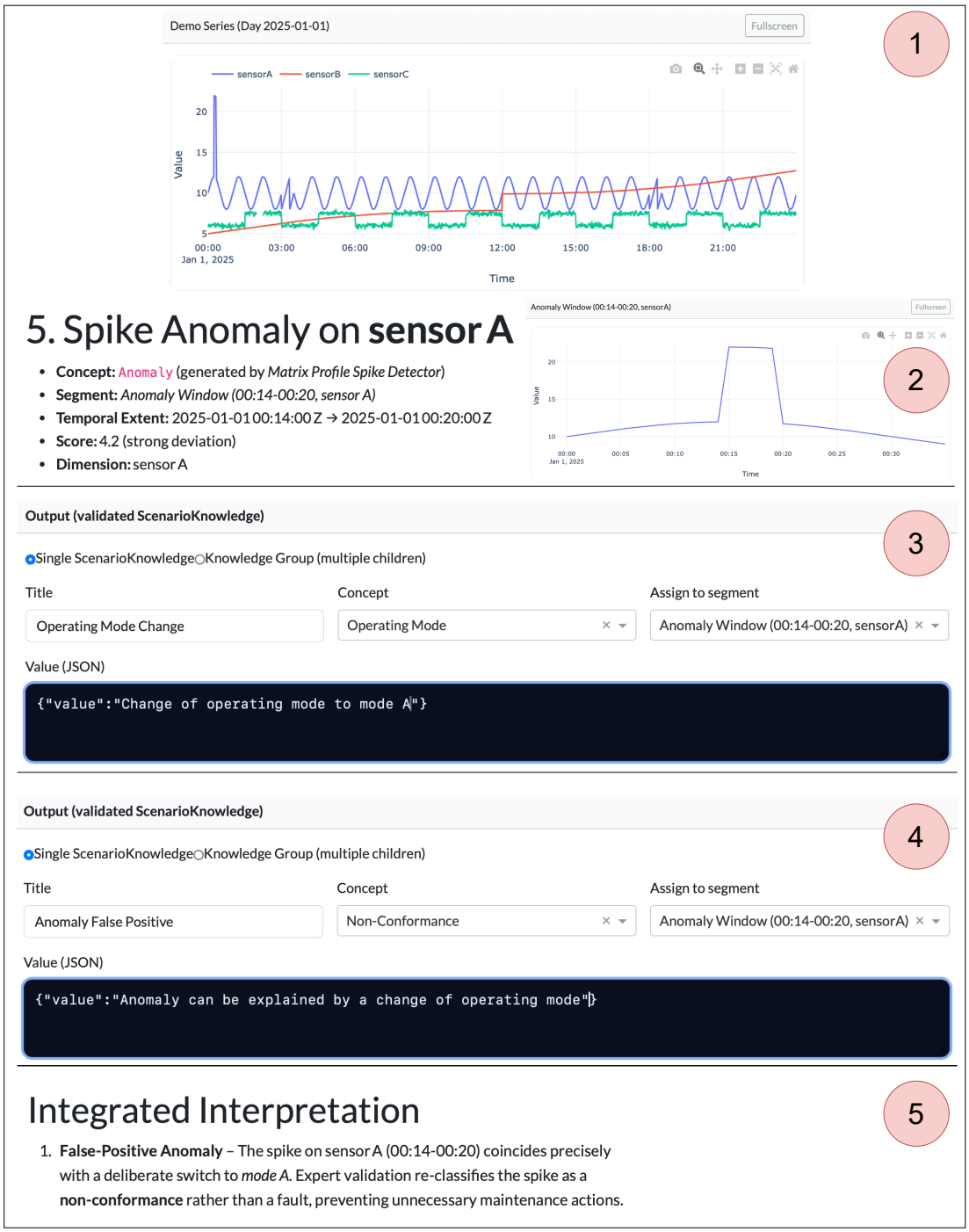


Figure 6.5: SemTS Explorer - Workflow demo: An identified anomaly is reclassified as *false positive* since the anomalous behavior can be explained by a change in operating mode. The final LLM report updates the intermediate report with full provenance including the anomaly interpretation.

report may include anomalies, forecasts, and scenario events associated with the individual segments. Although this does not perform formal reasoning, it exploits the effectiveness of LLMs to interprete structured information and to rephrase it in a human-readable form.

6.1.3 Application Example

A key objective of the SemTS Explorer is to integrate its various functions into coherent, multi-step processes that mirror typical data understanding workflows. One example of such a workflow involves the iterative visual inspection of analytical results, followed by validation and concluding with comprehensive reporting and interpretation. Figure 6.5 illustrates this workflow based on an example of how to use the SemTS Explorer and its provided functionalities. In the example, we consider a processing of the time series visualized in step 1. The generated report in step 2 indicates that the fifth entry of the identified knowledge corresponds to an anomaly in sensor A from 00:14 to 00:20 UTC on 1 January 2025. At a later point in time, indicated by step 3, a process expert adds the knowledge regarding a change in operating mode. A subsequent validation in step 4 identifies the anomaly to be a *false positive* as it correlates with the change of operating mode and can be explained by that. The final report, generated in step 5, respects these observations, adjusts the interpretation and explains the whole reasoning chain.

6.1.4 Summary and Future Work

In this section, we presented SemTS Explorer, an interactive web application designed to complement the SemTS ontology and vocabularies by providing visual exploration, validation, and reporting capabilities for time series knowledge. Due to a specialization to SemTS, the tool addresses the gap identified in existing solutions, extending the capabilities from series monitoring with simple annotations to a more detailed segment-specific exploration of time series knowledge. In particular, the SemTS Explorer can be used for: (i) visual exploration of knowledge-enhanced time series segments with provenance-aware propagation

graphs, (ii) SPARQL-based querying with interactive result visualization, (iii) authoring and validation of Scenario Knowledge through expert-driven reasoning steps, and (iv) LLM-assisted questioning and reporting.

Several directions for future work remain. First, the current implementation relies on pre-defined time series segments that must already exist in the uploaded knowledge graph. One potential extension would be to support custom segment definition directly within the application, allowing users to interactively select and define new segments from raw time series data before annotating them with knowledge. Second, SemTS Explorer currently operates on static, user-uploaded graphs. Integrating real-time data analysis capabilities in combination with Code2Onto would enable continuous monitoring scenarios where knowledge of new observations is dynamically incorporated and visualized. Third, enhancing the application with collaborative features for multi-user environments would support team-based validation workflows, enabling multiple domain experts to simultaneously annotate, review, and discuss knowledge instances.

6.2 Conditioned Knowledge Discovery

With the SemTS Explorer we presented a solution to visualize and manually propagate knowledge in a dedicated application. However, beyond manual propagation in some scenarios there is a need for automated knowledge discovery and enrichment (cf. Section 5.4.2). This section presents a conceptual framework for mapping analysis methods to time series data specifications, based on our prior work [GBD24]. The objective is to automatically identify data-compatible methods within the scope of knowledge discovery. To this end, we define a context-free grammar that captures hierarchies and compositions of time series data, discuss how semantic information influences method selection, and demonstrate the framework at a concrete example. We further describe an implementation grounded in SemTS and SPARQL-based condition evaluation before concluding with a summary and directions for future work.

6.2.1 Motivation and Related Work

Automated support for selecting and optimizing analytical methods has seen significant progress through intelligent systems designed for predefined tasks [KHS+21]. However, when the analysis task itself is not predetermined, the challenge of task-agnostic knowledge discovery remains difficult [Zha+23a]. In the context of SemTS, this challenge translates into the question of how a knowledge graph can be enriched beyond manually defined or task-driven analyses. The framework presented in this section supports automated knowledge propagation by systematically selecting compatible methods based on structural and semantic properties of the data defined by the knowledge graph. Since newly generated knowledge extends the graph state, subsequent iterations operate on an enriched basis, enabling the incremental derivation of new facts from existing information. This positions the approach as a complement to explicit, task-driven enrichment. Rather than requiring a predefined analytical objective, it can for instance operate as a background process that continuously evaluates which methods are applicable to the current graph state and may thereby reveal unexpected insights. A key factor contributing to the difficulty of automated knowledge discovery for time series data is the varying structural depth at which different analysis methods operate. Methods designed for univariate sequences are not necessarily applicable to collections of multivariate data, and conversely, collection-level approaches may be unsuitable at the level of individual subsequences. The hierarchical composition of time series data therefore restricts which analytical approaches are compatible.

Beyond structural composition, time series data is sometimes accompanied by additional information such as labels, statistical features, or domain-specific annotations [Chr+18]. The presence or absence of such knowledge further influences the selection of suitable methods, as informed approaches can yield improved accuracy or explainability when prior knowledge is available [Von+23]. Existing classification schemes for analysis algorithms address this partially but remain limited to predefined tasks [SS21; Yin+20]. In practice, it is often unclear which analytical conclusions a given data setting permits without first examining both its structural and semantic properties.

6.2.2 Approach

We categorize the complexity of time series data along two primary layers: a structural layer describing hierarchies and compositions, and a semantic layer capturing associated knowledge. While additional layers could be considered, such as algorithmic properties or technical constraints, these are either not data-specific or derivable from the structural and semantic information. For instance, the total amount of data or the number of labeled instances can be determined from the combination of both layers.

Grammar

The structural layer captures data complexity arising from combinations of time series at different granularities, independent of any semantic context. To formalize the possible compositions that occur in real-world scenarios, in Listing 6.1 we present a context-free grammar generating data combinations across hierarchical levels. The grammar defines sets of time series collections (STSC), individual time series collections (TSC), and single time series (TS) distinguishing multivariate (MTS) and univariate (UTS) variants. Subsequences along both the index and dimensionality axes are expressed through recursive production rules. Each time series terminates at its associated knowledge K, which in the context of SemTS corresponds to any knowledge linked to a time series segment via the `segmentKnowledge` or `multiSegmentKnowledge` property.

The production rules are described as follows:

1 An STSC produces a list of at least one time series collection.

2 The collection list is constructed recursively from individual TSC elements.

3 Each TSC produces a list of at least one time series.

4 The time series list follows the same recursive structure as the collection list.

5 A TS resolves to either its multivariate or univariate form.

6 An MTS can be split into subsequences (`mts_subseq`), projected to a lower

Listing 6.1: Context-free grammar to represent different hierarchical levels and compositions of time series data. Lowercase symbols allow for path tracking. The terminal symbol is K representing any associated knowledge.

```
STSC     → stsc(TSC_LIST)
TSC_LIST → TSC | TSC, TSC_LIST

TSC      → tsc(TS_LIST)
TS_LIST  → TS | TS, TS_LIST

TS       → MTS | UTS

MTS      → mts_subseq(TS_LIST) | mts_reduce(TS) | mts_leaf(K)

UTS      → uts_subseq(UTS_LIST) | uts_leaf(K)
UTS_LIST → UTS | UTS, UTS_LIST
```

dimensionality (`mts_reduce`), or terminated at its knowledge (`mts_leaf`).

7 A UTS can be split into univariate subsequences (`uts_subseq`) or terminated at its knowledge (`uts_leaf`).

8 The UTS list follows the same recursive construction as the other list symbols.

Figure 6.6 provides an additional illustration. Since the terminal symbol K represents semantic information that constitutes its own layer of data complexity, its role in method selection is discussed next.

Rule-Based Method Mapping

Temporal dependencies in time series data lead to characteristic variations including seasonality, trend and noise. Different analytical methods handle these variations with varying effectiveness, and the availability of prior knowledge about such characteristics opens additional methodical options [Von+23]. The semantic layer therefore complements the structural layer by determining which methods can leverage the knowledge associated with a given data setting.

In the context of SemTS, the terminal symbol K corresponds to knowledge associated with time series segments. As described in Section 4.4, SemTS distinguishes three primary knowledge categories: *DataKnowledge* encompasses insights de-

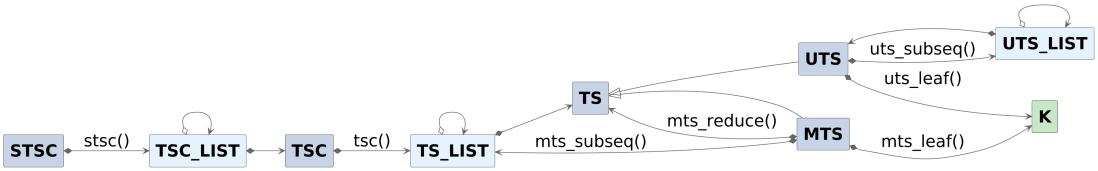


Figure 6.6: Object diagram representing the grammar from Listing 6.1 as visual description of time series hierarchies. Arrows represent individual rules, while compositions are defined by diamond symbols. Non-terminal symbols are colored in blue, non-terminal list symbols in lightblue and terminal symbols in green.

rived from the data itself, including statistical features, predictions, and segment relations. *ScenarioKnowledge* represents domain-specific information such as labels, annotations, and contextual descriptions. *MethodKnowledge* captures information about suitable analytical approaches.

Automated method selection requires that the semantic information of decomposed time series entities is matched against the preconditions of candidate algorithms.

Validation of algorithmic requirements

Let $T_{i,K}$ denote the set of all knowledge instances associated with the i-th time series segment. Within SemTS, this set comprises all instances reachable via the `segmentKnowledge` and `multiSegmentKnowledge` properties. Individual knowledge instances are further characterized by their `knowledgeConcept`, referencing concepts from the associated vocabularies.

Following [GBD24], the requirements of a given algorithm can be expressed as a logical formula V over a set of conditions C_k:

$$V = C_1 * C_2 * ... * C_m$$

where $*$ denotes a logical operator from $\{\wedge, \vee\}$ implicitly including $\neg$ for negated conditions.

As an illustrative case, a query-based similarity search using Euclidean distance over labeled time series requires the following conditions:

Listing 6.2: SPARQL condition verifying that all segments have associated labels.

```
PREFIX semts: <https://w3id.org/semts/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
ASK {
    {{SEGMENTS}}
    FILTER NOT EXISTS {
        ?seg a semts:TimeSeriesSegment .
        FILTER NOT EXISTS {
            ?seg semts:segmentKnowledge ?k .
            ?k semts:knowledgeConcept/skos:broader*
            <https://w3id.org/semts/vocabulary/scenario-knowledge#Label> .
        }
    }
}
```

- $C_1 := \forall i, j, \text{ if } i \neq j : \text{length}(T_i) = \text{length}(T_j)$
- $C_2 := \forall i, j, \text{ if } i \neq j : \text{dtype}(T_i) = \text{dtype}(T_j)$
- $C_3 := \forall i : \exists k \in T_{i,K} : \text{concept}(k) \sqsubseteq \texttt{Label}$

The conjunction $C_1 \wedge C_2 \wedge C_3$ enforces that all segments share the same length (C_1) and data type (C_2), and that each segment carries at least one knowledge instance whose concept is a `Label` or a subclass thereof (C_3).

Implementation via SPARQL Conditions

The validation of algorithmic requirements can be implemented by querying the SemTS knowledge graph. Each condition C_k translates to a query that evaluates whether the required semantic information is present and satisfies the specified constraints. SPARQL ASK queries provide a mechanism for this purpose, returning a boolean result indicating whether the condition holds.

Listing 6.2 shows an example condition that verifies whether all segments in a given set have associated label knowledge. The query checks that no segment exists without at least one knowledge instance whose concept is `Label` or a narrower concept in the SKOS hierarchy.

The placeholder `{{SEGMENTS}}` is replaced at runtime with a VALUES clause that binds the variable `?seg` to the specific segment URIs under evaluation. This parameterization allows the same condition definition to be reused across different

segment sets.

For conditions that require information not stored in the knowledge graph, such as the actual length of time series data stored in external files, a hybrid approach is employed. The system first attempts to evaluate the condition via SPARQL using metadata available in the graph (e.g., temporal duration from the segment index). If this information is unavailable, a fallback function retrieves and inspects the referenced data directly.

The concrete implementation follows a catalog-based architecture consisting of two registries: a *condition catalog* and a *method catalog*. The condition catalog maintains reusable condition definitions of three types: SPARQL-based conditions that query the knowledge graph directly, programmatic conditions that access external data sources, and hybrid conditions that combine both approaches with automatic fallback. Each condition implements an evaluation interface that accepts a graph reference, a set of segment URIs, and an optional context for runtime parameters. Conditions can be composed using logical operators ($\wedge$, $\vee$, $\neg$) to form complex requirement expressions. The method catalog registers analysis methods, each specifying its applicable hierarchy levels from the grammar and a composed condition representing its requirements. A discovery engine traverses the segment hierarchy, evaluates method requirements against the current graph state, and returns applicable methods at each level. This separation of conditions from methods enables reuse, as common requirements such as uniform data type or presence of labels can be defined once and referenced across multiple method definitions thereby reducing redundancy and ensuring consistency.

6.2.3 Application Example

To demonstrate the applicability of the conceptual framework, we consider a scenario involving two univariate time series collections as illustrated in Figure 6.7. Collection 1 contains sensor measurements from a manufacturing process where a recurring pattern has been identified. Collection 2 contains measurements from a similar process where we want to determine if the same pattern occurs.

The segment hierarchy is structured as follows: Collection 1 contains two time

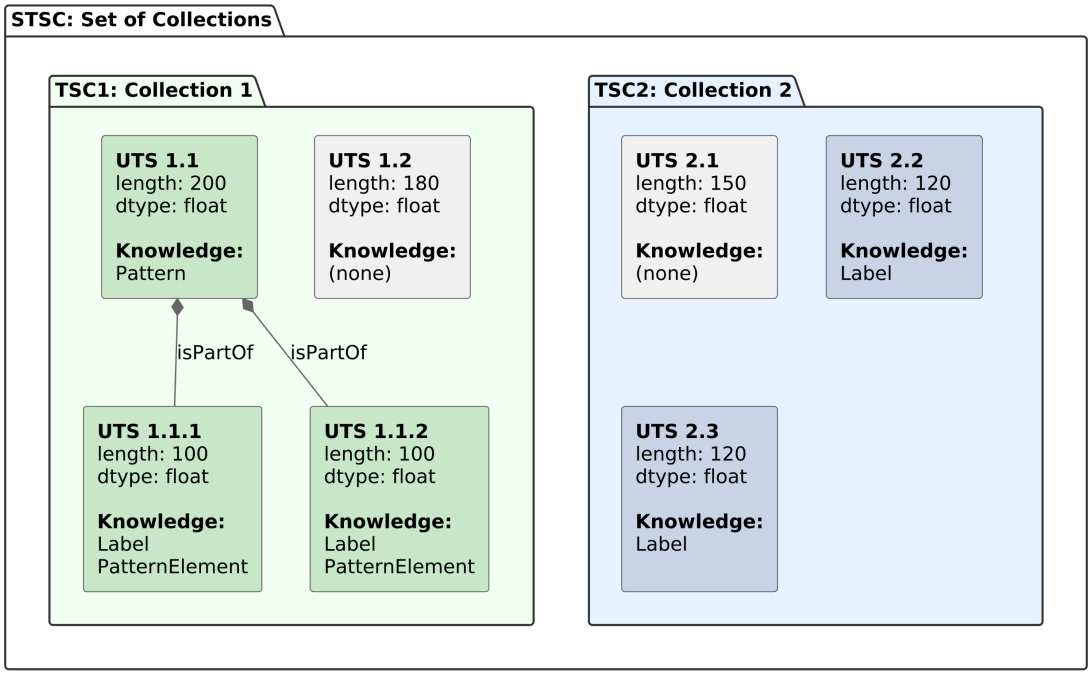


Figure 6.7: Simplified visualization of a concrete data setting.

series segments, where segment 1.1 has been further decomposed into two subsequences (1.1.1 and 1.1.2) representing instances of the identified pattern. Both pattern subsequences have associated `Label` and `PatternElement` knowledge and a length of 100 data points. Segment 1.2 has no associated knowledge. Collection 2 contains three time series segments with varying lengths (150, 120, and 120 data points respectively). Segments 2.2 and 2.3 have associated `Label` knowledge, while segment 2.1 has no knowledge. All segments have `float` as data type. We assume that in a current iteration of the knowledge discovery process at the collection level, we evaluate whether query-based similarity methods exist to determine if the pattern from Collection 1 can be identified in Collection 2. The system traverses the grammar productions and validates method requirements at each applicable level.

For a similarity search using Euclidean distance, the following conditions are evaluated:

- C_1 (uniform length): Evaluating all segments yields lengths {100, 100, 150, 120, 120}. The condition *fails* due to heterogeneous lengths.
- C_2 (uniform dtype): All segments have `float` data type. The condition *passes*.
- C_3 (has labels): Segment 2.1 has no associated `Label` knowledge. The condition *fails*.

Since Euclidean distance search is not applicable, the system evaluates Dynamic Time Warping (DTW) [BC94] as an alternative, which does not require uniform length. However, the example conditions still require labeled data for supervised search having the same data type. The system identifies a subset of segments satisfying the label condition: {1.1.1, 1.1.2, 2.2, 2.3}. For this filtered subset:

- C_2 (uniform dtype): All segments have `float` data type. The condition *passes*.
- C_3 (has labels): All segments in the subset have associated `Label` knowledge. The condition *passes*.

Consequently, DTW-based similarity search is selected as an applicable method. The pattern instances from segments 1.1.1 and 1.1.2 can be compared against segments 2.2 and 2.3 from Collection 2 to identify potential pattern occurrences.

This example illustrates how the framework navigates the method space by validating semantic conditions against the knowledge graph, automatically excluding inapplicable methods and identifying valid alternatives based on the available segment knowledge.

6.2.4 Summary and Future Work

The presented framework provides a systematic approach to support knowledge discovery by mapping data compositions to compatible analysis procedures. Through the grammar-based derivation of structural subsets and subsequent semantic validation, the space of applicable methods can be reduced before a metric-based evaluation takes place. This positions the approach as a preliminary stage for AutoML methods that typically assume a predefined task.

Beyond individual analyses, the framework can validate entire pipelines [Gra+23]. Since analytical findings stored in the knowledge graph enrich the available semantic information, which is formally grounded in the propagation algebra of Section 5.4.1, subsequent iterations of the discovery process benefit from previously generated knowledge. The integration with SPARQL enables declarative specification of method requirements that can be evaluated directly against the knowledge graph. This combines the expressiveness of semantic queries for graph-based conditions with the flexibility of programmatic evaluation for conditions requiring data access and provides a practical foundation for automated method selection in knowledge discovery workflows.

Future work could extend the approach to utilize AI-based guidance for a selection of methods as many methods might be compatible with data setups but not semantically appropriate. This further includes the consideration of graph information that provides semantic information of already conducted tasks or data-specific information. One realization of such guidance could leverage knowledge graph embeddings over populated SemTS graphs to learn latent representations of segments, associated knowledge, and methods, enabling a similarity-based ranking of condition-satisfying methods by their expected utility rather than treating all compatible methods equally. Regarding outcomes of executed analyses the system could further use existing knowledge to condition weights over time, thereby improving the quality of recommendations in iterative knowledge discovery scenarios. In the following section we will present an approach that seeks to close this gap through an agentic solution leveraging LLMs in combination with a SemTS-based graph memory to perform automated, informed data analytics.

6.3 LLM-based Data Science Agents

Time series analytics typically combines human expertise, classical statistical methods, and modern AI techniques to find and interpret patterns in a dataset. Selecting the right method and incorporating human judgment often requires substantial manual effort, which is especially difficult for users without a data science background.

In this section we present SAGE (Smart LLM Agents with Graph RAG for Time Series Exploration), a semantic agent framework that builds upon our prior work [Gra+25c] and related publications [Bac+25; Col+25]. SAGE organizes multiple LLM-powered agents in a layered structure where each agent has a specific role in exploring time series data. Together, these agents build up a shared knowledge base over time. At the center of the system sits a SemTS knowledge graph that acts as persistent memory, storing findings about the data, its properties, and the methods applied. By connecting these agents to the graph through Graph RAG, the framework can ground its outputs in real data while also pulling in relevant background information, for example historical events or domain knowledge that helps interpret the results.

After summarizing related work, we describe the conceptual foundation and system architecture of our framework. We then evaluate its effectiveness on forecasting tasks across different datasets using multiple LLMs.

6.3.1 Motivation and Related Work

Transformer-based models have significantly advanced time series analysis in recent years. Smaller, specialized transformer architectures have shown strong forecasting performance, particularly over long prediction horizons [Zho+22; Wu+21; Zho+21; Li+19]. Concurrently, the breakthrough of Large Language Models (LLMs) such as GPT [Ope23] and LLaMA [Tou+23] extended sequence modeling capabilities further. Due to their capacity for generating contextually rich natural language, foundation models are increasingly investigated for predicting and explaining numerical data. Several distinct strategies exist for applying these architectures to temporal tasks [Nie+23; CPC25; Jin+24; Liu+24; Jia+24c]. While early direct alignment efforts frequently leveraged smaller language models like GPT-2 [Rad+19] and BERT [Dev+19], modern methodologies attempt to overcome limitations with more comprehensive models.

However, applying general-purpose LLMs directly to complex time series problems is architecturally limited. Because they are not inherently pre-trained on high-frequency continuous data, general LLMs often fail to capture deep, time-dependent structures [Woo+24]. Fons et al. [Fon+24] demonstrate that standard

prompting yields poor downstream results because these models lack built-in token representations of geometric temporal dependencies and perform unreliably on precise numerical calculations. Cross-modal fine-tuning methods such as PromptCast [XS24] improve empirical accuracy but remain dependent on implicit, opaque model knowledge, which makes them prone to hallucinations. Conversely, static analytical toolkits and standalone question-answering pipelines [Sun+24] address only isolated elements of the problem, as they lack the reasoning-driven capabilities needed to adapt to dynamically changing contexts.

As foundation models developed stronger multi-step reasoning capabilities, research shifted toward using them as autonomous agents that generate and execute code to orchestrate downstream machine learning models [Qin+23; Bub+23]. Initially, applications of this agentic paradigm to time series analytical tasks remained limited [Jia+24c]. Recent developments have begun addressing these gaps. The emergence of agentic AI frameworks [Yao+23], adapted directly to temporal tasks like TimeCap [Lee+25], enables LLMs to function via autonomous loops capable of iterative planning, tool execution, and contextual self-refinement.

Nonetheless, a critical architectural limitation persists, which is that agentic approaches treat each execution block in isolation. They lack a structured, persistent semantic memory capable of incrementally storing and reusing analytical outcomes across distinct analyses. This problem motivated the need for the SemTS-based solution proposed in the following.

6.3.2 Approach

SAGE is built as a modular system of specialized agents that use LLMs and Graph RAG to perform time series analysis automatically. The foundation is a knowledge graph that follows the SemTS ontology and captures semantic information about time series datasets. Agents connect to this graph through Graph RAG, which lets them look up verified facts using semantic similarity search over embedded triples. This grounds every response and decision in structured, validated knowledge.

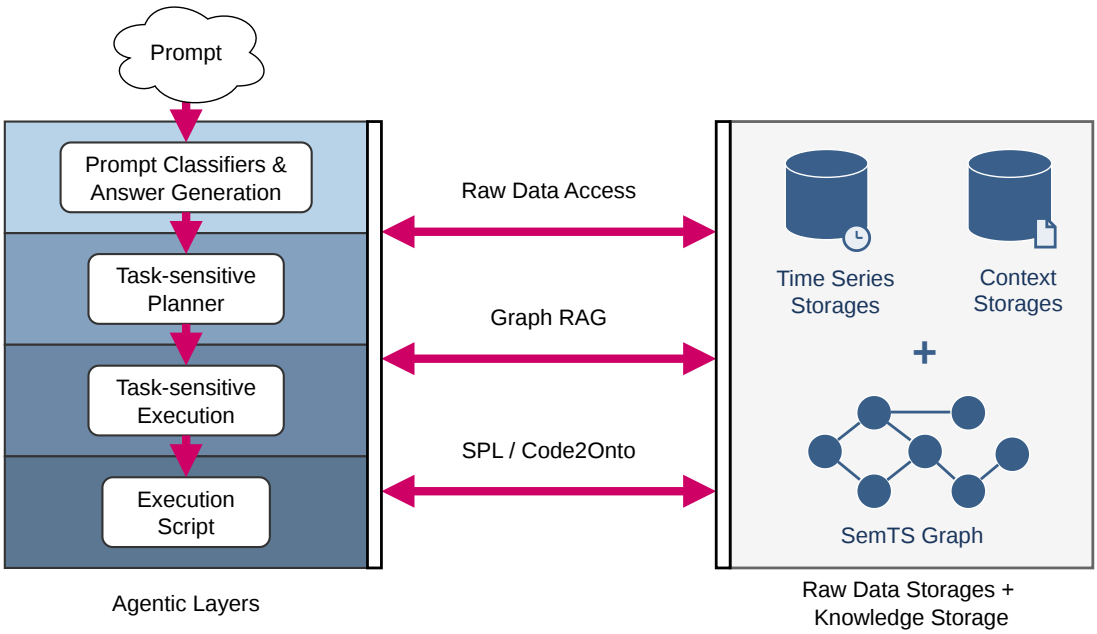


Figure 6.8: Hierarchical architecture of the agent system interacting with the incrementally enriched, unified knowledge graph. Red arrows indicate the workflow and data exchange.

The agents are organized in a hierarchy with four levels. At the top, a *top-level agent* receives user requests, classifies them, and formulates initial responses. Below that, *planning agents* gather relevant context and task specifications from the knowledge graph. *Execution agents* then pick suitable analysis models, set their parameters, and assemble scripts. At the lowest level, execution components run the generated code, and dedicated *code generation agents* handle requests that go beyond predefined capabilities. The framework also uses three types of retrievers to access different information sources. A graph retriever navigates the ontology structure and embeddings to find relevant entities and values. A code documentation retriever supplies model specifications and parameter descriptions. A raw data retriever locates storage paths, file formats, and encodings so that data loading works correctly. All these components interact continuously with the shared knowledge graph rather than operating in isolation.

The overall pipeline illustrated in Figure 6.8 is designed to avoid unnecessary computation. When a request arrives, the top-level agent first checks whether

the knowledge graph already contains enough information to answer it. If Graph RAG finds a match, the system returns the stored answer directly without triggering any further processing. If the needed information is missing, the system activates the lower levels. The planning layer combines graph knowledge with raw data context to create a detailed task description that specifies datasets, time ranges, and analysis parameters. The execution layer then selects a model, configures it, and prepares the analysis script. For unusual requests that fall outside the standard capabilities, the code generation agents write custom solutions. Throughout this process, every decision about model choice and parameter setting is informed by documentation and knowledge from the graph, which reduces trial and error and keeps results consistent.

What sets SAGE apart from other frameworks is its automated feedback loop. After every analysis, SAGE converts the results into formal ontology instances and writes them back into the knowledge graph. This means that analytical outputs like forecasts, detected anomalies, or cluster assignments become persistent, structured knowledge that the system can reuse later. To handle this conversion automatically, SAGE integrates Code2Onto [Gra+25b], which transforms analysis outputs from code into ontology individuals aligned with SemTS. This removes the need for manual mapping, preserves how results were produced, and systematically captures dataset details, parameter configurations, and outcomes in the unified graph. The knowledge graph therefore functions as a growing memory. For instance, anomalies found in one analysis can automatically be taken into account when a forecast is requested later (see Figure 4.1). Because Code2Onto itself uses agent-enhanced LLMs, it fits naturally into the rest of the agentic pipeline and completes the automation from raw request to stored result. Over time, this incremental enrichment leads to better retrievals, faster identification of suitable models for new tasks, and generally better analytical outcomes. A natural language user interface makes this entire process accessible to users regardless of their technical background.

6.3.3 Application Example

Figure 6.9 illustrates how a user initiates a forecasting request. The top-level

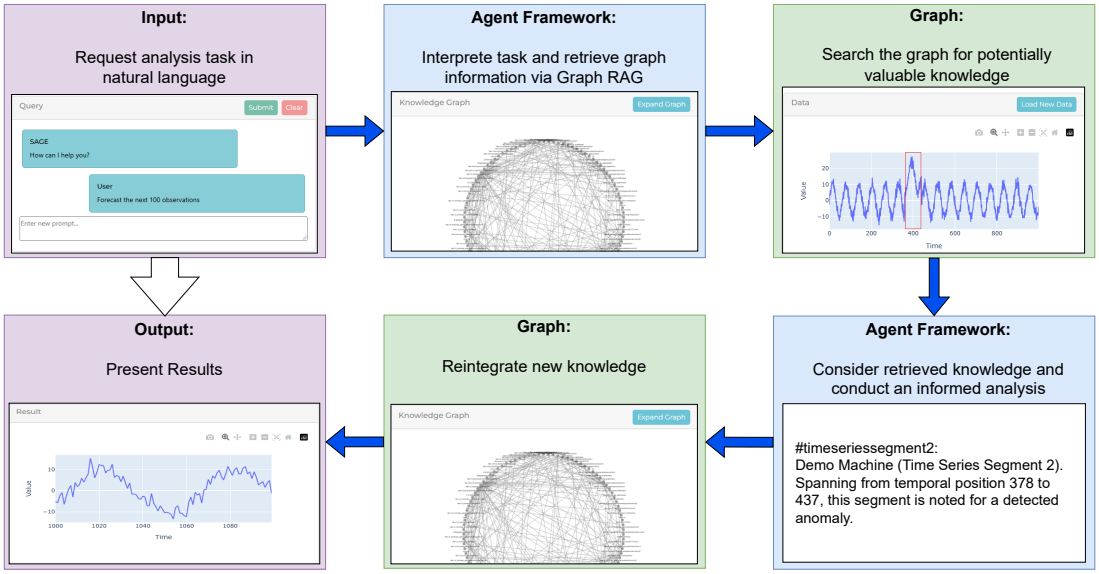


Figure 6.9: Application scenario on the example of a forecast. The identified anomaly from a previous analysis is considered during the forecast with new knowledge being reintegrated.

agent first queries the knowledge graph to check whether the answer is already available. If the existing graph content is not sufficient, the system classifies the request, in this case as a forecasting task, and notifies the user that a forecast will be created. In this example, a previous analysis had already identified an anomaly in the data. The system retrieves this anomaly and accounts for it during model training to prevent it from biasing the forecast. Next, SAGE activates the planning layer, which constructs a detailed task description by pulling information from the graph and the raw dataset. The execution layer then selects a forecasting model and determines its parameters. For this selection, it retrieves relevant documentation and dataset characteristics from the graph. Listing 6.3 shows a simple example of such retrieved knowledge, which in this case is the minimum value of the time series. The chosen model is then executed with the configured parameters. Once the forecast is complete, the results are written back into the knowledge graph so they can be reused in future queries. Finally, the response flows back up the hierarchy and the top-level agent presents the forecast to the user.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
xmlns:dcat="http://www.w3.org/ns/dcat#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:semts="https://w3id.org/semts/ontology#"
xmlns:time="http://www.w3.org/2006/time#"
>
<rdf:Description rdf:about="https://example.org/semts#
    SegmentDataCharacteristic_e0af2c3b-0634-4a3d-b462-56883481c230">
<rdf:type rdf:resource="https://w3id.org/semts/ontology#
    SegmentDataCharacteristic"/>
<semts:title>minimum</semts:title>
<semts:hasValue rdf:resource="https://example.org/semts#EmbeddedValue_883b9cd5
    -987a-4e78-a30a-ff0f7756ace3"/>
<semts:knowledgeConcept rdf:resource="https://w3id.org/semts/vocabulary/data-
    knowledge#MinValue"/>
<semts:knowledgeGenerationEntity rdf:resource="https://example.org/semts#
    KnowledgeGenerationMethod_0ad8af58-62a1-448b-ad28-338e78e02109"/>
</rdf:Description>
<rdf:Description rdf:about="https://example.org/semts#EmbeddedValue_883b9cd5
    -987a-4e78-a30a-ff0f7756ace3">
<rdf:type rdf:resource="https://w3id.org/semts/ontology#EmbeddedValue"/>
<semts:valueString>39.0</semts:valueString>
</rdf:Description>
</rdf:RDF>
```

Listing 6.3: A simple RDF example returned from the Graph RAG system showing a retrieved minimum value of a time series.

6.3.4 Evaluation

We evaluated SAGE along two dimensions. The first targets core functionality, specifically task classification, retrieval decisions, and factual accuracy. The second tests the framework on end-to-end forecasting tasks where it must select a model and produce a result. Parts of the core functionality evaluation reuse the experimental setup and results from earlier works [Gra+25c; Col+25]. Both evaluations use multiple state-of-the-art LLMs including DeepSeek [Guo+25], Mixtral [Jia+24a], Gemma [Gem+24], GPT-4o [Ope24a], GPT-OSS [Ope25] and LLaMA [Gra+24]. All agents are connected to the SemTS knowledge graph through Graph RAG.

Core Functionality Evaluation

We tested three fundamental capabilities of SAGE across several LLMs of mostly low to moderate complexity. The results are summarized in Figure 6.10 and were



Figure 6.10: SAGE Evaluation. Comparison of different metrics across models for zero-shot classification, Graph RAG information checks, and factual correctness tests.

first reported in [Gra+25c].

Task Classification. This test checks whether the framework can correctly assign user requests to the right analytical task. Getting this right matters because user queries are often ambiguous, for example when it is unclear whether a request asks for anomaly detection or classification. The test therefore also reveals which LLMs are suitable for the top-level agent role. Most models performed well. Gemma2 27b reached 87.33% accuracy and Llama-3 70b reached 87.67%, both with very low failure rates. Mixtral showed a higher failure rate and produced misclassifications or no classification at all for some queries.

Knowledge Availability Assessment. The second test measures whether the system can correctly decide if a user query can be answered from existing graph knowledge or whether a new analysis is needed. Accuracy was overall high on average with values being at least above 80%. False negative rates were low across all models, meaning the system rarely launches expensive computations unnecessarily. False positive rates were slightly higher for some models, where in practice a user can simply rephrase the query to trigger new analysis when needed.

Factual Accuracy. The third test measures whether graph-based answers to user

Dataset	Best Model	Result (MAE)
Airline Passengers	BATS	15.003
Lynx	AutoARIMA	487.639
Shampoo Sales	BATS	74.736
US Change	AutoARIMA	1.332
Longley	Prophet	6921.991
Macroeconomic	Theta	74.791
Solar	Theta	0.018

Table 6.1: Baseline results showing the best performing model and corresponding MAE for each dataset.

queries are correct. Llama-3 70b scored high on factual correctness, confirming that the framework can return precise answers for questions about trends or anomalies. Response times differed substantially between models. GPT-4o was the slowest at 60.14 seconds on average despite good accuracy, illustrating that there is a trade-off between quality and speed. Apart from the smaller DeepSeek models, correctness remained consistently high, which provides a reliable basis for the more complex forecasting evaluation that follows.

Forecasting Capability Evaluation

To test whether SAGE can perform useful forecasts, we compared its agent-driven model selection and execution against a set of baselines. We used the sktime library [Lön+19], which offers several forecasting algorithms (called estimators) and time series datasets. The baseline consists of all estimators run with default parameters and ranked by Mean Absolute Error (MAE). The estimator pool contains sktime's Naive Forecaster, AutoARIMA, AutoETS, AutoREG, Theta [AN00], Holt-Winters exponential smoothing [HA18], Croston [Cro72], BATS and TBATS [DHS11], Vector Autoregression, Prophet [TL18], TimeLLM using GPT-2 [Jin+24], and statsforecast [Gar+22] implementations of AutoARIMA, AutoETS, and AutoTheta. Table 6.1 lists the best performing model per dataset and Figure 6.11 shows the overall distribution of baseline performance.

Estimator Classification. This test checks whether the framework selects a suit-

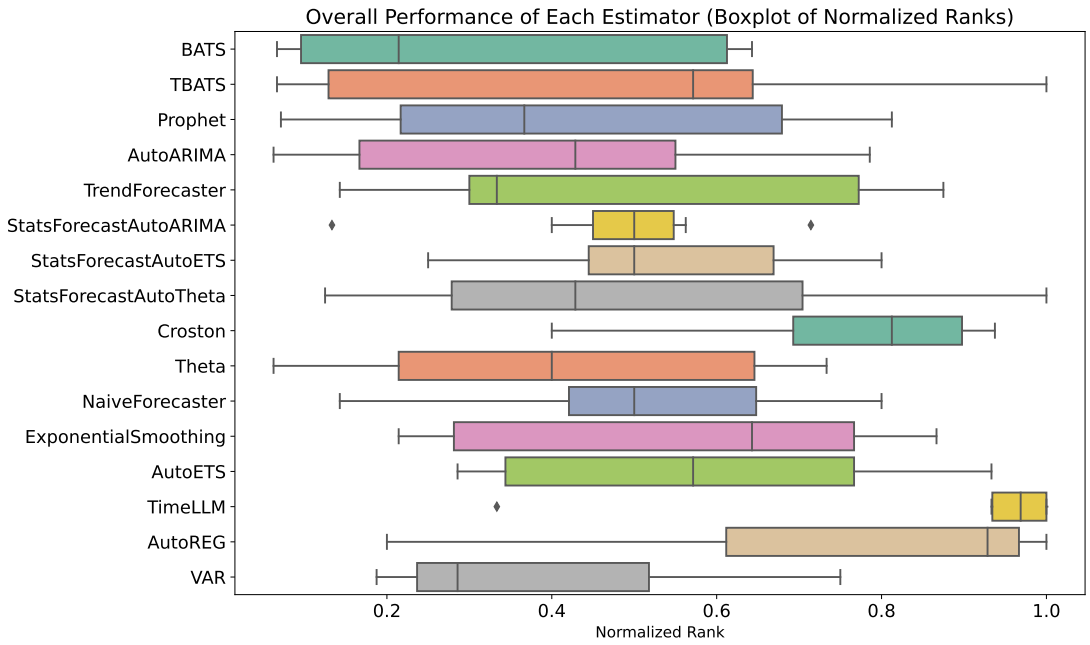


Figure 6.11: SAGE Evaluation. Baseline evaluation of all forecasting estimators, ranked by normalized position. Lower values correspond to higher positions and better results.

able forecasting estimator when given a dataset and a forecast request. The multi-agent system uses its retrieval mechanisms to gather information before making a selection, so this test also measures how much the knowledge graph helps with model choice. We measured whether the selected estimator matches the best baseline algorithm, and additionally whether it falls within the top 3, 5, or 7 performers. As shown in Figure 6.12, the four best open-source LLMs mostly picked estimators that rank among the top five across all tested scenarios. All four models outperformed random selection (computed as the corresponding probability of picking a top-k estimator by chance) on every metric, which confirms that the knowledge graph information meaningfully guides the selection process.

Estimator Execution. This test evaluates whether the framework can take a selected estimator and actually run it successfully. We tested three approaches for this. In the first, the agent generates the full instantiation code for the estimator. In the second, it completes a partial forecasting script. In the third, it outputs

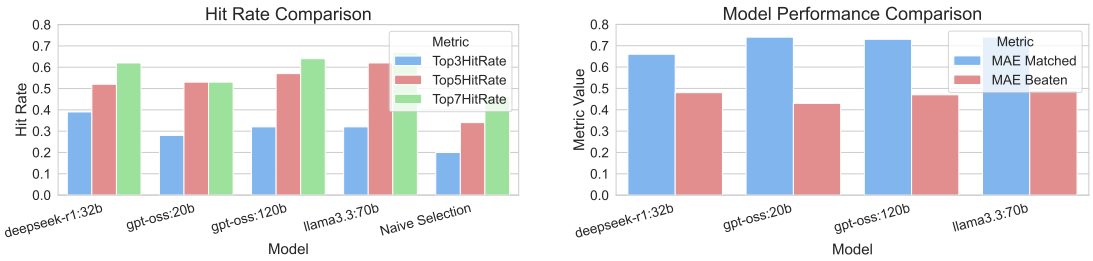


Figure 6.12: SAGE Evaluation. Performance analysis of the top four LLMs on successful tests.

the parameters as JSON which is then inserted into a template script. Each approach was tested with different retriever configurations. We evaluated results in stages, first checking whether valid code or JSON was produced, then whether it ran without errors (distinguishing instantiation errors from fit/predict errors), and finally comparing the forecast accuracy against the baseline MAE. As shown in Figure 6.12, the framework matched or beat the baseline for nearly all models in more than 70% of successfully executed cases, and in about 45% of those it produced a better forecast than the default-parameter baseline. These results show that semantically informed agent-driven analysis has practical value and that the framework can turn model selection into working forecasting pipelines that produce competitive results.

6.3.5 Summary and Future Work

In this section we introduced SAGE, a framework that automates data exploration by combining multiple specialized agents with an incrementally growing knowledge graph. The main idea behind SAGE is to reuse insights from earlier analyses while still being able to perform new analysis when needed. The framework connects its agents to a SemTS graph through Graph RAG, giving them access to structured information about time series data and past analyses. We further showed how the framework handles user queries along the whole analyses pipeline. This includes classifying the request, retrieving relevant information, running the analysis, and storing new results back in the graph. Our forecasting experiments show that the system makes reasonable model selection choices where

information from documentation and the knowledge graph is reused. The main limitation right now is that the graph retrieval mechanism does not always find the most relevant information. Future versions will address this by adding reranking strategies and better ways to filter and combine retrieved context. We also plan to evaluate SAGE in detail on additional time series tasks such as anomaly detection, classification, and clustering. In this context we also plan to address the open questions, which types of graph information help the most and how to best present them to LLMs of different sizes.

Chapter Summary

In this chapter we investigated RQ 3 by demonstrating how a SemTS knowledge graph can be exploited in modern data analytics workflows. First, with *SemTS Explorer* we demonstrated an interactive, web-based environment for visually exploring knowledge-enhanced time series segments, authoring Scenario Knowledge through expert-driven reasoning steps, and generating LLM-assisted narrative reports integrating capabilities. Through the *Conditioned Knowledge Discovery* framework, we introduced a context-free grammar for representing time series data hierarchies and a SPARQL-based condition evaluation mechanism that enables automated, semantically informed method selection as a knowledge discovery complement to AutoML. Finally, we demonstrated with *SAGE* how LLM-driven data science agents can leverage the SemTS knowledge graph as an incrementally evolving memory, where Graph RAG grounds agent decisions, while Code2Onto automatically reintegrates analytical outputs. The resulting cycle of retrieval, analysis, and enrichment enables informed, automated analytics accessible to users with varying levels of expertise.

What follows. Chapter 7 concludes this thesis by summarizing the contributions, reflecting on their limitations, and identifying directions for future research.

7 Conclusion and Future Work

In this thesis, we addressed the challenge of systematically classifying, managing, and exploiting knowledge derived from and associated with time series analytics. Driven by the observation that analytical insights commonly remain implicit, fragmented, or tied to isolated workflows, we pursued the goal of establishing a comprehensive, semantically structured knowledge graph that serves as a central, machine-interpretable repository for time series knowledge. The following paragraphs summarize this work through the three research questions that guided it.

The first step in our approach was to understand and classify knowledge, as a necessary foundation for managing and exploiting it. Addressing RQ 1, we therefore investigated which information qualifies as time series knowledge and how it can be structured. This investigation led us to distinguish two complementary knowledge types at the most abstract level: *Data Knowledge*, representing insights derived from analytical processes, and *Scenario Knowledge*, capturing contextual information. Rather than adopting the discrete, unidirectional progression of the DIKW hierarchy, we proposed a bidirectional, continuous model of knowledge significance that reflects the dynamic relationship between data-driven insights and contextual validation observed in practice. Through a systematic, iterative taxonomy development, we organized both knowledge types into hierarchical categories and formalized them as two machine-readable SKOS vocabularies. A coverage analysis against seven prominent time series libraries, a saturation check with ten independent libraries, and a cross-domain applicability evaluation across five representative domains confirmed partial practical completeness.

The next question was how to represent and manage the classified knowledge in a machine-interpretable form. Addressing RQ 2, we introduced the Seman-

tic Time Series Ontology (SemTS) as a segment-based knowledge representation that integrates with established schemas. SemTS organizes knowledge across three interconnected layers, namely Data Management for segment-based indexing, Knowledge Classification for semantically typing insights, and Knowledge Generation for representing how knowledge is produced and propagated with full provenance. The recursive definition of `KnowledgeGroup` aligned with SKOS hierarchies enables the representation of arbitrarily nested knowledge compositions, while axioms support reasoning and consistency checking and SHACL shapes enforce both structural and semantic validation.

To show the practical applicability of theoretical concepts, we developed two complementary approaches for SemTS population. The SemTS Population Library (SPL), to provide programmatic, type-safe ontology population, and Code2Onto as a multi-agent framework that automates ontology population from program code and its runtime context by splitting responsibilities between local runtime inspection and remote semantic validation. We further formalized knowledge propagation through a many-sorted algebraic specification including formal proof sketches and performed a mechanical validation using the Maude rewriting engine. Together, these results guarantee traceable provenance, monotone knowledge growth, and segment-safe distribution.

For knowledge retrieval, we conducted a systematic evaluation of graph embeddings across thirteen retrieval tasks organized into five structural categories. The evaluation demonstrated that embeddings can leverage the structured knowledge graph enriched by materialization across topology-dependent tasks, confirming that semantic structuring provides a measurable advantage for information access. These results, combined with the established effectiveness of integrating prior knowledge into machine learning pipelines as demonstrated in the Informed Machine Learning literature [Von+23; Kar+21; Rue+22], provide evidence for the central proposition of this thesis, that incrementally enriched, semantically structured knowledge graphs constitute a valuable foundation for cyclically informed analytics. Furthermore, the embedding evaluation to some extent addressed scalability aspects by operating on graphs of varying structural complexity, indicating that the retrieval mechanisms remain effective as the knowledge graph grows.

Addressing RQ 3, we demonstrated this exploitation through three complementary contributions. SemTS Explorer provides an interactive, web-based environment for visually exploring knowledge-enhanced time series segments, authoring Scenario Knowledge through expert-driven reasoning steps, and generating LLM-assisted narrative reports. The Conditioned Knowledge Discovery framework introduces a context-free grammar for representing time series data hierarchies and a SPARQL-based condition evaluation mechanism. Finally, SAGE demonstrates how LLM-driven data science agents can leverage the SemTS knowledge graph as an incrementally evolving memory, where Graph RAG grounds agent decisions in structured knowledge and Code2Onto automatically reintegrates analytical outputs into the graph.

In the thesis, different real-world application scenarios including computer vision for animal welfare, grinding bead wear forecasting, and anomaly detection in blow molding served as references to demonstrate the applicability and versatility of SemTS across different domains and analytical tasks. All ontology artifacts, vocabularies, SHACL shapes, code repositories, and documentation have been published under persistent W3ID identifiers and a CC BY 4.0 license.

7.1 Limitations

While the contributions of this thesis establish a comprehensive semantic framework for time series knowledge, the proposed solutions are subject to several limitations. We discuss these along three thematic threads: the boundaries of the classification and representation layer, the challenges of automated population and formal verification, and the constraints of the exploitation tools.

The Data Knowledge and Scenario Knowledge vocabularies were developed to be domain-agnostic and were evaluated across multiple libraries and application domains. Nevertheless, they are inherently bounded by the analytical paradigms and standards available at the time of development. As new types of analytical outputs emerge, for instance through novel generative model architectures, the vocabularies will require extension. The SKOS-based design explicitly supports such extension without modifying the core vocabulary. However, community gov-

ernance has not yet been established to manage this evolution systematically.

This concern extends to the ontology itself. SemTS deliberately employs a coarse ontological classification of knowledge into three types while delegating fine-grained distinctions to the SKOS-based vocabularies. This design ensures that the ontology can accommodate new knowledge types without structural modification, but it also means that constraints on novel knowledge types can only be enforced once they have been defined in the vocabularies.

Regarding scalability, the embedding evaluation demonstrated effective retrieval across graphs of varying sizes, providing initial evidence that the approach scales beyond small configurations. The full performance characteristics under production-scale conditions involving millions of time series segments and continuous enrichment over extended periods remain to be characterized in deployed systems.

Code2Onto automates the population process through LLM-guided planning, which introduces inherent non-determinism. Even with conservatively chosen temperature settings and validation, identical source code can produce varying ontology populations across runs. The evaluation demonstrated solid accuracy for the best-performing model, though a portion of the generated semantic content benefits from subsequent validation. The runtime context resolver currently supports standard Python object navigation and would benefit from extensions for more complex patterns such as dynamically generated properties. We consider these aspects as practical engineering refinements rather than fundamental constraints of the approach.

We conducted the embedding evaluation on synthetically generated graphs designed to reflect realistic structural patterns. While we identified consistent advantages of structured representations, validating these findings on real-world SemTS graphs populated through actual analytical workflows would strengthen external validity. We also observed through ablation that the information included in retrieval representations benefits from selection. Providing overly comprehensive metadata can dilute discriminative signal in the fixed-dimensional embedding space, reducing retrieval precision for specific tasks. This finding points toward the need for task-aware retrieval strategies that select relevant knowledge

subsets rather than presenting all available information at once.

The exploitation layer, comprising SemTS Explorer, the Conditioned Knowledge Discovery framework and SAGE, demonstrates how the knowledge graph can be used in practice, but each tool operates within a defined scope. SemTS Explorer currently operates on static, user-uploaded knowledge graphs and does not yet support real-time data integration or collaborative multi-user workflows. These boundaries define the current scope for operational monitoring scenarios where knowledge graphs evolve continuously. The grammar-based Conditioned Knowledge Discovery framework provides a formalism for representing time series hierarchies and evaluating method requirements against the knowledge graph. However, the expressiveness of the grammar for edge cases, such as overlapping but non-nested segments, has not yet been formally characterized. For SAGE, we have empirically validated forecasting as the first analytical task using baseline models with default parameters. Extending the evaluation to additional tasks and comparing against tuned baselines would provide a broader picture of the framework's capabilities.

7.2 Future Work

The limitations discussed above, together with open research directions that emerged throughout this work, define a future research agenda. We organize future work along the same three thematic threads.

As an important need we identified a structured process for vocabulary evolution. The SKOS vocabularies are designed as living documentation, maintainable through the public GitHub repository, but the absence of a formal governance process limits community-driven growth. We plan to establish contribution guidelines, review workflows, and periodic release cycles with semantic versioning to ensure that the vocabularies keep pace with emerging analytical paradigms. Of particular interest is the systematic extension of Scenario Knowledge coverage for domains with weaker representation in the current vocabulary, such as finance contexts, where our cross-domain evaluation identified the need for more precise terms.

For Code2Onto, we plan to improve the depth and accuracy of code analysis through customizable runtime context resolvers tailored to common data science libraries. Incorporating human-in-the-loop interactions at critical decision points would allow developers to confirm or correct the LLM's mapping before finalization. We also plan to evaluate the framework with structurally diverse ontologies beyond SemTS to further substantiate the claim of ontology-agnosticism.

Moreover, we plan to validate the embedding results on real-world SemTS graphs from the industrial application scenarios to confirm the observed advantages of structured representations in production settings. The finding that presenting all available knowledge at once can reduce retrieval precision motivates the development of task-aware retrieval strategies that dynamically select relevant information subsets based on the query context.

A central direction for future work concerns the direct empirical validation of incremental enrichment over extended operational periods. The central proposition that cyclically informed analytics outperform independent workflows is indirectly supported by two converging lines of evidence we indicated in this thesis and that are grounded in the broader literature. First, the Informed Machine Learning literature demonstrates that integrating prior knowledge into machine learning pipelines improves performance. Second, the embedding evaluation conducted in this thesis shows that the structured, semantically typed representations provided by SemTS yield measurable retrieval advantages over unstructured alternatives, confirming that the ontological design effectively makes knowledge accessible for downstream consumption. Together, these results establish that SemTS provides structured knowledge in a form that demonstrably supports retrieval and reuse. What remains is the longitudinal validation of the cumulative effect. We plan to track how a SemTS knowledge graph evolves over many enrichment cycles in a real-world deployment, measuring whether and to what extent the compounding knowledge leads to progressively better analytical outcomes.

The exploitation tools developed in this thesis are solutions in progress that we will continue to develop and refine.

For SemTS Explorer we plan to extend the application with interactive segment definition to remove the current requirement that segments must already exist in

the uploaded graph.

For the Conditioned Knowledge Discovery framework, we plan to leverage knowledge from previously conducted analyses to refine method rankings and condition weights over time, thereby improving the quality of recommendations in iterative knowledge discovery scenarios. Formally characterizing the grammar's expressiveness, in particular identifying configurations it cannot represent and determining whether extensions are needed, would be an appropriate addition.

Regarding SAGE, an obvious next step is the evaluation of additional analytical tasks beyond forecasting. Of particular interest to us is the analysis of which graph information contributes most to improving agent performance and how this information should optimally be presented to LLMs of varying sizes and capabilities.

Cutting across all three threads, a longitudinal case study tracking the evolution of a SemTS knowledge graph over an extended period of continuous enrichment in an industrial deployment would address several of the open questions identified above simultaneously. Such a study would reveal how knowledge quality evolves over successive enrichment cycles, whether the formal properties of the propagation algebra hold in practice, how the graph's growing size affects retrieval and reasoning performance, and how the vocabularies need to be extended as new analytical paradigms are adopted. We are planning this kind of sustained, real-world validation with future projects, as it would provide the most comprehensive assessment of the incremental enrichment proposition.

This thesis demonstrates that the semantic organization of time series knowledge through ontology-based knowledge graphs offers a viable and valuable path towards transforming isolated analytical workflows into cyclically informed processes. By establishing standardized classifications, formal representations, and practical tools for populating, retrieving, propagating, and exploiting time series knowledge, the proposed SemTS ecosystem contributes to making data-driven insights more accessible, traceable, and valuable across domains and expertise levels.

Code repository (GitHub): https://github.com/semts-ontology.

7.3 Vision

The concept of incremental knowledge graph enrichment explored in this thesis addresses a shift in how intelligent systems operate and learn over time. Today, many AI systems lack the ability to retain and build upon custom factual knowledge and new discoveries over time as evidenced by the emergence and significance of GraphRAG systems. The work presented here provides a foundation for systems that accumulate understanding persistently, where every interaction contributes to a shared, growing intelligence that potentially benefits from subsequent interactions. Importantly, this accumulation is not limited to what machines generate automatically. The graph equally grows through the contributions of domain experts, analysts and decision makers who validate, annotate and refine what is stored, effectively working on a shared knowledge base.

This idea becomes particularly relevant as AI systems take on increasingly autonomous roles. Current large language models and AI agents already demonstrate strong reasoning abilities. Connecting such systems to incrementally enriched, semantically structured graphs gives them access to validated, traceable facts that improve over time. As these graphs mature through continuous enrichment cycles, AI systems transition to tools that remember, contextualize and anticipate. A sufficiently populated graph could support AI agents in identifying relevant analyses based on accumulated evidence to identify that a newly observed pattern shares structural similarity with a validated explanation elsewhere, or that a particular configuration has consistently worked well under specific conditions.

While this thesis demonstrates the principle through time series analytics, the underlying idea of incremental semantic enrichment is not limited to this domain. LLM-based agents already follow a similar logic when they search the internet for up-to-date information. Any field where knowledge is generated iteratively and where past findings could inform future work can benefit from the same architectural pattern. Manufacturing, medical diagnostics or civil engineering routinely produce insights that are used once and never revisited. Applying the same principle of structured, persistent and incrementally growing knowledge to these areas would allow organizations and AI systems alike to build on what

already existed rather than repeatedly starting over.

At an organizational level, this means that analytical knowledge becomes a durable asset rather than a temporary outcome of individual projects. Accumulated understanding would no longer be lost to staff turnover, technology migrations or shifting priorities. The semantically structured graph would not only preserve what was found but potentially why it matters, how confident we are and under which conditions it holds. Future analysts and AI agents can then build on a foundation that is validated and grows richer with every interaction.

Looking further ahead, the presented solution could enable knowledge transfer at scale. Where domains share structural similarities, patterns validated in manufacturing could inform monitoring in healthcare. Anomaly explanations from energy systems could accelerate diagnostics in transportation. Standardized ontologies and vocabularies could facilitate federated knowledge ecosystems where insights can be transferred between domains using shared semantic standards. Combined with the rapid advancement of AI, these ecosystems could form the persistent memory layer that current AI systems often lack, enabling systems to build on shared, structured knowledge rather than operating independently.

Bibliography

[AAH23] Abdulaziz Aldoseri, Khalifa N Al-Khalifa, and Abdel Magid Hamouda. "Re-thinking data strategy and integration for artificial intelligence: concepts, opportunities, and challenges". In: *Applied Sciences* 13.12 (2023), p. 7082 (cit. on p. 1).

[Abb+20] Ralph Abboud, Ismail Ilkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. "BoxE: A Box Embedding Model for Knowledge Base Completion". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 9649–9661 (cit. on pp. 133, 135).

[Abb12] Sunitha Abburu. "A survey on ontology reasoners and comparison". In: *International Journal of Computer Applications* 57.17 (2012), pp. 33–39 (cit. on p. 90).

[Aca+17] Giovanni Acampora, Autilia Vitiello, Bruno Di Stefano, Wil van der Aalst, Christian Günther, and Eric Verbeek. "IEEE 1849: The XES standard". In: *IEEE Computational Intelligence Magazine* 12.2 (2017), pp. 4–8 (cit. on p. 57).

[Ack89] Russell L. Ackoff. "From Data to Wisdom". In: *Journal of Applied Systems Analysis* 16.1 (1989), pp. 3–9 (cit. on p. 35).

[AF94] James F Allen and George Ferguson. "Actions and events in interval temporal logic". In: *Journal of logic and computation* 4.5 (1994), pp. 531–579 (cit. on p. 91).

[AG00] Eugene Agichtein and Luis Gravano. "Snowball: Extracting relations from large plain-text collections". In: *Proceedings of the fifth ACM conference on Digital libraries*. 2000, pp. 85–94 (cit. on p. 115).

[AG08] Renzo Angles and Claudio Gutierrez. "Survey of graph database models". In: *ACM computing surveys (CSUR)* 40.1 (2008), pp. 1–39 (cit. on p. 30).

[Aga13] Shivam Agarwal. "Data mining: Data mining concepts and techniques". In: *2013 international conference on machine intelligence and research advancement*. IEEE. 2013, pp. 203–207 (cit. on p. 20).

[AGM85] Carlos E Alchourrón, Peter Gärdenfors, and David Makinson. "On the logic of theory change: Partial meet contraction and revision functions". In: *The Journal of Symbolic Logic* 50.2 (1985), pp. 510–530. DOI: 10.2307/2274239 (cit. on p. 153).

[AGN15] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. "Profiling relational data: a survey". In: *The VLDB Journal* 24.4 (Aug. 2015), pp. 557–581. ISSN: 1066-8888. DOI: 10.1007/s00778-015-0389-y (cit. on p. 42).

[AI21] Riccardo Albertoni and Antoine Isaac. "Introducing the data quality vocabulary (DQV)". In: *Semantic Web* 12.1 (2021), pp. 81–97 (cit. on p. 56).

[Al-21] Samaher Al-Janabi. "Overcoming the main challenges of knowledge discovery through tendency to the intelligent data analysis". In: *2021 international conference on data analytics for business and industry (ICDABI)*. IEEE. 2021, pp. 286–294 (cit. on p. 21).

[AL01] Maryam Alavi and Dorothy E Leidner. "Knowledge management and knowledge management systems: Conceptual foundations and research issues". In: *MIS quarterly* 25.1 (2001), pp. 107–136 (cit. on p. 35).

[Alb+24] Riccardo Albertoni, David Browning, Simon Cox, Alejandra Gonzalez Beltran, Andrea Perego, and Peter Winstanley. *Data Catalog Vocabulary (DCAT) - Version 3*. W3C Recommendation. World Wide Web Consortium (W3C), 2024. URL: https://www.w3.org/TR/vocab-dcat-3/ (cit. on pp. 74, 79).

[Ale+20] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. "GluonTS: Probabilistic and Neural Time Series Modeling in Python". In: *Journal of Machine Learning Research* 21.116 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/19-820.html (cit. on p. 292).

[Ali+19] Mohammed Ali, Mark W Jones, Xianghua Xie, and Mark Williams. "TimeCluster: dimension reduction applied to temporal data for visual analytics". In: *The Visual Computer* 35.6 (2019), pp. 1013–1026 (cit. on p. 183).

[Ali+21] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. "PyKEEN 1.0: a python library for training and evaluating knowledge graph embeddings". In: *Journal of Machine Learning Research* 22.82 (2021), pp. 1–6 (cit. on p. 135).

[Ali20] Moez Ali. *PyCaret: An open source, low-code machine learning library in Python*. PyCaret version 1.0.0. 2020. URL: https://www.pycaret.org (cit. on p. 292).

[All83] James F. Allen. "Maintaining knowledge about temporal intervals". In: *Communications of the ACM* 26.11 (1983), pp. 832–843. DOI: 10.1145/182.358434 (cit. on p. 91).

[Alq+21] Ali Alqahtani, Mohammed Ali, Xianghua Xie, and Mark W Jones. "Deep time-series clustering: A review". In: *Electronics* 10.23 (2021), p. 3001 (cit. on pp. 20, 22, 52).

[Als+22] Ahmad Alsharef, Karan Aggarwal, Sonia, Manoj Kumar, and Ashutosh Mishra. "Review of ML and AutoML solutions to forecast time-series data". In: *Archives of computational methods in engineering* 29.7 (2022), pp. 5297–5311 (cit. on p. 16).

[Alv+21] Sergio Alvarez-Napagao, Boki Ashmore, Marta Barroso, Cristian Barrué, Christian Beecks, Fabian Berns, Ilaria Bosi, Sisay Adugna Chala, Nicola Ciulli, Marta Garcia-Gasulla, Alexander Grass, Dimosthenis Ioannidis, Natalia Jakubiak, Karl Köpke, Ville Lämsä, Pedro Megias, Alexandros Nizamis, Claudio Pastrone, Rosaria Rossini, Miquel Sànchez-Marrè, and Luca Ziliotti. "knowlEdge Project – Concept, Methodology and Innovations for Artificial Intelligence in Industry 4.0". In: *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*. SAN orchestrated the publication in the scope of the knowlEdge EU project (under grant agreement 957331). All authors contributed with their in-

dividual expertise to the corresponding sections. 2021, pp. 1–7 (cit. on p. 106).

[AN00] Vassilis Assimakopoulos and Konstantinos Nikolopoulos. "The theta model: a decomposition approach to forecasting". In: *International Journal of Forecasting* 16.4 (2000), pp. 521–530. ISSN: 0169-2070. DOI: 10.1016/S0169-2070(00)00066-2. URL: https://www.sciencedirect.com/science/article/pii/S0169207000000662 (cit. on p. 209).

[And11] Theodore W Anderson. *The statistical analysis of time series*. John Wiley & Sons, 2011 (cit. on pp. 41, 44).

[Ani+11] Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic. "EP-SPARQL: a unified language for event processing and stream reasoning". In: *Proceedings of the 20th international conference on World wide web*. 2011, pp. 635–644 (cit. on p. 173).

[Ans+23] Yaqoob Ansari, Omar Mourad, Khalid Qaraqe, and Erchin Serpedin. "Deep learning for ECG Arrhythmia detection and classification: an overview of progress for period 2017–2023". In: *Frontiers in Physiology* 14 (2023), p. 1246746 (cit. on p. 53).

[AP11] Marcelo Arenas and Jorge Pérez. "Querying semantic web data with SPARQL". In: *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2011, pp. 305–316 (cit. on pp. 27, 32).

[Apa15] Apache Software Foundation. *Apache Atlas - Data Governance and Metadata Framework*. https://atlas.apache.org/. 2015 (cit. on p. 184).

[Apa21] Apache Software Foundation. *Apache Jena*. 2021. URL: https://jena.apache.org/ (cit. on p. 109).

[APM15] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. "Leveraging linguistic structure for open domain information extraction". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 344–354 (cit. on p. 115).

[Ara+19] Gustavo Aragón, Harsh Puri, Alexander Grass, Sisay Chala, and Christian Beecks. "Incremental deep-learning for continuous load prediction in energy management systems". In: *2019 IEEE Milan PowerTech*. HP developed and implemented the discussed approaches, GA drafted the publication and supported with domain expertise, AG and SC supported with technical expertise, CB reviewed and revised the publication. IEEE. 2019, pp. 1–6 (cit. on p. 51).

[Ari+20] Paola Ariza Colpas, Enrico Vicario, Emiro De-La-Hoz-Franco, Marlon Pineres-Melo, Ana Oviedo-Carrascal, and Fulvio Patara. "Unsupervised human activity recognition using the clustering approach: A review". In: *Sensors* 20.9 (2020), p. 2702 (cit. on p. 52).

[Ash+23] Mohsena Ashraf, Farzana Anowar, Jahanggir H Setu, Atiqul I Chowdhury, Eshtiak Ahmed, Ashraful Islam, and Abdullah Al-Mamun. "A survey on dimensionality reduction techniques for time-series data". In: *IEEE Access* 11 (2023), pp. 42909–42923 (cit. on p. 22).

[ASW15] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering–a decade review". In: *Information systems* 53 (2015), pp. 16–38 (cit. on pp. 20, 22, 52).

[AV19] Automotive Industry Action Group (AIAG) and Verband der Automobilindustrie (VDA). *AIAG & VDA FMEA Handbook. Failure Mode and Effects Analysis*. 1st ed. AIAG and VDA QMC, 2019 (cit. on p. 59).

[AW10] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459 (cit. on p. 53).

[Ayr+02] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. "Sequential pattern mining using a bitmap representation". In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 429–435 (cit. on p. 53).

[BAB24] Kanaiya Bhatt, Chetan Agrawal, and Anil M Bisen. "A review on emerging applications of IoT and sensor technology for Industry 4.0". In: *Wireless Personal Communications* 134.4 (2024), pp. 2371–2389 (cit. on p. 1).

[Bac+24] Manlio Bacco, Alexander Kocian, Stefano Chessa, Antonino Crivello, and Paolo Barsocchi. "What are data spaces? Systematic survey and future outlook". In: *Data in Brief* 57 (2024), p. 110969 (cit. on p. 27).

[Bac+25] Nicolas Bach, Christian Beecks, Gesa Benndorf, Julia Burr, Franziska Dechert, Frederik Diederichs, Hartmut Eigenbrod, Georg Fuchs, Alexander Graß, Varun Gowtham, Jan Hamaekers, Dirk Hecker, Sonja Holl-Supra, Maximilian Hörstrup, Bjarne Huuck, Wilhelm Jaspers, Wolfgang Kratsch, Christopher Krauss, Jonathan Krauß, Christian Kühnert, André Kummerow, Matthis Leicht, Gerhard Paass, Angelika Voß, Pascal Nicolai Linden, Miriam Mock, Tim Wirtz, Daniel David Scherer, Christian Ufrecht, Maximilian Röglinger, Gyunam Park, Wil van der Aalst, Niklas Uhl, Florian Rippstein, Jonathan Ott, Christopher Mutschler, Maximilian Stahlke, Alper Yaman, Christian Seidler, Julian Raible, Mathias Ziebarth, Andreas Wilde, Ina Steinmetz, Janko Petereit, Alexander Zorn, Christian Pionzewski, Andrej Plenne, Florian Strohm, and Felix Ott. *Jenseits von Text und Bild - Generative KI für vielfältige Datenwelten: Praxisnahe Einblicke und innovative Anwendungsbeispiele zur Erschließung komplexer Unternehmensdaten*. Ed. by Fraunhofer-Allianz Big Data und Künstliche Intelligenz. Publication in the scope of the Fraunhofer-Allianz Big Data und Künstliche Intelligenz. All authors contributed with their individual expertise to the corresponding sections. Fraunhofer Verlag, 2025. 64 pp. ISBN: 978-3-8396-2133-2 (cit. on p. 202).

[Bag+17] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances". In: *Data mining and knowledge discovery* 31.3 (2017), pp. 606–660 (cit. on p. 23).

[Bak+05] Tom Baker, Makx Dekkers, Thomas Fischer, and Rachel Heery. *Dublin Core™ Application Profile Guidelines*. Tech. rep. Dublin Core Metadata Initiative, Sept. 2005. URL: https://dublincore.org/specifications/dublin-core/application-profile-guidelines/ (cit. on p. 33).

[Bar+10] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. "Querying RDF streams with C-SPARQL". In: *ACM SIGMOD Record* 39.1 (2010), pp. 20–26 (cit. on p. 173).

[Bar+20] Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. "TSFEL: Time series feature extraction library". In: *SoftwareX* 11 (2020), p. 100456 (cit. on p. 20).

[Bar02] Stuart Barnes, ed. *Knowledge Management Systems: Theory and Practice*. London: Thomson Learning, 2002 (cit. on p. 35).

[BB19] Karianne J Bergen and Gregory C Beroza. "Earthquake fingerprints: Extracting waveform features for similarity-based earthquake detection". In: *Pure and Applied Geophysics* 176.3 (2019), pp. 1037–1059 (cit. on p. 51).

[BC04] Max Boisot and Agustí Canals. "Data, information and knowledge: have we got it right?" In: *Journal of evolutionary economics* 14.1 (2004), pp. 43–67 (cit. on p. 35).

[BC94] Donald J Berndt and James Clifford. "Using dynamic time warping to find patterns in time series". In: *AAAI-94 Workshop on Knowledge Discovery in Databases*. 1994, pp. 359–370 (cit. on pp. 51, 200).

[BCM04] Gene Bellinger, Durval Castro, and Anthony Mills. *Data, information, knowledge, and wisdom*. http://www.systems-thinking.org/dikw/dikw.htm. 2004 (cit. on p. 35).

[Bee+19] Christian Beecks, Kjeld Willy Schmidt, Fabian Berns, and Alexander Graß. "Gaussian Processes for Anomaly Description in Production Environments". In: *Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference*. Ed. by Paolo Papotti. Vol. 2322. CEUR Workshop Proceedings. CB drafted the publication, KWS and FB supported the implementation process. AG reviewed and revised the publication. 2019 (cit. on p. 46).

[Bee+24a] Christian Beecks, Anandraj Amalraj, Alexander Graß, Marc Jentsch, Felix Kitschke, Maximilian Norz, and Patric Schäffer. "Leveraging YOLO for Real-Time Video Analysis of Animal Welfare in Pig Slaughtering Processes". In: *German Conference on Artificial Intelligence (Künstliche Intelligenz)*. CB, AG and AA took care of the methodology and the implementation. FK supported the implementation, MJ and MN reviewed and revised the publication, PS supported with domain expertise. Springer. 2024, pp. 275–281 (cit. on p. 103).

[Bee+24b] Christian Beecks, Alexander Graß, Anandraj Amalraj, Marc Jentsch, Felix Kitschke, Maximilian Norz, and Patric Schäffer. "Entwicklung einer KI für automatisierte Tierschutzkontrollen in der Schweineschlachtung". In: *INFORMATIK 2024*. Ed. by Maike Klein, Daniel Krupka, Cornelia Winter, Martin Gergeleit, and Ludger Martin. Lecture Notes in Informatics (LNI) P-352. CB, AG and AA took care of the methodology and the implementation. FK supported the implementation, MJ and MN reviewed and revised the publication, PS supported with domain expertise. Bonn: Gesellschaft für Informatik e.V., 2024, pp. 1463–1477 (cit. on p. 102).

[Ben12] Colin Bentley. *PRINCE2: a practical handbook*. Routledge, 2012 (cit. on p. 59).

[Ber06] Tim Berners-Lee. *Linked Data – Design Issues*. https://www.w3.org/DesignIssues/LinkedData.html. Accessed: 2025-05-12. 2006 (cit. on p. 93).

[BG14] Dan Brickley and R. V. Guha. *RDF Schema 1.1*. W3C Recommendation REC-rdf-schema-20140225. World Wide Web Consortium (W3C), Feb. 2014. URL: https://www.w3.org/TR/rdf-schema/ (cit. on p. 28).

[BG16] Christian Beecks and Alexander Graß. "Multi-step threshold algorithm for efficient feature-based query processing in large-scale multimedia databases". In: *2016 IEEE International Conference on Big Data (Big Data)*. CB drafted the publication, AG supported with the development and implementation of the discussed approach. IEEE. 2016, pp. 596–605 (cit. on p. 51).

[Bha+21] Aadyot Bhatnagar, Paul Kassianik, Chenghao Liu, Tian Lan, Wenzhuo Yang, Rowan Cassius, Doyen Sahoo, Devansh Arpit, Sri Subramanian, Gerald Woo, Amrita Saha, Arun Kumar Jagota, Gokulakrishnan Gopalakrishnan, Manpreet Singh, K. C. Krithika, Sukumar Maddineni, Daeki Cho, Bo Zong, Yingbo Zhou, Caiming Xiong, Silvio Savarese, Steven Hoi, and Huan Wang. "Merlion: A Machine Learning Library for Time Series". In: *arXiv preprint arXiv:2109.09265* (2021) (cit. on p. 292).

[Bha23] Jasmin Bharadiya. "The impact of artificial intelligence on business processes". In: *European journal of technology* 7.2 (2023), pp. 15–25 (cit. on p. 1).

[BHL23] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities". In: *Linking the world's information: essays on Tim Berners-Lee's invention of the World Wide Web*. ACM, 2023, pp. 91–103 (cit. on p. 16).

[BHP16] Eva Blomqvist, Karl Hammar, and Valentina Presutti. "Engineering ontologies with patterns–the eXtreme design methodology". In: *Ontology Engineering with Ontology Design Patterns*. IOS Press, 2016, pp. 23–50 (cit. on p. 78).

[Bif+24] Stefan Biffl, Sebastian Kropatschek, Kristof Meixner, David Hoffmann, and Arndt Lüder. "Configuring and Validating Multi-aspect Risk Knowledge for Industry 4.0 Information Systems". In: *Advanced Information Systems Engineering: 36th International Conference, CAiSE 2024*. Berlin, Heidelberg: Springer-Verlag, 2024, pp. 492–508. ISBN: 978-3-031-61056-1. DOI: 10.1007/978-3-031-61057-8_29 (cit. on p. 59).

[BJ70] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day, 1970 (cit. on p. 51).

[BK07] Derya Birant and Alp Kut. "ST-DBSCAN: An algorithm for clustering spatial–temporal data". In: *Data & knowledge engineering* 60.1 (2007), pp. 208–221 (cit. on p. 52).

[BLZ+19] Junjie Bai, Fang Lu, Ke Zhang, et al. *ONNX: Open Neural Network Exchange*. https://github.com/onnx/onnx. 2019 (cit. on p. 63).

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge, UK: Cambridge University Press, 1998. DOI: 10.1017/CBO9781139172752 (cit. on pp. 172, 329, 332).

[Bor+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. "Translating embeddings for modeling multi-relational data". In: *Advances in Neural Information Processing Systems*. Vol. 26. 2013, pp. 2787–2795 (cit. on pp. 31, 133, 135).

[Bos+16] Terry Bossomaier, Lionel Barnett, Michael Harré, and Joseph T Lizier. "Transfer entropy". In: *An introduction to transfer entropy: Information flow in complex systems*. Springer, 2016, pp. 65–95 (cit. on p. 48).

[Box+15] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015 (cit. on pp. 17, 21, 37).

[BPS10] Harold Boley, Adrian Paschke, and Omair Shafiq. "RuleML 1.0: The Overarching Specification of Web Rules". In: *Semantic Web Rules*. Vol. 6403. Lecture Notes in Computer Science. Springer, 2010, pp. 162–178 (cit. on p. 59).

[Bra+06] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. *Extensible Markup Language (XML) 1.1 (Second Edition)*. W3C Recommendation REC-xml11-20060816. World Wide Web Consortium (W3C), Aug. 2006. URL: https://www.w3.org/TR/2006/REC-xml11-20060816/ (cit. on p. 28).

[Bri98] Sergey Brin. "Extracting patterns and relations from the world wide web". In: *International workshop on the world wide web and databases*. Springer. 1998, pp. 172–183 (cit. on p. 115).

[Bro+15] Kay H Brodersen, Fabian Gallusser, Jim Koehler, Nicolas Remy, and Steven L Scott. "Inferring causal impact using Bayesian structural time-series models". In: *The Annals of Applied Statistics* 9.1 (2015), pp. 247–274. DOI: 10.1214/14-AOAS788 (cit. on p. 48).

[BS04] Christian Bizer and Andy Seaborne. "D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs". In: *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004), Poster Track*. 2004 (cit. on pp. 109, 115).

[Bub+23] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. "Sparks of artificial general intelligence: Early experiments with GPT-4". In: *arXiv preprint arXiv:2303.12712* (2023) (cit. on p. 203).

[Cal+17] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. "Ontop: Answering SPARQL queries over relational databases". In: *Semantic Web* 8.3 (2017), pp. 471–487 (cit. on p. 109).

[Cao+24] Jiahang Cao, Jinyuan Fang, Zaiqiao Meng, and Shangsong Liang. "Knowledge graph embedding: A survey from the perspective of representation spaces". In: *ACM Computing Surveys* 56.6 (2024), pp. 1–42 (cit. on pp. 27, 31, 132).

[Car26] David Carlisle. *Mathematical Markup Language (MathML) Version 4.0*. W3C Working Draft. World Wide Web Consortium (W3C), 2026. URL: https://www.w3.org/TR/mathml4/ (cit. on p. 58).

[Cas+12] Carmelo Cassisi, Placido Montalto, Marco Aliotta, Andrea Cannata, and Alfredo Pulvirenti. "Similarity measures and dimensionality reduction techniques for time series data mining". In: *Advances in Data Mining Knowledge Discovery and Applications*. Ed. by Adem Karahoca. Rijeka: InTech, 2012, pp. 71–96 (cit. on p. 51).

[CD97] Surajit Chaudhuri and Umeshwar Dayal. "An overview of data warehousing and OLAP technology". In: *ACM Sigmod record* 26.1 (1997), pp. 65–74 (cit. on p. 18).

[CG23] Sisay Adugna Chala and Alexander Graß. "Human-AI Interaction for Semantic Knowledge Enrichment of AI Model Output". In: *Artificial Intelligence in Manufacturing: Enabling Intelligent, Flexible and Cost-Effective Production Through AI*. SC drafted the publication. AG reviewed and revised the publication. Springer Nature Switzerland Cham, 2023, pp. 43–54 (cit. on p. 54).

[Cha00] Chris Chatfield. *Time-series forecasting*. Chapman and Hall/CRC, 2000 (cit. on p. 37).

[Cha22] Harrison Chase. *LangChain*. https://github.com/langchain-ai/langchain. Accessed: 2025-10-26. 2022 (cit. on p. 116).

[Chr+18] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)". In: *Neurocomputing* 307 (2018), pp. 72–77. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2018.03.067. URL:

https://www.sciencedirect.com/science/article/pii/S0925231218304843 (cit. on pp. 20, 67, 76, 193).

[Chr12] Peter Christen. "The data matching process". In: *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer, 2012, pp. 23–35 (cit. on p. 115).

[Chr26] Maximilian Christ. *Awesome Time Series*. https://github.com/MaxBenChrist/awesome_time_series_in_python. GitHub repository, accessed 2026-01-18. 2026 (cit. on p. 41).

[CI21] Giuseppe Ciaburro and Gino Iannace. "Machine learning-based algorithms to knowledge extraction from time series data: A review". In: *Data* 6.6 (2021), p. 55 (cit. on p. 20).

[CK10] Sven F Crone and Nikolaos Kourentzes. "Feature selection for time series prediction–A combined filter and wrapper approach for neural networks". In: *Neurocomputing* 73.10-12 (2010), pp. 1923–1936 (cit. on p. 21).

[CK21] Mainak Chakraborty and Ajit Pratap Kundan. "Grafana". In: *Monitoring Cloud-Native Applications*. Berkeley, CA: Apress, 2021, pp. 187–240. DOI: 10.1007/978-1-4842-6888-9_6 (cit. on p. 184).

[CKK24] Felipe A Csaszar, Harsh Ketkar, and Hyunjin Kim. "Artificial intelligence and strategic decision-making: Evidence from entrepreneurs and investors". In: *Strategy Science* 9.4 (2024), pp. 322–345 (cit. on p. 1).

[CL22] Simon Cox and Chris Little. *Time Ontology in OWL*. 2022. URL: https://www.w3.org/TR/owl-time/ (visited on 04/17/2024) (cit. on p. 78).

[Cla+20] Manuel Clavel, Francisco Durán, Steven Eker, Santiago Escobar, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, Rubén Rubio, and Carolyn Talcott. *Maude Manual (Version 3.1)*. SRI International. 2020 (cit. on p. 172).

[CLW69] James W Cooley, Peter AW Lewis, and Peter D Welch. "The fast Fourier transform and its applications". In: *IEEE Transactions on Education* 12.1 (1969), pp. 27–34 (cit. on p. 46).

[CM05] Dan Connolly and Libby Miller. *RDF Calendar – An Application of the Resource Description Framework to iCalendar Data*. W3C Interest Group Note NOTE-rdfcal-20050929. World Wide Web Consortium (W3C), Sept. 2005. URL: https://www.w3.org/TR/rdfcal/ (cit. on p. 57).

[CMS24] Vitor Cerqueira, Nuno Moniz, and Carlos Soares. "VEST: Automatic feature engineering for forecasting". In: *Machine Learning* 113.7 (2024), pp. 4523–4545 (cit. on p. 25).

[Col+25] Diego Collarana, Christopher Ingo Pack, Yan-Ying Liao, Marlena Flüh, Jonathan Lehmkuhl, Abhishek Nadgeri, Alexander Graß, Moritz Busch, Prinon Das, Lara Dingels, Stefan Decker, and Christian Beecks. "Graph RAG in the Wild: Insights and Best Practices from Real-World Applications". In: *Semantic Web – Interoperability, Usability, Applicability* (2025). Under review (minor revision); All authors contributed to their corresponding application listed in the publication. (cit. on pp. 99, 127, 202, 207).

[CPC25] Ching Chang, Wenjie Peng, and Tien-Fu Chen. "LLM4TS: Aligning Pre-Trained LLMs as Data-Efficient Time-Series Forecasters". In: *ACM Transactions on Intelligent Systems and Technology* (2025). DOI: 10.1145/3719207. arXiv: 2308.08469. URL: https://arxiv.org/abs/2308.08469 (cit. on p. 202).

[CR14] Richard Cyganiak and Dave Reynolds. *The RDF Data Cube Vocabulary*. World Wide Web Consortium (W3C) Recommendation. Jan. 2014. URL: https://www.w3.org/TR/vocab-data-cube/ (cit. on p. 74).

[Cro72] John D Croston. "Forecasting and stock control for intermittent demands". In: *Journal of the operational research society* 23.3 (1972), pp. 289–303 (cit. on p. 209).

[CZC18] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE transactions on knowledge and data engineering* 30.9 (2018), pp. 1616–1637 (cit. on p. 133).

[Das+98] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. "Rule Discovery from Time Series". In: *Proceedings of*

the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98). 1998, pp. 16–22 (cit. on p. 20).

[Dav19] Cameron Davidson-Pilon. "lifelines: survival analysis in Python". In: *Journal of Open Source Software* 4.40 (2019), p. 1317. DOI: 10.21105/joss.01317 (cit. on p. 67).

[DB-25] DB-Engines. *DB-Engines Ranking Categories*. https://db-engines.com/en/ranking_categories. Accessed: 2025-08-01. 2025 (cit. on p. 72).

[DCM12] DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, June 2012. URL: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/ (cit. on p. 29).

[Del+09] Emanuele Della Valle, Stefano Ceri, Frank Van Harmelen, and Dieter Fensel. "It's a streaming world! Reasoning upon rapidly changing information". In: *IEEE Intelligent Systems* 24.6 (2009), pp. 83–89 (cit. on p. 173).

[Del+14] Daniele Dell'Aglio, Emanuele Della Valle, Jean-Paul Calbimonte, and Oscar Corcho. "RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems". In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.4 (2014), pp. 17–44 (cit. on p. 173).

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186 (cit. on p. 202).

[Dey01] Anind K Dey. "Understanding and using context". In: *Personal and ubiquitous computing* 5.1 (2001), pp. 4–7 (cit. on p. 37).

[DF79] David A Dickey and Wayne A Fuller. "Distribution of the estimators for autoregressive time series with a unit root". In: *Journal of the American statistical association* 74.366a (1979), pp. 427–431 (cit. on p. 48).

[DHR15] Laura Daniele, Frank den Hartog, and Jasper Roes. "Created in close interaction with the industry: the smart appliances reference (SAREF) ontology". In: *International Workshop Formal Ontologies Meet Industries*. Springer. 2015, pp. 100–112 (cit. on p. 174).

[DHS11] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. "Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing". In: *Journal of the American Statistical Association* 106.496 (2011), pp. 1513–1527. DOI: 10.1198/jasa.2011.tm09771 (cit. on p. 209).

[Din+07] Li Ding, Pranam Kolari, Zhongli Ding, and Sasikanth Avancha. "Using ontologies in the semantic web: A survey". In: *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*. Springer, 2007, pp. 79–113 (cit. on p. 27).

[DKK+18] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, et al. *The UCR Time Series Classification Archive*. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. Oct. 2018 (cit. on p. 95).

[Doc26] Docker Inc. *Docker*. 2026. URL: https://www.docker.com (cit. on p. 172).

[Doy79] Jon Doyle. "A truth maintenance system". In: *Artificial intelligence* 12.3 (1979), pp. 231–272 (cit. on p. 35).

[DP98] Thomas H. Davenport and Laurence Prusak. *Working Knowledge: How Organizations Manage What They Know*. Boston, MA: Harvard Business School Press, 1998 (cit. on p. 37).

[DR02] Hong-Hai Do and Erhard Rahm. "COMA—a system for flexible combination of schema matching approaches". In: *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Morgan Kaufmann, 2002, pp. 610–621 (cit. on p. 115).

[Duv+13] David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Zoubin Ghahramani. "Structure discovery in nonparametric regression through compositional kernel search". In: *International Conference on Machine Learning*. PMLR. 2013, pp. 1166–1174 (cit. on p. 46).

[Duv14] David Duvenaud. "Automatic model construction with Gaussian processes". PhD thesis. University of Cambridge, 2014 (cit. on p. 46).

[DYD24] Ioannis Dasoulas, Duo Yang, and Anastasia Dimou. "MLSea: A Semantic Layer for Discoverable Machine Learning". In: *European Semantic Web Conference*. Springer. 2024, pp. 178–198 (cit. on pp. 62, 71, 75, 90).

[EA12] Philippe Esling and Carlos Agon. "Time-series data mining". In: *ACM Computing Surveys (CSUR)* 45.1 (2012), pp. 1–34 (cit. on pp. 20, 41).

[Edg+24] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*. arXiv:2404.16130 [cs]. Apr. 2024. URL: http://arxiv.org/abs/2404.16130 (cit. on p. 150).

[EER+20] Tarek Elsaleh, Shirin Enshaeifar, Roonak Rezvani, et al. "IoT-Stream: A lightweight ontology for internet of things data streams and its use with data analytics and event detection services". In: *Sensors* 20.4 (2020), p. 953 (cit. on p. 75).

[Ela11] Elastic NV. *Kibana*. Elastic NV. 2011. URL: https://www.elastic.co (cit. on p. 184).

[Eld+21] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. "Time-series representation learning via temporal and contextual contrasting". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*. 2021, pp. 2352–2359. DOI: 10.24963/ijcai.2021/324. arXiv: 2106.14112 (cit. on p. 21).

[EMN+15] Diego Esteves, Diego Moussallem, Ciro Baron Neto, et al. "MEX vocabulary: a lightweight interchange format for machine learning experiments". In: *Proceedings of the 11th International Conference on Semantic Systems*. 2015, pp. 169–176 (cit. on p. 75).

[Est+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. 1996, pp. 226–231 (cit. on p. 52).

[Etz+11] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. "Open information extraction: The second generation". In: *IJCAI*. 2011, pp. 3–10 (cit. on p. 115).

[Eur24] European Commission. *DCAT Application Profile for Data Portals in Europe (DCAT-AP)*. Specification. Version 3.0.0. European Commission, Interoperable Europe (SEMIC Support Centre), June 2024. URL: https://interoperable-europe.ec.europa.eu/collection/semic-support-centre/solution/dcat-application-profile-data-portals-europe/release/300 (cit. on p. 33).

[Fac21] Facebook Infrastructure Data Science. *Kats: a kit to analyze time series data in Python*. https://github.com/facebookresearch/Kats. 2021 (cit. on p. 68).

[Fao22] Johann Faouzi. "Time series classification: A review of algorithms and implementations". In: *Machine Learning (Emerging Trends and Applications)*. Proud Pen, 2022 (cit. on pp. 22, 52).

[FB18] Diogo Fernandes and Jorge Bernardino. "Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB". In: *Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018)*. SciTePress, 2018, pp. 373–380. DOI: 10.5220/0006910203730380 (cit. on p. 31).

[FDJ19] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. "Unsupervised scalable representation learning for multivariate time series". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019, pp. 4650–4661 (cit. on p. 21).

[Fir23] Omari Firas. "A combination of SEMMA & CRISP-DM models for effectively handling big data using formal concept analysis based knowledge discovery: A data mining approach". In: *World Journal of Advanced Engineering Technology and Sciences* 8.1 (2023), pp. 009–014 (cit. on p. 20).

[Fon+24] Elizabeth Fons, Rachneet Kaur, Soham Palande, Zhen Zeng, Tucker Balch, Manuela Veloso, and Svitlana Vyetrenko. "Evaluating Large Language Models on Time Series Feature Understanding: A Comprehensive Taxonomy and Benchmark". In: *Proceedings of the 2024 Conference*

on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami. 2024, pp. 21598–21634 (cit. on p. 202).

[Fra+16] Max Franz, Christian T. Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D. Bader. "Cytoscape.js: a graph theory library for visualisation and analysis". In: *Bioinformatics* 32.2 (2016), pp. 309–311. DOI: 10.1093/bioinformatics/btv557 (cit. on p. 188).

[Fra+18] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. "Cypher: An evolving query language for property graphs". In: *Proceedings of the 2018 international conference on management of data*. 2018, pp. 1433–1445 (cit. on p. 32).

[Fra22] Fraunhofer FIT DSAI. *AP-first: Methodology for Developing Application Profiles*. https://github.com/Fraunhofer-FIT-DSAI/drk-information-model. Accessed: 2025-08-01. 2022 (cit. on p. 100).

[Fra25] Fraunhofer Institute for Applied Information Technology (FIT). *Next Generation Dataspaces Initiative (NGDI)*. Accessed: 2025-08-01. 2025. URL: https://www.fit.fraunhofer.de/en/business-areas/data-science-and-artificial-intelligence/next-generation-dataspaces-initiative-NGDI.html (cit. on p. 99).

[Fri04] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2004. ISBN: 978-0471471639 (cit. on p. 58).

[Ful18] Ben D Fulcher. "Feature-based time-series analysis". In: *Feature engineering for machine learning and data analytics*. CRC press, 2018, pp. 87–116 (cit. on pp. 20, 44).

[FV17] Amin Fakhrazari and Hamid Vakilzadian. "A survey on time series data mining". In: *2017 IEEE International Conference on Electro Information Technology (EIT)*. IEEE. 2017, pp. 476–481 (cit. on p. 20).

[FWB22] Yuwei Fu, Di Wu, and Benoit Boulet. "Reinforcement Learning Based Dynamic Model Combination for Time Series Forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.6 (June 2022), pp. 6639–6647. DOI: 10.1609/aaai.v36i6.20618. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20618 (cit. on p. 23).

[FXJ20] Yujie Fang, Hui Xu, and Jie Jiang. "A survey of time series data visualization research". In: *IOP Conference Series: Materials Science and Engineering* 782.2 (2020), p. 022013 (cit. on p. 183).

[Gal+24] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. "Towards Foundation Models for Knowledge Graph Reasoning". In: *International Conference on Learning Representations (ICLR)*. 2024. arXiv: 2310.04562 (cit. on p. 150).

[Gam17] John Cristian Borges Gamboa. "Deep learning for time-series analysis". In: *arXiv preprint arXiv:1701.01887* (2017) (cit. on p. 21).

[Gar+22] Federico Garza, Max Mergenthaler Canseco, Cristian Challú, and Kin G Olivares. *StatsForecast: Lightning fast forecasting with statistical and econometric models*. PyCon Salt Lake City, Utah, US. 2022. URL: https://github.com/Nixtla/statsforecast (cit. on p. 209).

[Gar17] Daniel Garijo. "WIDOCO: a wizard for documenting ontologies". In: *International Semantic Web Conference*. Springer, Cham. 2017, pp. 94–102. URL: http://dgarijo.com/papers/widoco-iswc2017.pdf (cit. on pp. 78, 93).

[Gar23] Daniel Garijo. *A checklist for complete vocabulary metadata*. https://dgarijo.github.io/Widoco/doc/bestPractices/index-en.html. Accessed: 2025-05-12. 2023 (cit. on p. 93).

[GBC18] Alexander Graß, Christian Beecks, and José Ángel Carvajal Soto. "Unsupervised anomaly detection in production lines". In: *Machine Learning for Cyber Physical Systems: Selected papers from the International Conference ML4CPS 2018*. AG developed the approach, JAC and CB reviewed and revised the publication. Springer. 2018, pp. 18–25 (cit. on pp. 45, 49).

[GBD24] Alexander Graß, Christian Beecks, and Stefan Decker. "Structural and Semantic Data Layers in Time Series Analyses". In: *International Conference on Intelligent Data Engineering and Automated Learning*. AG drafted the publication, CB helped to refine the conceptual idea, SD reviewed and revised the publication. Springer. 2024, pp. 505–511 (cit. on pp. 182, 192, 196).

[GCP21] Daniel Garijo, Oscar Corcho, and María Poveda-Villalón. "FOOPS!: An Ontology Pitfall Scanner for the FAIR Principles". In: *International Semantic Web Conference (ISWC) 2021: Posters, Demos, and Industry Tracks*. Vol. 2980. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2980/paper321.pdf (cit. on pp. 78, 94).

[GDB22] Alexander Graß, Till Döhmen, and Christian Beecks. "Sample-based Kernel Structure Learning with Deep Neural Networks for Automated Structure Discovery". In: *2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW)*. AG developed the approach, TD and CB reviewed and revised the publication. IEEE. 2022, pp. 79–83 (cit. on pp. 46, 80).

[Gem+24] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. "Gemma 2: Improving open language models at a practical size". In: *arXiv preprint arXiv:2408.00118* (2024) (cit. on p. 207).

[Ger09] R. Gerhards. *RFC 5424: The Syslog Protocol*. IETF RFC 5424. 2009. URL: https://www.rfc-editor.org/rfc/rfc5424 (cit. on p. 57).

[GF18] Palash Goyal and Emilio Ferrara. "Graph embedding techniques, applications, and performance: A survey". In: *Knowledge-Based Systems* 151 (2018), pp. 78–94 (cit. on p. 132).

[GKT07] Todd J Green, Grigoris Karvounarakis, and Val Tannen. "Provenance semirings". In: *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2007, pp. 31–40. DOI: 10.1145/1265530.1265535 (cit. on p. 153).

[GL16] Aditya Grover and Jure Leskovec. "node2vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. New York, NY, USA: ACM, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754 (cit. on p. 135).

[GL23] Artur d'Avila Garcez and Luis C Lamb. "Neurosymbolic AI: the 3rd wave". In: *Artificial Intelligence Review* 56.11 (2023), pp. 12387–12406.

DOI: 10.1007/s10462-023-10448-w. arXiv: 2012.05876 (cit. on pp. 174, 179).

[Gli+14] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. "HermiT: an OWL 2 reasoner". In: *Journal of Automated Reasoning* 53.3 (2014), pp. 245–269 (cit. on p. 32).

[GMM03] Ramanathan Guha, Rob McCool, and Eric Miller. "Semantic Search". In: *Proceedings of the 12th International Conference on World Wide Web*. 2003, pp. 700–709 (cit. on p. 30).

[GNT03] Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. "An information theoretic tradeoff between complexity and accuracy". In: *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*. Springer. 2003, pp. 595–609 (cit. on p. 24).

[Gra+23] Alexander Graß, Christian Beecks, Sisay Adugna Chala, Christoph Lange, and Stefan Decker. "A Knowledge Graph for Query-Induced Analyses of Hierarchically Structured Time Series Information". In: *European Conference on Advances in Databases and Information Systems*. AG developed the concept, CB and SD guided the development, SC and CL reviewed and revised the publication. Springer. 2023, pp. 174–184 (cit. on pp. 76, 152, 201).

[Gra+24] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 (cit. on p. 207).

[Gra+25a] Alexander Graß, Rohit A. Deshmukh, Christian Beecks, and Stefan Decker. "Towards an Ontology for Representing Time Series Knowledge: Motivation, Requirements and Concept". In: *Intelligent Information Systems. CAiSE 2025*. Ed. by Luise Pufahl, Kristina Rosenthal, Sergio España, and Selmin Nurcan. SD initiated the idea, AG and RD developed the ontology, CB reviewed and revised the publication. Cham: Springer Nature Switzerland, 2025, pp. 103–110 (cit. on pp. 72, 76).

[Gra+25b] Alexander Graß, Jonathan Lehmkuhl, Diego Collarana, Stefan Decker, and Christian Beecks. "Code2Onto: Multi-Agent System for Code-Driven Ontology Population". In: *2025 IEEE International Conference on Big Data (BigData)*. AG developed the approach, JL helped with literature review, SD optimized the initial idea, DC and CB reviewed and revised the publication. 2025, pp. 5642–5651 (cit. on pp. 94, 108, 113, 116, 118, 121, 122, 125, 205).

[Gra+25c] Alexander Graß, Christopher I Pack, Diego Collarana, Stefan Decker, and Christian Beecks. "Semantic Intelligence: Graph RAG-Driven Agents for Time Series Analytics". In: *International Conference on Intelligent Data Engineering and Automated Learning*. AG developed the knowledge integration and CP developed the agentic approach, DC, CB and SD reviewed and revised the publication. Springer. 2025, pp. 226–231 (cit. on pp. 99, 127, 182, 202, 207, 208).

[Gra+26] Alexander Graß, Rohit A Deshmukh, Christoph Lange, Diego Collarana, Christian Beecks, and Stefan Decker. "SemTS: Ontology and Vocabularies for the Semantic Categorization of Time Series Knowledge". In: *European Semantic Web Conference*. SD initiated the idea, AG and RD developed the ontology, CL provided support with technical details, DC and CB reviewed and revised the publication. Springer. 2026, pp. 172–190 (cit. on pp. 72, 74, 82, 95, 96).

[Gra69] Clive W. J. Granger. "Investigating Causal Relations by Econometric Models and Cross-spectral Methods". In: *Econometrica* 37.3 (1969), pp. 424–438 (cit. on p. 48).

[Gu+24] Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I Wang. "CRUXEval: A benchmark for code reasoning, understanding and execution". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. 2024, pp. 16568–16621. arXiv: 2401.03065 (cit. on p. 122).

[Guo+25] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. "DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement

learning". In: *Nature* (2025). DOI: 10.1038/s41586-025-09422-z. arXiv: 2501.12948 (cit. on p. 207).

[HA18] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018 (cit. on p. 209).

[Haa+08] Peter Haase, Holger Lewen, Rudi Studer, Duc Thanh Tran, Michael Erdmann, Mathieu d'Aquin, and Enrico Motta. "The NeOn ontology engineering toolkit". In: *Proceedings of the 17th International World Wide Web Conference (WWW 2008), Developers Track*. 2008 (cit. on p. 109).

[Hal+17] Armin Haller, Krzysztof Janowicz, Simon J. D. Cox, Danh Le Phuoc, Kerry Taylor, and Maxime Lefrançois. *Semantic Sensor Network Ontology*. W3C Recommendation REC-vocab-ssn-20171019. World Wide Web Consortium (W3C), Oct. 2017. URL: https://www.w3.org/TR/vocab-ssn/ (cit. on p. 29).

[Ham20] James D Hamilton. *Time series analysis*. Princeton university press, 2020 (cit. on pp. 2, 17, 41).

[Han+19] Awni Y Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H Tison, Codie Bourn, Mintu P Turakhia, and Andrew Y Ng. "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network". In: *Nature medicine* 25.1 (2019), pp. 65–69 (cit. on p. 2).

[Han+24a] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. "Retrieval-augmented generation with graphs (GraphRAG)". In: *arXiv preprint arXiv:2501.00309* (2024) (cit. on pp. 26, 31).

[Han+24b] Jiuzhou Han, Nigel Collier, Wray Buntine, and Ehsan Shareghi. "PiVe: Prompting with iterative verification improving graph-based generative capability of LLMs". In: *Findings of the Association for Computational Linguistics: ACL 2024*. Association for Computational Linguistics, 2024, pp. 6702–6718. DOI: 10.18653/v1/2024.findings-acl.400. arXiv: 2305.12392 (cit. on p. 116).

[HAN21] Maki K Habib, Samuel A Ayankoso, and Fusaomi Nagata. "Data-driven modeling: concept, techniques, challenges and a case study". In: *2021 IEEE international conference on mechatronics and automation (ICMA)*. IEEE. 2021, pp. 1000–1007 (cit. on p. 16).

[Har+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2 (cit. on p. 122).

[HB11] Matthew Horridge and Sean Bechhofer. "The OWL API: A Java API for OWL ontologies". In: *Semantic web* 2.1 (2011), pp. 11–21 (cit. on p. 109).

[Hea92] Marti A Hearst. "Automatic acquisition of hyponyms from large text corpora". In: *COLING 1992 volume 2: The 14th international conference on computational linguistics*. 1992 (cit. on p. 115).

[Her+22] Julien Herzen, Francesco Lässig, Samuele Giuliano Piazzetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan Kościsz, Dennis Bader, Frédérick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and Gaël Grosch. "Darts: User-Friendly Modern Machine Learning for Time Series". In: *Journal of Machine Learning Research* 23.124 (2022), pp. 1–6 (cit. on p. 67).

[Her14] Ivan Herman. *OWL-RL: A simple OWL2 RL reasoner on top of RDFLib*. Version citable-version. Oct. 2014. DOI: 10.5281/zenodo.14543 (cit. on p. 111).

[HK87] Richard Hull and Roger King. "Semantic database modeling: Survey, applications, and research issues". In: *ACM Computing Surveys (CSUR)* 19.3 (1987), pp. 201–260 (cit. on p. 27).

[HKR09] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. Chapman and Hall/CRC, 2009 (cit. on pp. 16, 27).

[Hog+21] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. "Knowledge graphs". In: *ACM Computing Surveys (Csur)* 54.4 (2021), pp. 1–37 (cit. on pp. 16, 27, 30).

[Hor+04] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission. World Wide Web Consortium (W3C), May 2004. URL: https://www.w3.org/submissions/SWRL/ (cit. on p. 174).

[HPR19] Franziska Horn, Robert Pack, and Michael Rieger. "The autofeat python library for automated feature engineering and selection". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pp. 111–120 (cit. on p. 25).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 51).

[Hu26] Shuguo Hu. *awesome-time-series-papers*. https://github.com/TSCenter/awesome-time-series-papers. GitHub repository, accessed 2026-01-18. 2026. URL: https://github.com/TSCenter/awesome-time-series-papers (cit. on p. 41).

[Hua+16] Xiaohui Huang, Yunming Ye, Liyan Xiong, Raymond YK Lau, Nan Jiang, and Shaokai Wang. "Time series k-means: A new k-means type smooth subspace clustering for time series data". In: *Information Sciences* 367 (2016), pp. 1–13 (cit. on p. 52).

[Hui+22] Bei Hui, Lizong Zhang, Xue Zhou, Xiao Wen, and Yuhui Nian. "Personalized recommendation system based on knowledge embedding and historical behavior". In: *Applied Intelligence* 52.1 (2022), pp. 954–966 (cit. on p. 132).

[IEE16] IEEE Standards Association. *IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams*. Standard IEEE Std 1849-2016. IEEE, 2016. DOI: 10.1109/IEEESTD.2016.7740858. URL: https://ieeexplore.ieee.org/document/7740858 (cit. on p. 57).

[Inf15] InfluxData/Chronograf Team. *Chronograf*. 2015. URL: https://www.influxdata.com (cit. on p. 184).

[Int14] International Organization for Standardization. *Automation systems and integration – Key performance indicators for manufacturing operations management – Part 1: Overview and concepts*. Standard ISO 22400-1:2014. International Organization for Standardization, 2014. URL: https://www.iso.org/obp/ui/#iso:std:iso:22400:-1:ed-1:v1:en (cit. on p. 58).

[Int15] International Organization for Standardization. *Quality management systems - Requirements*. Standard ISO 9001:2015. Geneva: International Organization for Standardization, 2015. URL: https://www.iso.org/iso-9001-quality-management.html (cit. on pp. 59, 60).

[Int16a] International Organization for Standardization. *Petroleum, petrochemical and natural gas industries – Collection and exchange of reliability and maintenance data for equipment*. Standard ISO 14224:2016(E). International Organization for Standardization, 2016. URL: https://www.iso.org/standard/64076.html (cit. on p. 57).

[Int16b] International Society of Automation (ISA). *Management of Alarm Systems for the Process Industries*. Standard. Research Triangle Park, NC: International Society of Automation, 2016. URL: https://www.isa.org/products/ansi-isa-18-2-2016-management-of-alarm-systems-for (cit. on p. 58).

[Int18] International Electrotechnical Commission. *Failure modes and effects analysis (FMEA and FMECA)*. Standard IEC 60812:2018. International Electrotechnical Commission, 2018. URL: https://webstore.iec.ch/publication/26359 (cit. on p. 59).

[Int21] International Organization for Standardization. *Quality management systems – Guidance for documented information*. Tech. rep. ISO 10013:2021.

International Organization for Standardization, 2021. URL: https://www.iso.org/standard/75736.html (cit. on p. 59).

[Int22] International Electrotechnical Commission. *Management of alarm systems for the process industries*. Standard IEC 62682. International Electrotechnical Commission, 2022. URL: https://webstore.iec.ch/publication/7363 (cit. on p. 58).

[Int24] International Organization for Standardization. *Asset management – Vocabulary, overview and principles*. Standard ISO 55000:2024. Version 2. International Organization for Standardization, 2024. URL: https://www.iso.org/standard/83053.html (cit. on p. 57).

[ISA08] ISA-88 Standards Committee. *Machine and Unit States: An Implementation Example of ISA-88*. Technical Report ISA-TR88.00.02-2008. International Society of Automation, 2008. URL: https://webstore.ansi.org/standards/isa/isatr8800022008 (cit. on p. 58).

[ISA10] ISA. *Enterprise-Control System Integration – Part 1: Models and Terminology*. Standard ANSI/ISA-95.00.01-2010. International Society of Automation, 2010 (cit. on p. 58).

[ISA95] ISA (International Society of Automation). *ANSI/ISA-88.01-1995: Batch Control Part 1: Models and Terminology*. Standard. ISA, 1995 (cit. on p. 58).

[ISB22] Samuel Idowu, Daniel Strüber, and Thorsten Berger. "Asset Management in Machine Learning: State-of-research and State-of-practice". In: *ACM Computing Surveys* 55.7 (2022), pp. 1–35 (cit. on p. 73).

[Ism+19] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. "Deep learning for time series classification: a review". In: *Data mining and knowledge discovery* 33.4 (2019), pp. 917–963 (cit. on p. 22).

[ISO15] ISO. *Condition monitoring and diagnostics of machines – Prognostics – Part 1: General guidelines*. Standard ISO 13381-1:2015. International Organization for Standardization, 2015 (cit. on p. 59).

[Jac00] William G Jacoby. "Loess: a nonparametric, graphical tool for depicting relationships between variables". In: *Electoral studies* 19.4 (2000), pp. 577–613 (cit. on p. 46).

[Jai+21] Nitisha Jain, Jan-Christoph Kalo, Wolf-Tilo Balke, and Ralf Krestel. "Do embeddings actually capture knowledge graph semantics?" In: *European Semantic Web Conference*. Springer. 2021, pp. 143–159 (cit. on p. 132).

[Jam+23] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. "Linear regression". In: *An introduction to statistical learning: With applications in python*. Springer, 2023, pp. 69–134 (cit. on p. 50).

[JDA+25] JDADF Developers, Jesse Averbukh, Larry Bradley, Mario Buikhuizen, Ivo Busko, Brian Cherinka, Kyle Conroy, Nicholas Earl, Ori Fox, Robel Geda, Gilbert Green, Craig Jones, Hatice Karatay, Jenn Kotler, Pey Lian Lim, Brett Morris, Duy Nguyen, Richard O'Steen, Sara Ogaz, Patrick Ogle, O. Justin Otor, Camilla Pacifici, Matthew Portman, Thomas Robitaille, Clare Shanahan, Erik Tollerud, Jason Virts, and Sabrina Volfman. *Jdaviz*. Version v4.4.2. Nov. 2025. DOI: 10.5281/zenodo.17611036 (cit. on p. 183).

[JHC+19] Krzysztof Janowicz, Armin Haller, Simon JD Cox, et al. "SOSA: A lightweight ontology for sensors, observations, samples, and actuators". In: *Journal of Web Semantics* 56 (2019), pp. 1–10 (cit. on pp. 56, 75, 174).

[Jia+22] Yuchen Jiang, Xiang Li, Hao Luo, Shen Yin, and Okyay Kaynak. "Quo vadis artificial intelligence?" In: *Discover Artificial Intelligence* 2.1 (2022), p. 4 (cit. on p. 1).

[Jia+24a] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. "Mixtral of experts". In: *arXiv preprint arXiv:2401.04088* (2024) (cit. on p. 207).

[Jia+24b] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. "A survey on large language models for code generation". In: *arXiv preprint arXiv:2406.00515* (2024) (cit. on p. 25).

[Jia+24c] Yushan Jiang, Zijie Pan, Xikun Zhang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. *Empowering Time Series Analysis with Large Language Models: A Survey*. 2024. arXiv: 2402.03182 [cs.LG]. URL: https://arxiv.org/abs/2402.03182 (cit. on pp. 202, 203).

[Jin+24] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. "Time-LLM: Time Series Forecasting by Reprogramming Large Language Models". In: *International Conference on Learning Representations (ICLR)*. 2024. arXiv: 2310.01728 (cit. on pp. 202, 209).

[JPJ24] Aryan Jadon, Avinash Patil, and Shruti Jadon. "A comprehensive survey of regression-based loss functions for time series forecasting". In: *International Conference on Data Management, Analytics & Innovation*. Springer. 2024, pp. 117–147 (cit. on p. 22).

[JPT17] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. "Time series management systems: A survey". In: *IEEE Transactions on Knowledge and Data Engineering* 29.11 (2017), pp. 2581–2600 (cit. on p. 72).

[Kal15] Rohit Kumar Kaliyar. "Graph Databases: A Survey". In: *International Conference on Computing, Communication & Automation*. IEEE, 2015, pp. 785–790 (cit. on p. 30).

[Kar+21] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440 (cit. on pp. 16, 214).

[Kar+23] Md Rezaul Karim, Md Shajalal, Alexander Graß, Till Döhmen, Sisay Adugna Chala, Alexander Boden, Christian Beecks, and Stefan Decker. "Interpreting black-box machine learning models for high dimensional datasets". In: *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*. RK developed the approach and drafted the publication, the remaining authors reviewed and revised the publication. IEEE. 2023, pp. 1–10 (cit. on p. 24).

[Kee+14] C Maria Keet, Claudia d'Amato, Zubeida Casmod Khan, and Agnieszka Ławrynowicz. "Exploring Reasoning with the DMOP Ontology". In: *Proceedings of the 3rd International Workshop on OWL Reasoner Evaluation (ORE 2014)*. Vol. 1207. CEUR Workshop Proceedings. 2014, pp. 64–70 (cit. on pp. 74, 75).

[KFE12] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. "Optimal detection of changepoints with a linear computational cost". In: *Journal of the American Statistical Association* 107.500 (2012), pp. 1590–1598 (cit. on p. 49).

[KHS+21] Shubhra Kanti Karmaker, Md Mahadi Hassan, Micah J Smith, et al. "AutoML to date and beyond: Challenges and opportunities". In: *ACM Computing Surveys (CSUR)* 54.8 (2021), pp. 1–36 (cit. on p. 193).

[KK17] Holger Knublauch and Dimitris Kontokostas. *Shapes constraint language (SHACL)*. W3C Recommendation. World Wide Web Consortium (W3C), 2017. URL: https://www.w3.org/TR/shacl/ (visited on 05/23/2024) (cit. on pp. 27, 32).

[KKS12] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. "ELK reasoner: architecture and evaluation". In: *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012)*. Ed. by Ian Horrocks, Mikalai Yatskevich, and Ernesto Jiménez-Ruiz. Vol. 858. CEUR Workshop Proceedings. CEUR-WS.org, July 2012. URL: https://ceur-ws.org/Vol-858/ (cit. on p. 32).

[Klu20] Christopher Kluse. "A critical analysis of the AIAG-VDA FMEA. Does the newly released AIAG-VDA method offer improvements over the former AIAG method?" In: *Journal of Management & Engineering Integration* 13.1 (2020), pp. 71–85. DOI: 10.62704/10057/24747 (cit. on p. 59).

[Kon+19] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. "Text generation from knowledge graphs with graph transformers". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2284–2293 (cit. on p. 133).

[Koo95] Lambert H Koopmans. *The spectral analysis of time series*. Elsevier, 1995 (cit. on p. 46).

[Kre+25] Daniel Krech, Gunnar Aastrand Grimnes, Graham Higgins, Nicholas Car, Jörn Hees, Iwan Aucamp, Niklas Lindström, Natanael Arndt, Ashley Sommer, Edmond Chuc, Ivan Herman, Alex Nelson, Jamie McCusker, Tom Gillespie, Thomas Kluyver, Florian Ludwig, Pierre-

Antoine Champin, Mark Watts, Urs Holzer, Ed Summers, Whit Morriss, Donny Winston, Drew Perttula, Filip Kovacevic, Remi Chateauneu, Harold Solbrig, Benjamin Cogrel, and Veyndan Stuart. *RDFLib*. Version 7.4.0. Oct. 2025. DOI: 10.5281/zenodo.6845245. URL: https://github.com/RDFLib/rdflib (cit. on pp. 94, 108, 119).

[Kri+19] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. "Incorporating literals into knowledge graph embeddings". In: *International Semantic Web Conference (ISWC)*. Springer, 2019, pp. 347–363 (cit. on p. 150).

[KRU15] Egor V. Kostylev, Juan L. Reutter, and Martín Ugarte. "CONSTRUCT Queries in SPARQL". In: *18th International Conference on Database Theory (ICDT 2015)*. Vol. 31. Leibniz International Proceedings in Informatics (LIPIcs). 2015, pp. 212–229. DOI: 10.4230/LIPIcs.ICDT.2015.212 (cit. on p. 176).

[KS25] Holger Knublauch and Ashley Sommer. *SHACL 1.2 SPARQL Extensions*. W3C Working Draft. W3C Working Draft 18 December 2025. World Wide Web Consortium (W3C), Dec. 2025. URL: https://www.w3.org/TR/2025/WD-shacl12-sparql-20251218/ (cit. on p. 33).

[Kum+20] Abhijeet Kumar, Abhishek Pandey, Rohit Gadia, and Mridul Mishra. "Building knowledge graph using pre-trained language model for learning entity-aware relationships". In: *2020 IEEE international conference on computing, power and communication technologies (GUCON)*. IEEE. 2020, pp. 310–315 (cit. on p. 116).

[Kut+23] Damian Kutzias, Claudia Dukino, Falko Kötter, and Holger Kett. "Comparative Analysis of Process Models for Data Science Projects". In: *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023)*. Vol. 3. SciTePress, 2023, pp. 1052–1062. DOI: 10.5220/0011895200003393 (cit. on p. 74).

[KV15] James Max Kanter and Kalyan Veeramachaneni. "Deep feature synthesis: Towards automating data science endeavors". In: *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE. 2015, pp. 1–10 (cit. on p. 21).

[Lam+16] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. "Neural architectures for named entity recognition". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 2016, pp. 260–270. arXiv: 1603.01360 (cit. on p. 115).

[Lam17] Jean-Baptiste Lamy. "Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies". In: *Artificial intelligence in medicine* 80 (2017), pp. 11–28 (cit. on pp. 109, 119).

[Law19] Sean M. Law. "STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining". In: *The Journal of Open Source Software* 4.39 (2019), p. 1504. DOI: 10.21105/joss.01504 (cit. on pp. 67, 122).

[LB22] Irina Leshcheva and Alena Begler. "A method of semi-automated ontology population from multiple semi-structured data sources". In: *Journal of information science* 48.2 (2022), pp. 223–236 (cit. on p. 114).

[LC19] Nadège Levallet and Yolande E Chan. "Organizational knowledge retention and knowledge loss". In: *Journal of Knowledge Management* 23.1 (2019), pp. 176–199 (cit. on p. 73).

[Leb+13] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. *PROV-O: The PROV Ontology*. W3C Recommendation REC-prov-o-20130430. World Wide Web Consortium (W3C), Apr. 2013. URL: https://www.w3.org/TR/prov-o/ (cit. on pp. 56, 60, 74, 78).

[Lee+18] Changhee Lee, William Zame, Jinsung Yoon, and Mihaela Van Der Schaar. "Deephit: A deep learning approach to survival analysis with competing risks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018, pp. 2314–2321 (cit. on p. 50).

[Lee+25] Geon Lee, Wenchao Yu, Kijung Shin, Wei Cheng, and Haifeng Chen. "TimeCAP: Learning to contextualize, augment, and predict time series events with large language model agents". In: *Proceedings of the AAAI*

Conference on Artificial Intelligence. Vol. 39. 17. 2025, pp. 18082–18090 (cit. on p. 203).

[Lek+20] Fritz Lekschas, Brant Peterson, Daniel Haehn, Eric Ma, Nils Gehlenborg, and Hanspeter Pfister. "Peax: Interactive visual pattern search in sequential data using unsupervised deep representation learning". In: *Computer Graphics Forum*. Vol. 39. 3. Wiley Online Library. 2020, pp. 167–179 (cit. on p. 183).

[Les06] Albert Lester. *Project management, planning and control: managing engineering, construction and manufacturing projects to PMI, APM and BSI standards*. Elsevier, 2006 (cit. on p. 59).

[Lev11] Moria Levy. "Knowledge retention: minimizing organizational business loss". In: *Journal of knowledge management* 15.4 (2011), pp. 582–600 (cit. on p. 73).

[Lew+20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 9459–9474 (cit. on p. 26).

[Li+19] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019, pp. 5243–5253 (cit. on p. 202).

[Li+21] Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace Lai-Hung Wong. "Shapenet: A shapelet-neural network approach for multivariate time series classification". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 9. 2021, pp. 8375–8383 (cit. on pp. 44, 47).

[Li+23a] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. "Towards general text embeddings with multi-stage contrastive learning". In: *arXiv preprint arXiv:2308.03281* (2023) (cit. on p. 133).

[Li+23b] Zhipeng Li, Shanshan Feng, Jun Shi, Yang Zhou, Yong Liao, Yangzhao Yang, Yangyang Li, Nenghai Yu, and Xun Shao. "Future Event Prediction Based on Temporal Knowledge Graph Embedding". In: *Computer Systems Science & Engineering* 44.3 (2023), pp. 2411–2423. DOI: 10.32604/csse.2023.026823 (cit. on p. 31).

[Li+24] Junchen Li, Yang Zhang, Kele Xu, Tao Wang, and Huaimin Wang. "Understanding the challenges of data management in the AI application development". In: *2024 IEEE International Conference on Joint Cloud Computing (JCC)*. IEEE. 2024, pp. 68–75 (cit. on p. 1).

[Lia+21] Steve Liang, Tania Khalafbeigi, Hylke van Der Schaaf, Brian Miles, Katharina Schleidt, Sylvain Grellet, Mickaël Beaufils, and Marcus Alzona. *OGC SensorThings API Part 1: Sensing Version 1.1*. OGC Implementation Standard 18-088. Open Geospatial Consortium (OGC), Aug. 2021. HAL: hal-04235851. URL: http://www.opengis.net/doc/is/sensorthings/1.1 (cit. on p. 56).

[Lia+24] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. "Foundation models for time series analysis: A tutorial and survey". In: *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*. 2024, pp. 6555–6565 (cit. on p. 25).

[Lin+07] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. "Experiencing SAX: a novel symbolic representation of time series". In: *Data Mining and knowledge discovery* 15.2 (2007), pp. 107–144 (cit. on p. 54).

[Lin+12a] Jessica Lin, Sheri Williamson, Kirk Borne, and David DeBarr. "Pattern recognition in time series". In: *Advances in Machine Learning and Data Mining for Astronomy*. Ed. by Michael J. Way, Jeffrey D. Scargle, Kamal M. Ali, and Ashok N. Srivastava. Boca Raton, FL: CRC Press, 2012, pp. 617–645 (cit. on p. 20).

[Lin+12b] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. "A shapelet transform for time series classification". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 289–297 (cit. on p. 47).

[Liu+23] Ya Liu, Yingjie Zhou, Kai Yang, and Xin Wang. "Unsupervised deep learning for IoT time series". In: *IEEE Internet of Things Journal* 10.16 (2023), pp. 14285–14306 (cit. on p. 21).

[Liu+24] Qingxiang Liu, Xu Liu, Chenghao Liu, Qingsong Wen, and Yuxuan Liang. "Time-FFM: Towards LM-Empowered Federated Foundation Model for Time Series Forecasting". In: *Advances in Neural Information Processing Systems*. Vol. 37. 2024. DOI: 10.52202/079017-2996. arXiv: 2405.14252 (cit. on p. 202).

[LKK01] Mark Last, Yaron Klein, and Abraham Kandel. "Knowledge discovery in time series databases". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 31.1 (2001), pp. 160–169 (cit. on p. 20).

[LKL14] Martin Längkvist, Lars Karlsson, and Amy Loutfi. "A review of unsupervised feature learning and deep learning for time-series modeling". In: *Pattern recognition letters* 42 (2014), pp. 11–24 (cit. on p. 21).

[LL17] Scott Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4765–4774. arXiv: 1705.07874 [cs.AI]. URL: https://arxiv.org/abs/1705.07874 (cit. on p. 25).

[Llo+14] James Lloyd, David Duvenaud, Roger Grosse, Joshua Tenenbaum, and Zoubin Ghahramani. "Automatic construction and natural-language description of nonparametric regression models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014, pp. 1242–1250 (cit. on p. 46).

[LMS14] Miodrag Lovrić, Marina Milanović, and Milan Stamenković. "Algorithmic methods for segmentation of time series: An overview". In: *Journal of Contemporary Economic and Business Issues* 1.1 (2014), pp. 31–53 (cit. on p. 19).

[LNM19] Mohamed Lubani, Shahrul Azman Mohd Noah, and Rohana Mahmud. "Ontology population: Approaches and design aspects". In: *Journal of Information Science* 45.4 (2019), pp. 502–515 (cit. on p. 108).

[Lön+19] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J Király. "sktime: A unified interface for machine learning with time series". In: *arXiv preprint arXiv:1909.07872* (2019). DOI: 10.48550/arXiv.1909.07872 (cit. on pp. 67, 94, 122, 209).

[Lu+22] Yue Lu, Renjie Wu, Abdullah Mueen, Maria A Zuluaga, and Eamonn Keogh. "Matrix profile XXIV: scaling time series anomaly detection to trillions of datapoints and ultra-fast arriving data streams". In: *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2022, pp. 1173–1182 (cit. on pp. 21, 53).

[Lub+19] Carl H. Lubba, Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. "catch22: CAnonical Time-series CHaracteristics". In: *Data Mining and Knowledge Discovery* 33.6 (2019), pp. 1821–1852. DOI: 10.1007/s10618-019-00647-x (cit. on p. 292).

[Lud+06] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao, and Yinxiao Zhao. "Scientific workflow management and the Kepler system". In: *Concurrency and Computation: Practice and Experience* 18.10 (2006), pp. 1039–1065. DOI: 10.1002/cpe.994 (cit. on p. 153).

[Luo+25] Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. "GFM-RAG: Graph Foundation Model for Retrieval Augmented Generation". In: *Advances in Neural Information Processing Systems*. Vol. 38. 2025. arXiv: 2502.01113 (cit. on pp. 133, 150).

[Mac18] Machine Learning Schema Community Group. *Machine Learning Schema*. https://www.w3.org/community/ml-schema/. Accessed: 2025-08-01. 2018 (cit. on pp. 75, 79).

[Man+24] Laura Manduchi, Clara Meister, Kushagra Pandey, Robert Bamler, Ryan Cotterell, Sina Däubener, Sophie Fellenz, Asja Fischer, Thomas Gärtner, Matthias Kirchler, et al. "On the challenges and opportunities in generative AI". In: *arXiv preprint arXiv:2403.00025* (2024) (cit. on p. 25).

[Mar19] Marquez Project Authors. *Marquez - Open Source Metadata Service*. https://github.com/MarquezProject/marquez. 2019 (cit. on p. 184).

[MB94] John McCarthy and Saša Buvač. *Formalizing Context (Expanded Notes)*. Tech. rep. CS-TN-94-13. Stanford, CA: Stanford University, 1994 (cit. on p. 37).

[McD21] Kate McDowell. "Storytelling wisdom: Story, information, and DIKW". In: *Journal of the Association for Information Science and Technology* 72.10 (2021), pp. 1223–1233 (cit. on pp. 35, 38).

[McP+15] Timothy McPhillips, Tianhong Song, Tyler Kolisnik, Steve Aulenbach, Khalid Belhajjame, Kyle Bocinsky, Yang Cao, Fernando Chirigati, Saumen Dey, Juliana Freire, Deborah Huntzinger, Christopher Jones, David Koop, Paolo Missier, Mark Schildhauer, Christopher Schwalm, Yaxing Wei, James Cheney, Mark Bieda, and Bertram Ludäscher. "YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts". In: *International Journal of Digital Curation* 10.1 (2015), pp. 298–313. arXiv: 1502.02403. URL: https://github.com/yesworkflow-org/yw-prototypes (cit. on p. 184).

[Mee+20] Wannes Meert, Kilian Hendrickx, Toon Van Craenendonck, Pieter Robberechts, Hendrik Blockeel, and Jesse Davis. *DTAIDistance*. 2020. DOI: 10.5281/zenodo.7158824. URL: https://github.com/wannesm/dtaidistance (cit. on p. 292).

[Mej+24] Nesryne Mejri, Laura Lopez-Fuentes, Kankana Roy, Pavel Chernakov, Enjie Ghorbel, and Djamila Aouada. "Unsupervised anomaly detection in time-series: An extensive evaluation and analysis of state-of-the-art methods". In: *Expert Systems with Applications* 256 (2024), p. 124922 (cit. on p. 22).

[Men+23] Qianwen Meng, Hangwei Qian, Yong Liu, Yonghui Xu, Zhiqi Shen, and Lizhen Cui. "Unsupervised representation learning for time series: A review". In: *arXiv preprint arXiv:2308.01578* (2023) (cit. on p. 21).

[MH08] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605 (cit. on p. 54).

[Mid+21] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. "HIVE-COTE 2.0: a new meta ensem-

ble for time series classification". In: *Machine Learning* 110.11 (2021), pp. 3211–3243 (cit. on pp. 44, 47).

[Mid+24] Matthew Middlehurst, Ali Ismail-Fawaz, Antoine Guillaume, Christopher Holder, David Guijo-Rubio, Guzal Bulatova, Leonidas Tsaprounis, Lukasz Mentel, Martin Walter, Patrick Schäfer, and Anthony Bagnall. "aeon: a python toolkit for learning from time series". In: *Journal of Machine Learning Research* 25.289 (2024), pp. 1–10 (cit. on p. 68).

[Mil13] Justin J. Miller. "Graph Database Applications and Concepts with Neo4j". In: *Proceedings of the Southern Association for Information Systems Conference (SAIS 2013)*. Paper 24. 2013. URL: https://aisel.aisnet.org/sais2013/24 (cit. on p. 31).

[Mil98] Eric Miller. "An Introduction to the Resource Description Framework". In: *D-Lib Magazine* 4.5 (May 1998). URL: http://www.dlib.org/dlib/may98/miller/05miller.html (cit. on p. 28).

[MIM+19] Frank Madrid, Shima Imani, Ryan Mercer, et al. "Matrix profile XX: Finding and visualizing time series motifs of all lengths using the matrix profile". In: *2019 IEEE International conference on big knowledge (ICBK)*. IEEE. 2019, pp. 175–182 (cit. on p. 76).

[Min+09] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. "Distant supervision for relation extraction without labeled data". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 2009, pp. 1003–1011 (cit. on p. 115).

[MN23] Calum Torin Mackay and David Nowell. "Informed machine learning methods for application in engineering: A review". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 237.24 (2023), pp. 5801–5818 (cit. on pp. 16, 25).

[Moh+24] Navid Mohammadi Foumani, Lynn Miller, Chang Wei Tan, Geoffrey I Webb, Germain Forestier, and Mahsa Salehi. "Deep learning for time series classification and extrinsic regression: A current survey". In: *ACM Computing Surveys* 56.9 (2024), pp. 1–45 (cit. on p. 22).

[Mon+20] Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. "FFORMA: Feature-based forecast model averaging". In: *International Journal of Forecasting* 36.1 (2020), pp. 86–92 (cit. on p. 45).

[Mor+23] John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. "Text embeddings reveal (almost) as much as text". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023, pp. 12448–12460 (cit. on p. 133).

[MP07] Alistair Miles and José R Pérez-Agüera. "SKOS: Simple knowledge organisation for the web". In: *Cataloging & Classification Quarterly* 43.3-4 (2007), pp. 69–83 (cit. on pp. 66, 79).

[MSA22] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "M5 accuracy competition: Results, findings, and conclusions". In: *International journal of forecasting* 38.4 (2022), pp. 1346–1364 (cit. on p. 2).

[Mus+24] Daham M Mustafa, Abhishek Nadgeri, Diego Collarana, Benedikt T Arnold, Christoph Quix, Christoph Lange, and Stefan Decker. "From Instructions to ODRL Usage Policies: An Ontology Guided Approach". In: *VLDB 2024 Workshop: LLM+KG*. 2024. arXiv: 2506.03301. URL: https://vldb.org/workshops/2024/proceedings/LLM+KG/LLM+KG-15.pdf (cit. on p. 116).

[Mus15] Mark A. Musen. "The protégé project: a look back and a look forward". In: *AI Matters* 1.4 (2015), pp. 4–12. DOI: 10.1145/2757001.2757003 (cit. on pp. 78, 109).

[Nae+21] Muhammad Naeem, Tauseef Jamal, Jorge Diaz-Martinez, Shariq Aziz Butt, Nicolo Montesano, Muhammad Imran Tariq, Emiro De-la-Hoz-Franco, and Ethel De-La-Hoz-Valdiris. "Trends and future perspective challenges in big data". In: *Advances in intelligent data analysis and applications: Proceeding of the sixth euro-China conference on intelligent data analysis and applications, 15–18 October 2019, Arad, Romania*. Springer. 2021, pp. 309–325 (cit. on p. 1).

[Nas+24] Nathalia Nascimento, Everton Guimaraes, Sai Sanjna Chintakunta, and Santhosh Anitha Boominathan. "Llm4ds: Evaluating large language mod-

els for data science code generation". In: *arXiv preprint arXiv:2411.11908* (2024) (cit. on p. 25).

[Nav+25] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. "A comprehensive overview of large language models". In: *ACM Transactions on Intelligent Systems and Technology* 16.5 (2025), pp. 1–72 (cit. on p. 1).

[Neo20] Neo4j Labs. *NeoDash: Neo4j Dashboard Builder*. https://neo4j.com/labs/neodash/. 2020 (cit. on p. 184).

[Nic+16] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A Review of Relational Machine Learning for Knowledge Graphs". In: *Proceedings of the IEEE* 104.1 (2016), pp. 11–33. DOI: 10.1109/JPROC.2015.2483592 (cit. on pp. 134, 174).

[Nie+23] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. "A time series is worth 64 words: Long-term forecasting with transformers". In: *International Conference on Learning Representations (ICLR)*. 2023. arXiv: 2211.14730 (cit. on p. 202).

[Nie19] Aileen Nielsen. *Practical time series analysis: Prediction with statistics and machine learning*. O'Reilly Media, 2019 (cit. on p. 21).

[NM01] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. Tech. rep. KSL-01-05. Stanford, CA, USA: Stanford Knowledge Systems Laboratory, 2001. URL: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf (cit. on p. 78).

[Nor+24] Sanaz Saki Norouzi, Adrita Barua, Antrea Christou, Nikita Gautam, Andrew Eells, Pascal Hitzler, and Cogan Shimizu. "Ontology population using LLMs". In: *arXiv preprint arXiv:2411.01612* (2024) (cit. on p. 114).

[NR07] Vit Niennattrakul and Chotirat Ann Ratanamahatana. "On clustering multimedia time series data using k-means and dynamic time warping". In: *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*. IEEE. 2007, pp. 733–738 (cit. on p. 52).

[NVM13] Robert C Nickerson, Upkar Varshney, and Jan Muntermann. "A method for taxonomy development and its application in information systems". In: *European journal of information systems* 22.3 (2013), pp. 336–359 (cit. on pp. 61, 62, 67).

[Nwi+23] Chike Nwibor, Shyqyri Haxha, Mian Mujtaba Ali, Mohamed Sakel, Anda Rexha Haxha, Karen Saunders, and Shakira Nabakooza. "Remote health monitoring system for the estimation of blood pressure, heart rate, and blood oxygen saturation level". In: *IEEE Sensors Journal* 23.5 (2023), pp. 5401–5411 (cit. on p. 17).

[OAS10] OASIS Emergency Management Technical Committee. *Common Alerting Protocol Version 1.2 (CAP v1.2)*. Ed. by Jacob Westfall. OASIS Standard. July 2010. URL: https://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html (cit. on p. 58).

[Obj11] Object Management Group (OMG). *Business Process Model and Notation (BPMN) Version 2.0*. Tech. rep. OMG document number: formal/2011-01-04. Object Management Group (OMG), Jan. 2011. URL: https://www.omg.org/spec/BPMN/2.0/ (cit. on p. 57).

[OE18] Achraf Oussidi and Azeddine Elhassouny. "Deep generative models: Survey". In: *2018 International conference on intelligent systems and computer vision (ISCV)*. IEEE. 2018, pp. 1–8 (cit. on p. 23).

[Ont24] Ontotext. *GraphDB Documentation*. Accessed: 2025. 2024. URL: https://graphdb.ontotext.com/ (cit. on pp. 31, 184).

[OOO25] Oluwafemi Oloruntoba, Deborah Olamide Oyeyemi, and Olasehinde Omolayo. "Designing Scalable ETL Pipelines for Multi-Source Graph Database Ingestion". In: *Journal of Computational Analysis & Applications* 34.7 (2025) (cit. on p. 114).

[OPC22] OPC Foundation. *OPC Unified Architecture – Part 9: Alarms and Conditions*. Standard OPC 10000-9. OPC Foundation, 2022. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture (cit. on p. 58).

[Ope23] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL] (cit. on p. 202).

[Ope24a] OpenAI. *GPT-4o System Card*. 2024. arXiv: 2410.21276 (cit. on p. 207).

[Ope24b] OpenAI. *OpenAI GPT Models*. Accessed on 2025-11-22. 2024. URL: https://platform.openai.com/docs/models (cit. on p. 121).

[Ope25] OpenAI. *gpt-oss-120b & gpt-oss-20b Model Card*. 2025. arXiv: 2508.10925 (cit. on p. 207).

[Pau17] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semantic web* 8.3 (2017), pp. 489–508 (cit. on pp. 174, 179).

[Pea01] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572 (cit. on p. 53).

[Pen+23] Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. "Knowledge graphs: Opportunities and challenges". In: *Artificial intelligence review* 56.11 (2023), pp. 13071–13102 (cit. on p. 16).

[Pen+24] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. "Graph retrieval-augmented generation: A survey". In: *ACM Transactions on Information Systems* (2024). DOI: 10.1145/3777378 (cit. on pp. 132, 150).

[PGP+21] Francesco Piccialli, Fabio Giampaolo, Edoardo Prezioso, et al. "Artificial intelligence and healthcare: Forecasting of medical bookings through multi-source time-series fusion". In: *Information Fusion* 74 (2021), pp. 1–16 (cit. on p. 72).

[PGS14] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. "OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation". In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 10.2 (2014), pp. 7–34 (cit. on pp. 78, 94).

[PJ21] Archana Patel and Sarika Jain. "Present and future of semantic web technologies: a research statement". In: *International Journal of Computers and Applications* 43.5 (2021), pp. 413–422 (cit. on p. 16).

[PKK24] Or Peretz, Michal Koren, and Oded Koren. "Naive Bayes classifier–An ensemble procedure for recall and precision enrichment". In: *Engineering Applications of Artificial Intelligence* 136 (2024), p. 108972 (cit. on p. 23).

[Plo17] Plotly Technologies Inc. *Dash Layout*. https://dash.plotly.com/layout. 2017 (cit. on p. 186).

[Pov+22] María Poveda-Villalón, Alba Fernández-Izquierdo, Mariano Fernández-López, and Raúl García-Castro. "LOT: An industrial oriented ontology engineering framework". In: *Engineering Applications of Artificial Intelligence* 111 (2022), p. 104755 (cit. on p. 78).

[PSD13] Panče Panov, Larisa Soldatova, and Sašo Džeroski. "OntoDM-KDD: ontology for representing the knowledge discovery process". In: *Discovery Science: 16th International Conference, DS 2013, Singapore, October 6-9, 2013. Proceedings 16*. Springer. 2013, pp. 126–140 (cit. on pp. 71, 74, 75).

[Pub+18] Gustavo Correa Publio, Diego Esteves, Agnieszka Ławrynowicz, Panče Panov, Larisa Soldatova, Tommaso Soru, Joaquin Vanschoren, and Hamid Zafar. *ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies*. 2018. arXiv: 1807.05351. URL: https://arxiv.org/abs/1807.05351 (cit. on pp. 71, 75).

[Qin+23] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. "Tool learning with foundation models". In: *arXiv preprint arXiv:2304.08354* (2023) (cit. on p. 203).

[QUD11] QUDT Organization. *QUDT; Quantities, Units, Dimensions and Types*. https://fairsharing.org/10.25504/FAIRsharing.d3pqw7. 2011 (cit. on p. 79).

[Rad+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. *Language models are unsupervised multitask learners*. Tech. rep. OpenAI, 2019 (cit. on p. 202).

[Ray25] Partha Pratim Ray. "A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions". In: *Authorea Preprints* (2025) (cit. on p. 117).

[RD00] Erhard Rahm and Hong Hai Do. "Data cleaning: Problems and current approaches". In: *IEEE Data Eng. Bull.* 23.4 (2000), pp. 3–13 (cit. on p. 42).

[RG19] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence embeddings using siamese BERT-networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3982–3992. arXiv: 1908.10084 (cit. on p. 133).

[RGP08] Pedro Pereira Rodrigues, Joao Gama, and Joao Pedroso. "Hierarchical clustering of time-series data streams". In: *IEEE transactions on knowledge and data engineering* 20.5 (2008), pp. 615–627 (cit. on p. 52).

[RH22] Joe Reis and Matt Housley. *Fundamentals of data engineering*. O'Reilly Media, Inc., 2022 (cit. on p. 20).

[Roj+21] Thomas Rojat, Raphaël Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Díaz-Rodríguez. "Explainable artificial intelligence (XAI) on timeseries data: A survey". In: *arXiv preprint arXiv:2104.00950* (2021) (cit. on p. 25).

[Ros+21] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. "Knowledge graph embedding for link prediction: A comparative analysis". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15.2 (2021), pp. 1–49 (cit. on pp. 31, 132).

[Ros+86] A. Rosenthal, S. Heiler, U. Dayal, and F. Manola. "Traversal Recursion: A Practical Approach to Supporting Recursive Applications". In: *ACM SIGMOD Record* 15.2 (1986), pp. 166–176 (cit. on p. 32).

[Row07] Jennifer Rowley. "The wisdom hierarchy: representations of the DIKW hierarchy". In: *Journal of information science* 33.2 (2007), pp. 163–180 (cit. on pp. 35, 38).

[RP11] Yuan Ren and Jeff Z Pan. "Optimising ontology stream reasoning with truth maintenance system". In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. 2011, pp. 831–836 (cit. on p. 35).

[RP16] Petar Ristoski and Heiko Paulheim. "RDF2Vec: RDF graph embeddings for data mining". In: *International semantic web conference*. Springer. 2016, pp. 498–514 (cit. on p. 31).

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "'Why should I trust you?': Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144 (cit. on p. 25).

[Rub02] Albert Rubio. "A Fully Syntactic AC-RPO". In: *Information and Computation* 178.2 (2002), pp. 515–533. DOI: 10.1006/inco.2002.3158 (cit. on pp. 330, 331).

[Rue+22] Laura von Rueden, Sebastian Houben, Kostadin Cvejoski, Christian Bauckhage, and Nico Piatkowski. "Informed pre-training on prior knowledge". In: *arXiv preprint arXiv:2205.11433* (2022) (cit. on pp. 25, 214).

[Run18] Jakob Runge. "Causal network reconstruction from time series: From theoretical assumptions to practical estimation". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.7 (2018), p. 075310. DOI: 10.1063/1.5025050 (cit. on p. 67).

[Run20] Jakob Runge. "Discovering contemporaneous and lagged causal relations in autocorrelated nonlinear time series datasets". In: *Conference on uncertainty in artificial intelligence*. PMLR. 2020, pp. 1388–1397 (cit. on p. 48).

[SAL+19] Renan Souza, Leonardo Azevedo, Vítor Lourenço, et al. "Provenance data in the machine learning lifecycle in computational science and engineering". In: *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. IEEE. 2019, pp. 1–10 (cit. on p. 75).

[Sap+25] Patrick Sapel, Lina Molinas Comet, Iraklis Dimitriadis, Christian Hopmann, and Stefan Decker. "A review and classification of manufacturing ontologies". In: *Journal of Intelligent Manufacturing* 36.6 (2025), pp. 3669–3693. DOI: 10.1007/s10845-024-02425-z (cit. on p. 55).

[Sar21] Iqbal H Sarker. "Data science and analytics: an overview from data-driven smart computing, decision-making and applications perspective". In: *SN Computer Science* 2.5 (2021), p. 377 (cit. on pp. 1, 16).

[SC15] Barry Smith and Werner Ceusters. "Aboutness: Towards Foundations for the Information Artifact Ontology". In: *Proceedings of the International Conference on Biomedical Ontology (ICBO)*. Vol. 1515. CEUR Workshop Proceedings. 2015, pp. 47–51 (cit. on p. 60).

[SC24] Ashley Sommer and Nicholas Car. *pySHACL*. Version v0.26.0. 2024. DOI: 10.5281/zenodo.10958008 (cit. on pp. 111, 121).

[Sch+23] Jona Scholz, Maike Holtkemper, Alexander Graß, and Christian Beecks. "Anomaly Detection in Manufacturing". In: *Artificial Intelligence in Manufacturing: Enabling Intelligent, Flexible and Cost-Effective Production Through AI*. JS drafted the publication. JS and AG developed and implemented the presented methodology, MH supported with writing the technical sections. CB supported with writing the general sections and reviewed and revised the publication. Springer Nature Switzerland Cham, 2023, pp. 351–360 (cit. on pp. 2, 106).

[Sch00] Thomas Schreiber. "Measuring Information Transfer". In: *Physical Review Letters* 85.2 (2000), pp. 461–464 (cit. on p. 48).

[SCY17] Robert Sanderson, Paolo Ciccarese, and Benjamin Young. *Web Annotation Data Model*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 2017. URL: https://www.w3.org/TR/2017/REC-annotation-model-20170223/ (cit. on p. 56).

[SD84] Said E. Said and David A. Dickey. "Testing for unit roots in autoregressive-moving average models of unknown order". In: *Biometrika* 71.3 (1984), pp. 599–607. DOI: 10.1093/biomet/71.3.599 (cit. on p. 48).

[SEL21] Patrick Schäfer, Arik Ermshaus, and Ulf Leser. "Clasp-time series segmentation". In: *Proceedings of the 30th ACM international conference on information & knowledge management*. 2021, pp. 1578–1587 (cit. on p. 19).

[SF22] Ali Shojaie and Emily B Fox. "Granger causality: A review and recent advances". In: *Annual Review of Statistics and Its Application* 9.1 (2022), pp. 289–319 (cit. on p. 48).

[SGF11] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. "The NeOn methodology for ontology engineering". In: *Ontology engineering in a networked world*. Springer, 2011, pp. 9–34 (cit. on p. 78).

[SGO20] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019". In: *Applied soft computing* 90 (2020), p. 106181 (cit. on p. 72).

[Shc+23] Oleksandr Shchur, Ali Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Bernie Wang. "AutoGluon–TimeSeries: AutoML for probabilistic time series forecasting". In: *International Conference on Automated Machine Learning*. PMLR. 2023, pp. 1–9 (cit. on p. 16).

[She+23] Kevin Sheppard et al. *bashtage/arch: Release v5.5.0*. 2023. DOI: 10.5281/zenodo.8338721. URL: https://github.com/bashtage/arch (cit. on p. 292).

[Sho+15] Mohammad Shokoohi-Yekta, Yanping Chen, Bilson Campana, Bing Hu, Jesin Zakaria, and Eamonn Keogh. "Discovery of meaningful rules in time series". In: *Proceedings of the 21st ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1085–1094 (cit. on p. 20).

[Shu24] Jiwu Shu. "Distributed storage systems". In: *Data Storage Architectures and Technologies*. Springer, 2024, pp. 185–224 (cit. on p. 1).

[Sir+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. "Pellet: A practical OWL-DL reasoner". In: *Journal of Web Semantics* 5.2 (2007), pp. 51–53 (cit. on p. 32).

[SKG21] Christoph Schröer, Felix Kruse, and Jorge Marx Gómez. "A systematic literature review on applying CRISP-DM process model". In: *Procedia Computer Science* 181 (2021), pp. 526–534 (cit. on p. 20).

[Sol24] John Soldatos, ed. *Artificial Intelligence in Manufacturing: Enabling Intelligent, Flexible and Cost-Effective Production Through AI*. Springer Nature, 2024 (cit. on p. 76).

[SP10] Skipper Seabold and Josef Perktold. "statsmodels: Econometric and statistical modeling with python". In: *Proceedings of the 9th Python in Science Conference*. 2010, pp. 92–96 (cit. on p. 67).

[SP17] James M. Snell and Evan Prodromou. *Activity Streams 2.0*. W3C Recommendation. World Wide Web Consortium (W3C), May 2017. URL: https://www.w3.org/TR/activitystreams-core/ (cit. on p. 57).

[Spi+11] Stephan Spiegel, Julia Gaebler, Andreas Lommatzsch, Ernesto De Luca, and Sahin Albayrak. "Pattern recognition and classification for multivariate time series". In: *Proceedings of the fifth international workshop on knowledge discovery from sensor data*. 2011, pp. 34–42 (cit. on p. 20).

[SS17] Robert H Shumway and David S Stoffer. "ARIMA models". In: *Time series analysis and its applications: with R examples*. Springer, 2017, pp. 75–163 (cit. on p. 51).

[SS21] Radhey Shyam and Ria Singh. "A taxonomy of machine learning techniques". In: *Journal of Advancements in Robotics* 8.3 (2021), pp. 18–25 (cit. on p. 193).

[ST12] Donald Sannella and Andrzej Tarlecki. *Foundations of algebraic specification and formal software development*. Springer, 2012 (cit. on p. 152).

[Sta18] Stardog Union. *Stardog Studio*. https://www.stardog.com/studio/. 2018 (cit. on p. 184).

[STK22] Payal Soni, Yogya Tewari, and Deepa Krishnan. "Machine learning approaches in stock price prediction: a systematic review". In: *Journal of Physics: Conference Series* 2161.1 (2022), p. 012065 (cit. on p. 17).

[STN19] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. "The performance of LSTM and BiLSTM in forecasting time series". In: *2019 IEEE International conference on big data (Big Data)*. IEEE. 2019, pp. 3285–3292 (cit. on p. 51).

[Sun+19] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. "RotatE: Knowledge graph embedding by relational rotation in complex space". In: *International Conference on Learning Representations (ICLR)*. 2019. arXiv: 1902.10197 (cit. on pp. 133, 135).

[Sun+24] Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. *A Survey on Large Language Model-based Agents for Statistics and Data Science*. 2024. arXiv: 2412.14222 [cs.AI]. URL: https://arxiv.org/abs/2412.14222 (cit. on p. 203).

[Sur+12] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. "Multi-instance multi-label learning for relation extraction". In: *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. 2012, pp. 455–465 (cit. on p. 115).

[SWP22] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. "Anomaly detection in time series: a comprehensive evaluation". In: *Proceedings of the VLDB Endowment* 15.9 (2022), pp. 1779–1797 (cit. on pp. 49, 76).

[Tav+20] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. "Tslearn, A Machine Learning Toolkit for Time Series Data". In: *Journal of Machine Learning Research* 21.118 (2020), pp. 1–6 (cit. on pp. 68, 122).

[The24] Johannes Theissen-Lipp. "Semantic Foundations of Dataspaces". Dissertation. RWTH Aachen University, 2024. DOI: 10.18154/RWTH-2024-09760 (cit. on p. 78).

[Tho+04] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. *XML Schema Part 1: Structures Second Edition*. W3C Recommendation REC-xmlschema-1-20041028. World Wide Web Consortium, 2004. URL: https://www.w3.org/TR/xmlschema-1/ (cit. on p. 28).

[Tho+19] Katherine Thornton, Harold Solbrig, Gregory S. Stupp, Jose Emilio Labra Gayo, Daniel Mietchen, Eric Prud'hommeaux, and Andra Waagmeester. "Using shape expressions (ShEx) to share RDF data models and to guide curation with rigorous validation". In: *European Semantic Web Conference*. Cham: Springer International Publishing, May 2019, pp. 606–620 (cit. on p. 33).

[Tka+20] Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. *Label Studio: Data labeling software*. Open source software

available from https://github.com/HumanSignal/label-studio. 2020. URL: https://github.com/HumanSignal/label-studio (cit. on p. 184).

[TL18] Sean J Taylor and Benjamin Letham. "Forecasting at scale". In: *The American Statistician* 72.1 (2018), pp. 37–45. DOI: 10.1080/00031305.2017.1380080 (cit. on pp. 209, 292).

[Tou+23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. "LLaMA: Open and Efficient Foundation Language Models". In: *CoRR* abs/2302.13971 (2023). DOI: 10.48550/ARXIV.2302.13971. arXiv: 2302.13971 (cit. on p. 202).

[Tou03] Stephen E. Toulmin. *The Uses of Argument*. Updated. Cambridge University Press, 2003. ISBN: 978-0521534833 (cit. on p. 60).

[TOV20] Charles Truong, Laurent Oudre, and Nicolas Vayatis. "Selective review of offline change point detection methods". In: *Signal Processing* 167 (2020), p. 107299 (cit. on pp. 122, 292).

[Tri+21] Oskar Triebe, Hansika Hewamalage, Polina Pilyugina, Nikolay Laptev, Christoph Bergmeir, and Ram Rajagopal. "NeuralProphet: Explainable Forecasting at Scale". In: *arXiv preprint arXiv:2111.15397* (2021) (cit. on p. 292).

[Van+17] Pierre-Yves Vandenbussche, Ghislain A Atemezing, María Poveda-Villalón, and Bernard Vatant. "Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web". In: *Semantic Web* 8.3 (2017), pp. 437–452 (cit. on p. 93).

[VF05] Michel Verleysen and Damien François. "The curse of dimensionality in data mining and time series prediction". In: *International work-conference on artificial neural networks*. Springer. 2005, pp. 758–770 (cit. on p. 24).

[VH20] Jesper E Van Engelen and Holger H Hoos. "A survey on semi-supervised learning". In: *Machine learning* 109.2 (2020), pp. 373–440 (cit. on p. 23).

[Von+23] Laura Von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. "Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems". In: *IEEE Transactions on Knowledge and Data Engineering* 35.1 (2023), pp. 614–633. DOI: 10.1109/TKDE.2021.3079836 (cit. on pp. 16, 72, 193, 195, 214).

[VS10] Joaquin Vanschoren and Larisa Soldatova. "Exposé: An ontology for data mining experiments". In: *International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010)*. 2010, pp. 31–46 (cit. on pp. 71, 74, 75).

[W3C12] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation REC-owl2-overview-20121211. World Wide Web Consortium (W3C), Dec. 2012. URL: https://www.w3.org/TR/owl2-overview/ (cit. on p. 28).

[Wan+23] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and Philip S Yu. "A survey on heterogeneous graph embedding: methods, techniques, applications and sources". In: *IEEE transactions on big data* 9.2 (2023), pp. 415–436 (cit. on pp. 132, 134).

[Wan+24] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Chen Wang, Mingsheng Long, and Jianmin Wang. "Deep time series models: A comprehensive survey and benchmark". In: *arXiv preprint arXiv:2407.13278* (2024) (cit. on p. 41).

[Wen+23] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. "Transformers in time series: A survey". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23), Survey Track*. 2023, pp. 6778–6786. DOI: 10.24963/ijcai.2023/759. arXiv: 2202.07125 (cit. on p. 25).

[Wil+16] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific data* 3.1 (2016), pp. 1–9 (cit. on p. 27).

[Woo+24] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. "Unified training of universal time series forecasting transformers". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. 2024, pp. 53140–53164. arXiv: 2402.02592 (cit. on p. 202).

[Wor12] World Wide Web Consortium (W3C). *R2RML: RDB to RDF Mapping Language*. W3C Recommendation. World Wide Web Consortium, Sept. 2012. URL: https://www.w3.org/TR/r2rml/ (cit. on pp. 109, 115).

[Wu+21] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22419–22430 (cit. on p. 202).

[Wu+22] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. "Graph neural networks in recommender systems: a survey". In: *ACM Computing Surveys* 55.5 (2022), pp. 1–37 (cit. on p. 30).

[Wu+24] Xinle Wu, Xingjian Wu, Bin Yang, Lekui Zhou, Chenjuan Guo, Xiangfei Qiu, Jilin Hu, Zhenli Sheng, and Christian S Jensen. "AutoCTS++: zero-shot joint neural architecture and hyperparameter search for correlated time series forecasting". In: *The VLDB Journal* 33.5 (2024), pp. 1743–1770 (cit. on p. 25).

[WW20] Xi Wang and Chen Wang. "Time series data cleaning: A survey". In: *IEEE Access* 8 (2020), pp. 1866–1881 (cit. on p. 18).

[WYZ+22] Qingsong Wen, Linxiao Yang, Tian Zhou, et al. "Robust time series analysis and applications: An industrial perspective". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 4836–4837 (cit. on p. 72).

[Xia+23] Qiao Xiao, Khuan Lee, Siti Aisah Mokhtar, Iskasymar Ismail, Ahmad Luqman bin Md Pauzi, Qiuxia Zhang, and Poh Ying Lim. "Deep learning-based ECG arrhythmia classification: A systematic review". In: *Applied Sciences* 13.8 (2023), p. 4964 (cit. on p. 53).

[XS24] Hao Xue and Flora D. Salim. "PromptCast: A New Prompt-Based Learning Paradigm for Time Series Forecasting". In: *IEEE Trans. Knowl. Data Eng.* 36.11 (2024), pp. 6851–6864. DOI: 10.1109/TKDE.2023.3342137 (cit. on p. 203).

[Xu+15] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. "Semantic relation classification via convolutional neural networks with simple negative sampling". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015, pp. 536–540. arXiv: 1506.07650 (cit. on p. 115).

[Xu+22] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. "Anomaly transformer: Time series anomaly detection with association discrepancy". In: *International Conference on Learning Representations (ICLR)*. 2022. arXiv: 2110.02642 (cit. on p. 49).

[Yan+15] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding entities and relations for learning and inference in knowledge bases". In: *International Conference on Learning Representations (ICLR)*. 2015. arXiv: 1412.6575 (cit. on pp. 133, 135).

[Yao+23] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. "ReAct: Synergizing Reasoning and Acting in Language Models". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. 2023 (cit. on p. 203).

[YHB24] Jameel Shehu Yalli, Mohd Hilmi Hasan, and Aisha Abubakar Badawi. "Internet of Things (IoT): origins, embedded technologies, smart applications, and its growth in the last decade". In: *IEEE Access* 12 (2024), pp. 91357–91382 (cit. on p. 1).

[Yin+20] Yuanxiang Ying, Juanyong Duan, Chunlei Wang, Yujing Wang, Congrui Huang, and Bixiong Xu. "Automated model selection for time-series anomaly detection". In: *arXiv preprint arXiv:2009.04395* (2020) (cit. on pp. 25, 193).

[YK09] Lexiang Ye and Eamonn Keogh. "Time Series Shapelets: A New Primitive for Data Mining". In: *Proceedings of the 15th ACM SIGKDD In-*

ternational Conference on Knowledge Discovery and Data Mining. 2009, pp. 947–956 (cit. on p. 47).

[YZU+16] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, et al. "Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets". In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE. 2016, pp. 1317–1322 (cit. on p. 53).

[Zah+18] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. "Accelerating the machine learning lifecycle with MLflow." In: *IEEE Data Eng. Bull.* 41.4 (2018), pp. 39–45 (cit. on pp. 3, 100).

[Zak01] Mohammed J Zaki. "SPADE: An efficient algorithm for mining frequent sequences". In: *Machine learning* 42.1 (2001), pp. 31–60 (cit. on p. 53).

[Zer+21] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. "A transformer-based framework for multivariate time series representation learning". In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 2114–2124 (cit. on p. 21).

[Zha+19] Qin Zhang, Jia Wu, Peng Zhang, Guodong Long, and Chengqi Zhang. "Salient subsequence learning for time series clustering". In: *IEEE transactions on pattern analysis and machine intelligence* 41.9 (2019), pp. 2193–2207 (cit. on pp. 21, 52).

[Zha+23a] Yanwei Zhai, Zheng Lv, Jun Zhao, and Wei Wang. "Knowledge discovery and variable scale evaluation for long series data". In: *Artificial Intelligence Review* 56.4 (2023), pp. 3157–3180 (cit. on p. 193).

[Zha+23b] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* (2023) (cit. on p. 25).

[Zha19] Dengsheng Zhang. "Wavelet transform". In: *Fundamentals of image data mining: Analysis, Features, Classification and Retrieval*. Springer, 2019, pp. 35–44 (cit. on p. 54).

[Zho+21] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. "Informer: Beyond efficient transformer for long sequence time-series forecasting". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 11106–11115 (cit. on p. 202).

[Zho+22] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting". In: *International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR. 2022, pp. 27268–27286 (cit. on p. 202).

[Zhu+21] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. "Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021. arXiv: 2106.06935 (cit. on p. 150).

[Zhu+23] Zhaocheng Zhu, Xinyu Yuan, Mikhail Galkin, Sophie Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. "A*Net: A Scalable Path-based Reasoning Approach for Knowledge Graphs". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 59323–59336. arXiv: 2206.04798 (cit. on p. 150).

[Zin07] Chaim Zins. "Conceptual approaches for defining data, information, and knowledge". In: *Journal of the American society for information science and technology* 58.4 (2007), pp. 479–493 (cit. on p. 35).

[ZNL19] Yue Zhao, Zain Nasrullah, and Zheng Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html (cit. on p. 292).

[ZSG+21] Baifan Zhou, Yulia Svetashova, Andre Gusmao, et al. "SemML: Facilitating development of ML models for condition monitoring with semantics". In: *Journal of Web Semantics* 71 (2021), p. 100664 (cit. on p. 75).

[ZZZ+22] Dongzhuoran Zhou, Baifan Zhou, Zhuoxun Zheng, et al. "Ontology reshaping for knowledge graph construction: applied on Bosch welding case". In: *International Semantic Web Conference*. Springer. 2022, pp. 770–790 (cit. on p. 75).

List of Figures

1.1 Comparison of an independent, linear analytical workflow (Approach A) against a cyclically informed workflow (Approach B). . 2

1.2 Concept of an incrementally enriched *Time Series Knowledge Graph* serving as common, standardized and evolving knowledge resource and semantic index for various consumers and processes. Gathered knowledge is iteratively used in subsequent queries and analyses. 5

1.3 Layered overview of the thesis contributions, organized by increasing abstraction from knowledge classification at the base to application-level exploitation at the top. Each layer is annotated with the corresponding research question it primarily addresses. . 15

2.1 Illustration of a univariate time series and an exemplary representation of two segments defining a time range and a single point in time. In case of multivariate time series data, a segment can also define a (sub-)space consisting of more than one variable. 19

2.2 Unsupervised vs. supervised anomaly detection: In the unsupervised setting, anomalies are detected using only unlabeled data; for example, by comparing individual segments as indicated by dotted lines. In the supervised setting, labeled examples of anomalies are available, turning anomaly detection into a classification problem by learning to distinguish normal from anomalous segments. . . . 26

2.3 An adapted representation of the Semantic Web technology stack, emphasizing technologies and standards employed in this work including concrete examples. Layers are ordered by increasing abstraction, ranging from identifiers and serialization formats at the base to application-level integration at the top. 28

3.1 Two different models describing the value of processed data: DIKW [Row07; McD21] model with discrete, unidirectional progression (left side) and our customized model with continuous, bidirectional progression (right side). 38

3.2 Examples of general dataset properties (Section 3.2) and time series characteristics (Section 3.3). 45

3.3 Causal Effects 49

3.4 Anomalies / Discords 49

3.5 Predicted Event 50

3.6 Forecast 50

3.7 (Dis-)Similarity 52

3.8 Subsequence Clustering 52

3.9 Pattern / Motif 54

3.10 Discrete Projection 54

3.11 Major categories of the two knowledge taxonomies. Both, Data Knowledge and Scenario Knowledge are presented with a hierarchical depth of 2 levels. 65

4.1 Impact of knowledge reuse: Example of two Fourier regression models, representing outcomes with knowledge integration (blue) and without it (green). Excluding the anomaly reduces the Mean Squared Error (MSE) by 58% from 1.60 to 0.67. Both, the time series and the anomaly represent knowledge segments. 73

4.2 Overview of SemTS organized by its three conceptual layers: Data Management (green) handles segment structure and value representation, Knowledge Classification (blue) provides the coarse taxonomy of insight types, and Knowledge Generation (red) captures provenance and derivation processes. External schemas are indicated by namespace prefixes and dotted borders; classes rooted in SKOS appear with rounded corners to signal their role as extensible concept anchors. The image shows an updated representation of the architecture diagram from [Gra+26] 82

4.3 A simplified example showing how a pattern discovery process is defined by the three SemTS layers. A single input segment yields two new segments (linked via `isPartOf`) whose associated knowledge items demonstrate the distinction between per-segment properties (`segmentKnowledge`) and relational properties spanning multiple segments (`multiSegmentKnowledge`). Vocabulary concepts annotate each knowledge instance to provide machine-interpretable typing. 86

4.4 Further visually simplified knowledge generation patterns based on SemTS. 87

4.5 Example for the hierarchical mapping of particular knowledge concepts from the data knowledge vocabulary (`semts-dv`) to nested knowledge groups. Concepts extending `skos:Concept` are shown in rounded boxes, while those also categorized as `skos:Collection` are additionally emphasized by bold borders. 89

5.1 Scenario 1: Description of the use of knowledge groups and different value classes in SemTS. 104

5.2 Scenario 2: SemTS representation focusing on the synergy of Data Knowledge and Scenario Knowledge. Furthermore it illustrates the use of artificial time series segments, describing generated time series data. 105

5.3 Scenario 3: Visualization of knowledge propagation via SemTS in a concrete use case. 107

5.4 Simplified illustration of Code2Onto's architecture and interactions based on [Gra+25b]. 118

5.5 Simple code example of a mean computation through the tsfresh library. The code was mapped to SemTS via Code2Onto. 120

5.6 Evaluation results for Code2Onto across three LLMs [Gra+25b]. . 125

5.7 Examples of categories of time series characteristics aligned with the Data Knowledge vocabulary. Such categories can, for instance, be used to semantically group individual characteristics for feature engineering, enabling the construction of feature vectors with a consistent, predefined meaning. 128

5.8 Link prediction performance across KGE models. Left: MRR with standard deviation bars. Right: Hits@k for $k \in \{1, 3, 10\}$. 141

5.9 Retrieval performance (adjusted precision@10) across all tasks with 95% CI (mean $\sigma = 0.024$). 143

5.10 Average adjusted precision@10 across all 13 tasks per embedding method. 145

5.11 Example of incremental knowledge propagation [Gra+23]: (top left) segmentation, (top right) feature extraction from segments, (bottom left) feature-based clustering of segments, (bottom right) anomaly detection in a single segment. 152

5.12 Contextual reasoning in SemTS via SPARQL CONSTRUCT . . . 178

6.1 SemTS Explorer - *Explore* page 185

6.2 SemTS Explorer - *Query* page 186

6.3 SemTS Explorer - *Author & Validate* page 187

6.4 SemTS Explorer - *LLM Report* page 189

6.5 SemTS Explorer - Workflow demo: An identified anomaly is re-classified as *false positive* since the anomalous behavior can be explained by a change in operating mode. The final LLM report updates the intermediate report with full provenance including the anomaly interpretation. 190

6.6 Object diagram representing the grammar from Listing 6.1 as visual description of time series hierarchies. Arrows represent individual rules, while compositions are defined by diamond symbols. Non-terminal symbols are colored in blue, non-terminal list symbols in lightblue and terminal symbols in green. 196
6.7 Simplified visualization of a concrete data setting. 199
6.8 Hierarchical architecture of the agent system interacting with the incrementally enriched, unified knowledge graph. Red arrows indicate the workflow and data exchange. 204
6.9 Application scenario on the example of a forecast. The identified anomaly from a previous analysis is considered during the forecast with new knowledge being reintegrated. 206
6.10 SAGE Evaluation. Comparison of different metrics across models for zero-shot classification, Graph RAG information checks, and factual correctness tests. 208
6.11 SAGE Evaluation. Baseline evaluation of all forecasting estimators, ranked by normalized position. Lower values correspond to higher positions and better results. 210
6.12 SAGE Evaluation. Performance analysis of the top four LLMs on successful tests. 211

List of Tables

3.1 Categories of time series analysis approaches, represented by associated knowledge output examples. 42
3.2 Alignment between our taxonomy-development procedure and the method of Nickerson et al. [NVM13] 62

4.1 Coverage of approaches from the related work section. 75
4.2 Evaluation of SemTS through competency questions. Example taken from [Gra+26]. 95
4.3 Competency Questions and Examples 96

5.1 Analytical tasks used for evaluation [Gra+25b], covering diverse analysis types and libraries to avoid concentrating on a narrow ontological subset. 122
5.2 Overview of retrieval tasks organized by structural category. Each task tests whether embeddings can recover a specific ontology-encoded relationship. The "Ont. Dist." column indicates the minimum path length between query and relevant segments in the base ontology (without enrichment shortcuts). Relation-specific KGE scoring operates via the corresponding materialized enrichment relation. 138
5.3 Link prediction results (mean ± std across all runs). Best values per column in bold. 140
5.4 Retrieval evaluation results (adjusted precision@10, mean across all runs) on the *full* graph variant. Best result per task in bold. KGE, Node2Vec, and text families are shown, with hybrid fusion reported separately in Table 5.6. 142

5.5 Shortcut ablation: adjusted precision@10 on the *full* graph (with `enrich:` shortcuts) versus the *no-shortcuts* variant (all enrichment relations removed), averaged across the 12 standard retrieval tasks. KGE models collapse to near-random when shortcuts are removed, whereas the text methods are robust by construction. 145

5.6 Score-level hybrid fusion (KGE + Text+, $\alpha = 0.5$) versus the corresponding KGE base model, mean adjusted precision@10 across the 12 standard tasks, plus selected mixed-criteria tasks. Equal-weight fusion helps only the weakest base model and on multi-criteria tasks, while degrading the strongest models overall. 146

5.7 Structure advantage comparing the best structured embedding (KGE, Node2Vec, or Text+) against the Flat key-value embedding. Positive Δ values indicate that leveraging ontological structure improves retrieval. Tasks sorted by absolute advantage. 147

5.8 Text+ feature ablation (mean adjusted precision@10 across the three instance graphs). Each row removes or isolates specific feature groups from the natural language description. Best value per column in bold. 148

5.9 Quick Reference Guide for the Knowledge Propagation Signature 156

5.10 Comparison of contextual reasoning approaches for SemTS. 180

6.1 Baseline results showing the best performing model and corresponding MAE for each dataset. 209

A.1 Selection of external sources with semantically aligned Scenario Knowledge vocabulary examples, organized by category. 291

B.1 Saturation analysis: mapping output types from ten independent libraries to existing Data Knowledge vocabulary categories. 292

B.2 Complete output-to-vocabulary mappings for the Data Knowledge coverage evaluation. For each of the seven libraries, the principal output categories were identified by systematically reviewing the public API documentation. Each row represents a semantically distinct output category; individual functions producing outputs of the same category are consolidated into a single entry. Vocabulary terms reference subcategories of the Data Knowledge vocabulary. 293

B.3 Vocabulary categories exercised by library outputs (comprehensive functional verification). Categories reference second-level concepts of the Data Knowledge vocabulary, grouped by parent branch. ✓ means at least one primary output mapped, ∼ means secondary or indirect mapping only, – means no outputs mapped. 296

B.4 Complete cross-domain applicability mappings for the Scenario Knowledge vocabulary. For each domain, representative contextual information items a practitioner would realistically need to capture are listed together with the vocabulary terms used for classification. 296

C.1 SemTS Ontology: Requirements and Solutions 299

Listings

2.1 Example use of the Semantic Sensor Network Ontology represented in the Turtle serialization format. 29

4.1 Example of a SHACL Node Shape to ensure that every `Segment Prediction` was generated by exactly one `Knowledge Generation Method` 99

5.1 Example of using the SemTS Ontology Population Library 113
5.2 Example competency question in SPARQL: Does the generated instance correctly capture the window size parameter from task 9? 123
5.3 SemTS-based Retrieval: Feature Engineering 129
5.4 SemTS-based Retrieval: Informed Analytics 130
5.5 SemTS-based Retrieval: Experimental Tracking 131
5.6 SPARQL CONSTRUCT query for reasoning in SemTS. 177

6.1 Context-free grammar to represent different hierarchical levels and compositions of time series data. Lowercase symbols allow for path tracking. The terminal symbol is K representing any associated knowledge. 195
6.2 SPARQL condition verifying that all segments have associated labels. 197
6.3 A simple RDF example returned from the Graph RAG system showing a retrieved minimum value of a time series. 207

Appendices

A Categorical Overview of Scenario Knowledge

Source and Description	Derived Class Examples
Initial Expert Iteration	Annotation, ContextualDescription, Event, Observation, Action, Alert, LifecycleEvent, CategorizedEvent, UncategorizedEvent, Status
Provenance, Annotations, and Data Quality	
W3C Web Annotation Data Model & PROV-O - Annotations, temporal selectors, provenance of activities, entities, and agents	Annotation, Note, Keyword, Code, Purpose, Event, Involvement, AffectedEntity, CausalFactor, Fact
W3C Data Quality Vocabulary (DQV) - Data quality dimensions, metrics, and measurements	QualityAnnotation, Assumption, Certainty, Status
Measurement and Sensing Infrastructure	
W3C/OGC SOSA/SSN & OGC SensorThings API - Sensor descriptions, deployments, and location metadata	Deployment, SensorDeployment, Location, EnvironmentalCondition
Events, Processes, and Operations	
IEEE 1849-2016 XES & OMG BPMN 2.0 - Event logs, lifecycle transitions, and business process modeling	LifecycleEvent, CategorizedEvent, Error, ProcessStep, Milestone, UserEvent
W3C ActivityStreams 2.0 & iCalendar (RDF) - User actions, activity streams, and calendar entries	BusinessEvent, Action, CalendarEntry
RFC 5424 Syslog - Log message structures with severity and facility classifiers	Alert, Severity, Code

Continued on next page

Source and Description	Derived Class Examples
ISO 55000 & ISO 14224 - Asset management principles and reliability/maintenance data collection	MaintenanceEvent, PreventiveMaintenanceEvent, CorrectiveMaintenanceEvent, PredictiveMaintenanceEvent, MaintenancePlan, AffectedEntity, CauseCode, DefectType, SensorReplacement
ISO 22400 & OMAC PackML / ISA-TR88.00.02 - Manufacturing KPIs, state models, and machine states	DowntimeCategory, PlannedDowntime, UnplannedDowntime, SetupChangeover, ProductionCycleStart, ProductionCycleComplete, OperatingMode, Status
ISA-88 (IEC 61512) & ISA-95 / IEC 62264 - Batch control, recipes, and operations management	Recipe, DispatchInstruction, WorkInstruction
Alerting and Alarm Management	
OASIS CAP, OPC UA Alarms & Conditions, ISA-18.2 / IEC 62682 - Alert formats, industrial alarm semantics, and lifecycle frameworks	Alert, AlarmRule, Urgency, Acknowledgment, ResetOrClear, ThresholdBreach, Guideline
Formal Knowledge and Documentation	
MathML & Modelica - Mathematical expressions and dynamic system models	MathExpression, AlgebraicEquation, PolynomialEquation, ExponentialEquation, LogarithmicEquation, TrigonometricEquation, IntegralEquation, DifferentialEquation
RuleML - Declarative rules and logic conditions	Rule, Constraint, Condition
ISO 9001 & ISO 10013 - Quality management systems and documentation practices	Documentation, Guideline, WorkInstruction, ControlPlan, MaintenancePlan, SafetyProcedure, NonConformance, InspectionEvent, ComplianceCheck
Failure and Risk Analysis	

Continued on next page

Source and Description	Derived Class Examples
AIAG/VDA FMEA, IEC 60812 & ISO 13381-1 - Failure mode analysis, risk prioritization, and prognostics	CausalFactor, DefectType, Severity, Assumption, NonConformance, RUL, Certainty
Socio-Behavioral and Organizational Context	
PMI & PRINCE2 - Project management, milestones, and resource tracking	Milestone, BusinessEvent
Validation and Reasoning	
W3C PROV-O - Derivation tracking and invalidation of asserted information	Inference, InferredFact, Revision, Retraction
Toulmin's Argumentation Model - Structured reasoning with claims, evidence, warrants, and rebuttals	Evidence, SupportingEvidence, ContradictingEvidence, Justification, Conclusion, Contradiction
Information Artifact Ontology (IAO) - Explicit definitions of hypothesis, evidence, and conclusion	Hypothesis, Assumption, Evidence, Conclusion, Explanation
ISO 9001 - Verification, validation processes, and approval workflows	ValidationEvent, ValidationStatus, ValidationOutcome, Approval, Rejection

Table A.1: Selection of external sources with semantically aligned Scenario Knowledge vocabulary examples, organized by category.

B SemTS Vocabularies: Evaluation Details

This appendix provides the detailed per-item mappings underlying the vocabulary evaluation presented in Section 3.6.4.

Table B.1: Saturation analysis: mapping output types from ten independent libraries to existing Data Knowledge vocabulary categories.

Library	**Key Outputs**	**Vocabulary Categories**
Prophet [TL18]	Forecasts, components, changepoints	DataLevelPrediction, SegmentDataCharacteristic, AnnotationLevelPrediction
PyOD [ZNL19]	Anomaly scores, binary labels	AnnotationLevelPrediction
GluonTS [Ale+20]	Probabilistic & quantile forecasts	DataLevelPrediction
Merlion [Bha+21]	Forecasts, anomaly scores	DataLevelPrediction, AnnotationLevelPrediction
NeuralProphet [Tri+21]	Forecasts, trend/seasonal/AR comp.	DataLevelPrediction, SegmentDataCharacteristic
ruptures [TOV20]	Change point indices	AnnotationLevelPrediction
PyCaret [Ali20]	Forecasts, model comparison metrics	DataLevelPrediction, AnnotationLevelPrediction, ModelInstance
arch [She+23]	Conditional variance, volatility fcst.	SegmentDataCharacteristic, DataLevelPrediction, ModelInstance
catch22 [Lub+19]	Feature values	SegmentDataCharacteristic
dtaidistance [Mee+20]	DTW distances, warping paths	SegmentRelation

Table B.2: Complete output-to-vocabulary mappings for the Data Knowledge coverage evaluation. For each of the seven libraries, the principal output categories were identified by systematically reviewing the public API documentation. Each row represents a semantically distinct output category; individual functions producing outputs of the same category are consolidated into a single entry. Vocabulary terms reference subcategories of the Data Knowledge vocabulary.

Output Category	**Vocabulary Category**
tsfresh (automated feature extraction)	
Mean, variance, skewness, kurtosis, quantiles	LocationProperty, DispersionProperty
Autocorrelation, partial autocorrelation, linear trend	Correlation, StructuralProperty, TrendProperty
FFT coefficients, CWT coefficients	StructuralProperty
Augmented Dickey–Fuller statistic and p-value	StructuralProperty
Count above mean, percentage of reoccurring values	DispersionProperty, DataInformationProperty
Feature relevance scores	AnnotationLevelPrediction (FeatureImportance)†
STUMPY (Matrix Profile computation)	
Matrix Profile (nearest-neighbor distances per subsequence)	Distance
Motifs (top-k recurring subsequences)	AnnotationLevelPrediction (PatternElement)
Discords (most dissimilar subsequences)	AnnotationLevelPrediction (Anomaly)
Pattern chains (temporally ordered motif sequences)	AnnotationLevelPrediction (PatternElement)
Snippets (representative regime segments)	SegmentGroupRepresentative†
FLOSS regime change scores	AnnotationLevelPrediction (Changepoint)
sktime (unified ML framework for time series)	
Forecaster predictions (point and interval)	DataLevelPrediction (Forecast)
Cluster labels (e.g., TimeSeriesKMeans)	AnnotationLevelPrediction (InferredClassID)
Shapelet features (ShapeletTransform)	SegmentGroupRepresentative (Shapelet)
DTW distance values	Distance (DTWDistance)

Continued on next page

Output Category	Vocabulary Category
STL decomposition (trend, seasonal, residual)	TrendProperty, StructuralProperty, DispersionProperty
Change point indices (e.g., ClaSPSegmentation)	AnnotationLevelPrediction (Changepoint)
Statistical summaries (SummaryTransformer)	LocationProperty, DispersionProperty
SAX symbolic representations	StructuralProperty
Fitted model objects (serializable)	ModelInstance
Classification predictions (class labels)	AnnotationLevelPrediction (InferredClassID)
Statsmodels (classical statistical modeling)	
OLS regression coefficients, R^2, p-values	DataLevelPrediction, TrendProperty
ARIMA forecasts and confidence intervals	DataLevelPrediction (Forecast, ProbabilisticForecast)
Granger causality F-statistic and p-values	Causality
Augmented Dickey–Fuller test results	StructuralProperty
Cointegration test results (Johansen, Engle–Granger)	Correlation
Seasonal decomposition components (trend, seasonal, residual)	TrendProperty, StructuralProperty, DispersionProperty
ACF and PACF values at specified lags	Correlation, StructuralProperty
Fitted model objects	ModelInstance
Regression residuals (fitted − observed)	DispersionProperty (ResidualComponent)†
Tigramite (causal discovery)	
PCMCI+ causal graph (DAG with lag structure)	Causality (CausalGraph)
Conditional independence test results (p-values per link)	Causality
Causal effect estimates (mediation, direct, total)	Causality (InterventionEffect)
lifelines (survival analysis)	
Kaplan–Meier survival curves	AnnotationLevelPrediction (PredictedEvent)
Nelson–Aalen cumulative hazard estimates	AnnotationLevelPrediction (PredictedEvent)
Cox proportional hazard ratios	Correlation
Log-rank test statistic and p-value	Correlation

Continued on next page

Output Category	Vocabulary Category
Median survival time	AnnotationLevelPrediction (PredictedEvent)
Fitted model objects	ModelInstance
Darts (probabilistic forecasting & anomaly detection)	
Point forecasts	DataLevelPrediction (Forecast)
Probabilistic forecasts (sample-based)	DataLevelPrediction (ProbabilisticForecast)
Anomaly scores (e.g., KMeansScorer)	AnnotationLevelPrediction (Anomaly)
Binary anomaly labels (e.g., QuantileDetector)	AnnotationLevelPrediction (Anomaly)
Seasonality check (period, significance)	StructuralProperty (SeasonalComponent)
Fitted model objects (save/load)	ModelInstance
Backtesting evaluation metrics	ModelInstance[†]

[†]Output category required interpretive judgment during classification (cf. Section 3.6.4).

Table B.3: Vocabulary categories exercised by library outputs (comprehensive functional verification). Categories reference second-level concepts of the Data Knowledge vocabulary, grouped by parent branch. ✓ means at least one primary output mapped, ∼ means secondary or indirect mapping only, – means no outputs mapped.

Vocabulary Category	*tsfresh*	*STUMPY*	*sktime*	*Statsmodels*	*Tigramite*	*lifelines*	*Darts*
SegmentDataCharacteristic							
LocationProperty	✓	–	✓	✓	–	–	–
DispersionProperty	✓	–	✓	∼	–	–	–
TrendProperty	✓	–	✓	✓	–	–	–
StructuralProperty	✓	–	✓	✓	–	–	∼
DataInformationProperty	✓	–	✓	∼	–	–	∼
SegmentPrediction							
DataLevelPrediction	–	–	✓	✓	✓	–	✓
AnnotationLevelPrediction	✓	✓	✓	✓	–	✓	✓
SegmentRelation							
Distance	–	✓	✓	–	–	–	✓
Similarity	–	–	✓	–	–	–	–
Correlation	✓	∼	✓	✓	✓	∼	✓
Causality	–	–	–	✓	✓	–	–
SegmentGroupRepresentative	–	∼	✓	–	–	–	–
ModelInstance	–	–	✓	✓	✓	✓	✓

Table B.4: Complete cross-domain applicability mappings for the Scenario Knowledge vocabulary. For each domain, representative contextual information items a practitioner would realistically need to capture are listed together with the vocabulary terms used for classification.

Contextual Information	**Vocabulary Term(s)**
Manufacturing - Blow Molding	

Continued on next page

Contextual Information	Vocabulary Term(s)
Temperature setpoint is 185°C	Setpoint
Setpoint was changed 10 min ago	SetpointChange
Alert: wall thickness out of specification	ThresholdBreach, NonConformance
Alarm rule: wall thickness < 0.8 mm	AlarmRule, Threshold
Operator acknowledged the alarm	Acknowledgment
Recipe: "PET-500ml-v3"	Recipe
Recipe was changed at shift start	RecipeChangeEvent
Shift change at 06:00	ShiftChange
Production cycle #4521 completed	ProductionCycleComplete
Corrective maintenance scheduled	CorrectiveMaintenanceEvent
Root cause: nozzle wear	CauseCode, WearState
Operator note: "Suspect material batch issue"	Note, Assumption
Healthcare - ICU Patient Monitoring	
Heart rate alarm threshold: 120 BPM	Threshold, AlarmRule
Alarm triggered: tachycardia detected	Alert, ThresholdBreach
Severity: moderate	Severity
Nurse note: "Patient agitated after waking"	Note, Explanation
Medication administered at 14:30	Action, CalendarEntry
Doctor hypothesis: dehydration-induced	Hypothesis
Low blood pressure confirms dehydration	SupportingEvidence, Correlation
Diagnosis confirmed by attending physician	Confirmation, Approval
Patient location: Ward 3, Bed 12	Location
Pulse oximeter on left index finger	SensorDeployment, Location
SpO2 sensor replaced (drift detected)	SensorReplacement, CalibrationStatus
Expected baseline heart rate: 60–80 BPM	Baseline, Reference
Protocol: escalate if HR > 150 for 5 min	Rule, TemporalConstraint
Patient in post-surgery recovery phase	ProcessStep, Status
Finance - Algorithmic Trading	
Earnings announcement at market open	BusinessEvent, ExternalEvent
Analyst note: "Price drop correlated with sector"	Note, Correlation, CausalFactor
Expected price range Q4: \$140–\$160	Reference, ExpectedProfile, Tolerance
Stop-loss threshold at -5%	Threshold, Constraint
Stop-loss changed from -5% to -3%	ThresholdChange
Regulatory constraint: min. 2-day holding period	TemporalConstraint, OperationalConstraint

Continued on next page

Contextual Information	Vocabulary Term(s)
Model prediction: bullish trend next 5 days	Forecast (under Plan)
Confidence level: 72%	Certainty
Positive RSI divergence supports prediction	SupportingEvidence
Inverted yield curve contradicts outlook	ContradictingEvidence
Trading halt imposed by exchange	ExternalEvent†
Market regime: high volatility	Status, OperatingMode†
Environmental Monitoring - Weather Station	
Sensor calibrated on 2024-06-15	CalibrationEvent, CalibrationStatus
Sensor drift detected over 3 months	WearState, QualityAnnotation
Sensor replaced	SensorReplacement
Station location: 48.1°N, 11.6°E, 520 m	Location
Urban heat island effect	EnvironmentalCondition, ExternalCondition
30-year climate normal for July	Baseline, Reference
Unusually high nighttime temperatures	Observation, UncharacteristicObservation
Sensor range: −40°C to +60°C	CapabilityLimit, Constraint
Hypothesis: heat dome causing anomaly	Hypothesis, ExternalCondition
Data quality flag: suspect readings during storm	QualityAnnotation
Communication loss during power outage	CommunicationLoss, PowerOutage
Software update applied to data logger	SoftwareUpdate
Smart Building - Energy Management	
HVAC mode: cooling → heating transition	ModeChange, OperatingMode
Temperature setpoint: 22°C	Setpoint
Setpoint changed for weekend schedule	SetpointChange, Schedule
Occupancy schedule: M–F 08:00–18:00	Schedule, CalendarEntry
Public holiday: building unoccupied	CalendarEntry, ExternalEvent
Maintain 20–24°C during occupied hours	Guideline, Constraint, Threshold
Power outage at 03:15	PowerOutage
Energy audit inspection	InspectionEvent, ComplianceCheck
Thermal model of building envelope	DifferentialEquation
Expected energy profile for summer	ExpectedProfile
Insulation degradation causes 15% increase	Conclusion, CausalFactor
HVAC filter change every 3 months	MaintenancePlan, PreventiveMaintenanceEvent

†Classifiable at parent level; no specific descendant term available in the vocabulary (cf. Section 3.6.4).

C SemTS Requirements

Table C.1: SemTS Ontology: Requirements and Solutions

SemTS Ontology: Requirements and Solutions		
Category	**Requirement**	**Solution in SemTS**
Data Management	The ontology should allow for scalability to handle large and complex time series datasets, including nested and hierarchical relationships.	Reuse of `dcat:Dataset`; dedicated transitive `semts:isPartOf` property for hierarchical segment relationships
Data Management	The ontology should facilitate a detailed and structured description of time series data specifics	Reuse of `dcat:Dataset`
Data Management	The ontology should be adaptable to diverse data structures and formats to accommodate varying data types of expressing knowledge	Class `ReferencedValue` extending `dcat:Dataset` to reference any complex data structure. Reuse of QUDT to define specifics regarding datatype and units
Data Management	The ontology should represent segment data information, including dimensionality and time interval details, to handle diverse time series configurations.	Class `TimeSeriesSegment` relates to `DataDimension` and reuses OWL-Time for index information

SemTS Ontology: Requirements and Solutions (continued)		
Category	**Requirement**	**Solution in SemTS**
Data Management	The ontology should incorporate descriptions of data not already in the dataset, such as information from forecasts.	A dedicated class `ArtificialTimeSeriesSegment` indicates segments which are not definable as part of existing segment information
Data Management	The ontology should enable the representation of overlapping segments within time series data, associating characteristic knowledge with arbitrary intervals or data points.	Index-specific overlappings of any type can be defined by the reuse of OWL-Time
Data Management	The ontology should allow for the description of metadata, such as dataset titles and descriptions, to enhance documentation.	Reuse of `dcterms:title` and `dcterms:description`
Data Management	The ontology should enable hierarchical segment organization.	Dedicated transitive, irreflexive `semts:isPartOf` relation for segment hierarchies
Data Management	The ontology should represent dimensional details using a dedicated class and corresponding properties.	Class `DataDimension`
Data Management	The ontology should accommodate diverse data formats and database systems to integrate heterogeneous data sources.	Utilization of QUDT in subclasses of `Value`
Data Management	The ontology should allow for a description of value units.	Utilization of QUDT in subclasses of `Value`

SemTS Ontology: Requirements and Solutions (continued)		
Category	**Requirement**	**Solution in SemTS**
General	The ontology should support wide applicability and flexibility to increase interoperability.	Reuse of multiple common ontologies and vocabularies such as DCAT, OWL-Time, QUDT, PROV, ML-Schema etc.
General	The ontology should adhere to Semantic Web best practices for documentation and publication.	W3ID for permanent identifier, Git for repository management and WIDOCO for documentation
General	The ontology should be application-agnostic to facilitate compatibility with various setups and platforms.	Modularization of methodical and scenario-specific details into SKOS concepts
General	The ontology should employ established semantic approaches, such as leveraging DCAT, OWL-Time, and ML-Schema, for compatibility and expressiveness.	Reuse of multiple common ontologies and vocabularies such as DCAT, OWL-Time, QUDT, PROV, ML-Schema etc.
General	The ontology should enable advanced querying, inference, and reasoning operations for efficient reuse of time series knowledge.	Serialization of the ontology into different formats; multiple OWL axioms for inference
General	The ontology should allow the indexing of time series data through segment-wise consideration of knowledge, indirectly serving as an index structure.	Relations `segmentKnowledge` and `multiSegmentKnowledge`

SemTS Ontology: Requirements and Solutions (continued)		
Category	**Requirement**	**Solution in SemTS**
General	The ontology should enable the representation and integration of generated time series insights and domain information.	Multiple dedicated subclasses of `Knowledge` for a coarse categorization of knowledge; detailed differentiation and description through concepts within additional vocabularies
Knowledge Generation	The ontology should support provenance tracking to allow backtracking of the origin and evolution of knowledge and insights.	Reuse of PROV between classes involved in the knowledge generation process
Knowledge Generation	The ontology should allow for references to descriptions of data analysis runs, supporting complex and iterative analytical workflows.	Class `KnowledgeGenerationMethod` extends `mls:Implementation` and thus can be referenced by instances from ML-Schema (e.g., `mls:Run mls:realizes semts:KnowledgeGenerationMethod`)
Knowledge Generation	The ontology should represent entities involved in knowledge generation, such as models, algorithms, or processes.	Either via subclasses of `KnowledgeGenerationEntity` or referenceable information from ML-Schema
Knowledge Generation	The ontology should represent inputs and outputs of knowledge generation processes.	Represented by subclasses of `KnowledgeGenerationIO`
Knowledge Generation	The ontology should support knowledge reasoning and inference tasks to describe advanced analytical processes.	Representable by knowledge propagation within SemTS; knowledge outputs can be referenced as `inputKnowledge` to subsequent knowledge generation processes

SemTS Ontology: Requirements and Solutions (continued)		
Category	**Requirement**	**Solution in SemTS**
Knowledge Generation	The ontology should allow for different types of knowledge generation including manual reasoning, results generated from algorithms or models and derived by comparisons.	Multiple subclasses of `KnowledgeGenerationEntity` are available
Knowledge Management	The ontology should promote the organized and semantic representation of time series knowledge to enable meaningful propagation of insights.	Representable by knowledge propagation within SemTS; knowledge outputs can be referenced as `inputKnowledge` to subsequent knowledge generation processes
Knowledge Management	The ontology should organize defined knowledge as characteristics derived from data analyses, methodical presets, and domain- or scenario-specific information.	Multiple subclasses of `Knowledge` are available
Knowledge Management	The ontology should allow for the mapping of knowledge to entire time series or subsequences of potentially reduced dimensionality.	Any segment is definable via separate index and dimension information
Knowledge Management	The ontology should enable nested groupings of knowledge to infer hierarchies of structural and semantic relationships.	Class `KnowledgeGroup` which is recursively defined as `Knowledge` and thus allows for nested hierarchies
Knowledge Management	The ontology should support the association of knowledge with specific time intervals or data points.	Any knowledge is directly or transitively referenced by one or multiple segments

SemTS Ontology: Requirements and Solutions (continued)		
Category	**Requirement**	**Solution in SemTS**
Knowledge Management	The ontology should allow classification of anomalies, motifs, and other characteristic features of time series data.	Methodical specifics can be defined in or used from the SemTS data knowledge vocabulary or compatible, external vocabularies such as vocabularies from MLSO
Knowledge Management	The ontology should provide mechanisms for associating domain expert knowledge with specific data intervals or sequences.	Knowledge can be defined via `ScenarioKnowledge` and detailed by concepts from the SemTS scenario knowledge vocabulary
Knowledge Management	The ontology should allow defining and linking knowledge qualities like reliability and validity to categorized time series knowledge.	Class `KnowledgeQualityMeasure`
Knowledge Management	The ontology should support advanced reuse of analytical details for recommendations.	Subclasses of `MethodKnowledge` inherit from ML-Schema to define details, while also allowing for the assignment of particular concepts available via different vocabularies such as compatible ones from MLSO
Knowledge Management	The ontology should provide a solution to compare segments through dedicated Knowledge classes.	Class `SegmentRelation`
Knowledge Management	The ontology should support the modular enrichment of specific knowledge concepts.	Use of `skos:Concept`, which allows for the assignment of details from dedicated vocabularies (either SemTS or external vocabularies such as MLSO)

SemTS Ontology: Requirements and Solutions (continued)		
Category	**Requirement**	**Solution in SemTS**
Knowledge Management	The ontology should support the definition of explicit model information associated with a particular run of an analysis task.	Reuse of ML-Schema
Knowledge Management	The ontology should allow for the reference of concrete analysis model instances independent of their level of explainability.	Class `ModelInstance` to represent knowledge that is, for instance, a reference to a pretrained AI model
References: Listed external schemas are documented in Section 4.3.2.		

D Real-World Application Scenarios: Code Listings

Animal Welfare Example

```
@prefix : <http://example.org/animal-welfare#> .
@prefix semts: <https://w3id.org/semts/ontology#> .
@prefix semts-dv: <https://w3id.org/semts/vocabulary/data-knowledge#> .
@prefix semts-sv: <https://w3id.org/semts/vocabulary/scenario-knowledge#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix unit: <http://qudt.org/vocab/unit/> .
@prefix qudt: <http://qudt.org/schema/qudt/> .
@prefix mls: <http://www.w3.org/ns/mls#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .


#    Knowledge Generation Concepts

:ObjectDetectionConcept a semts:KnowledgeGenerationConcept ;
skos:prefLabel "Object Detection"@en ;
rdfs:comment "Deep learning-based object detection method"@en .

:PeakAnalysisConcept a semts:KnowledgeGenerationConcept ;
skos:prefLabel "Peak Analysis"@en ;
rdfs:comment "Statistical peak detection and analysis method"@en .


#    Main Recording Segment - Time Index

:recording1-index a time:Interval ;
time:hasBeginning :recording1-start ;
time:hasEnd :recording1-end .
```

```
:recording1-start a time:Instant ;
time:inXSDDateTime "2025-09-15T08:00:00.000Z"^^xsd:dateTime .

:recording1-end a time:Instant ;
time:inXSDDateTime "2025-09-15T08:00:10.000Z"^^xsd:dateTime .


#    Data Dimensions

:dim-video a semts:DataDimension ;
semts:title "Video (frames)" ;
semts:description "Video frames captured from camera feed" ;
semts:dimensionPosition "1"^^xsd:int .

:dim-audio a semts:DataDimension ;
semts:title "Audio RMS amplitude" ;
semts:description "Root mean square amplitude of audio signal" ;
semts:dimensionPosition "2"^^xsd:int ;
semts:hasUnit unit:DeciB .

:dim-paddle a semts:DataDimension ;
semts:title "Paddle velocity" ;
semts:description "Velocity measurements from paddle sensor" ;
semts:dimensionPosition "3"^^xsd:int ;
semts:hasUnit unit:M-PER-SEC .


#    Distribution for Main Recording

:dist-recording1 a dcat:Distribution ;
dct:description "Logical dataset grouping; per-signal tables are referenced by
    each child segment." ;
dct:format "application/sql" ;
dcat:accessURL <postgresql://localhost:5432/animal_welfare> .


#    Main Recording Segment (Multivariate)

:seg-recording1 a semts:TimeSeriesSegment ;
dct:title "Recording multivariate data" ;
dct:description "10-second multivariate recording containing video, audio, and
    paddle velocity data" ;
semts:segmentIndex :recording1-index ;
semts:segmentDimension :dim-video, :dim-audio, :dim-paddle ;
dcat:distribution :dist-recording1 .


#    Dimensional Subsegments of Recording1
```

```
:seg-recording1-video a semts:TimeSeriesSegment ;
dct:title "Recording video dimension only" ;
dct:description "Video frames from the main recording" ;
semts:isPartOf :seg-recording1 ;
semts:segmentIndex :recording1-index ;
semts:segmentDimension :dim-video ;
dcat:distribution :dist-recording1 .

:seg-recording1-audio a semts:TimeSeriesSegment ;
dct:title "Recording audio dimension only" ;
dct:description "Audio RMS amplitude from the main recording" ;
semts:isPartOf :seg-recording1 ;
semts:segmentIndex :recording1-index ;
semts:segmentDimension :dim-audio ;
dcat:distribution :dist-recording1 .

:seg-recording1-paddle a semts:TimeSeriesSegment ;
dct:title "Recording paddle dimension only" ;
dct:description "Paddle velocity measurements from the main recording" ;
semts:isPartOf :seg-recording1 ;
semts:segmentIndex :recording1-index ;
semts:segmentDimension :dim-paddle ;
dcat:distribution :dist-recording1 .


#    Knowledge Generation Method: YOLO Object Detection

:yolo-detection-method a semts:KnowledgeGenerationMethod ;
semts:knowledgeGenerationConcept :ObjectDetectionConcept ;
semts:title "YOLOv8 Person Detection" ;
semts:description "Deep learning object detection using YOLOv8 for identifying
    persons in video frames" ;
mls:hasHyperParameter :yolo-confidence-param, :yolo-iou-param ;
semts:used :yolo-input ;
semts:generated :yolo-output .

:yolo-confidence-param a mls:HyperParameter ;
mls:hasValue "0.5"^^xsd:float .

:yolo-iou-param a mls:HyperParameter ;
mls:hasValue "0.4"^^xsd:float .

:yolo-input a semts:KnowledgeGenerationInput ;
semts:title "Video frames input" ;
semts:description "Input video frames for YOLO object detection" ;
semts:inputDataReference :seg-recording1-video .

:yolo-output a semts:KnowledgeGenerationOutput ;
```

```
semts:title "Person detection results" ;
semts:description "Output containing detected person with bounding boxes" ;
semts:outputKnowledge :person-detection-knowledge ;
semts:wasDerivedFrom :yolo-input .


#    Knowledge Generation Method: Paddle Hit Detection

:paddle-hit-method a semts:KnowledgeGenerationMethod ;
semts:knowledgeGenerationConcept :PeakAnalysisConcept ;
semts:title "Paddle Hit Detection via Peak Analysis" ;
semts:description "Statistical analysis method detecting velocity peaks above
    threshold to identify paddle hits" ;
mls:hasHyperParameter :paddle-velocity-threshold-param, :paddle-window-size-
    param ;
semts:used :paddle-input ;
semts:generated :paddle-output .

:paddle-velocity-threshold-param a mls:HyperParameter ;
mls:hasValue "3.0"^^xsd:float .

:paddle-window-size-param a mls:HyperParameter ;
mls:hasValue "50"^^xsd:integer .

:paddle-input a semts:KnowledgeGenerationInput ;
semts:title "Paddle velocity data input" ;
semts:description "Input paddle velocity time series for hit detection" ;
semts:inputDataReference :seg-recording1-paddle .

:paddle-output a semts:KnowledgeGenerationOutput ;
semts:title "Paddle hit detection results" ;
semts:description "Output containing detected paddle hit events" ;
semts:outputKnowledge :paddle-hit-knowledge ;
semts:wasDerivedFrom :paddle-input .


#    Person Detection Event Segment - Time Index (~4.5 seconds)

:idx-person-04480-04520 a time:Interval ;
time:hasBeginning :person-start ;
time:hasEnd :person-end .

:person-start a time:Instant ;
time:inXSDDateTime "2025-09-15T08:00:04.480Z"^^xsd:dateTime .

:person-end a time:Instant ;
time:inXSDDateTime "2025-09-15T08:00:04.520Z"^^xsd:dateTime .
```

```
#    Person Detection Segment

:seg-video-person-04480-04520 a semts:TimeSeriesSegment ;
dct:title "Video segment with person detection" ;
dct:description "Video frames where a person was detected at approximately 4.5
    seconds" ;
semts:isPartOf :seg-recording1 ;
semts:segmentIndex :idx-person-04480-04520 ;
semts:segmentDimension :dim-video ;
dcat:distribution :dist-recording1 ;
semts:segmentKnowledge :person-detection-knowledge .


#    Person Detection Knowledge Group

:person-detection-knowledge a semts:KnowledgeGroup ;
semts:knowledgeConcept semts-dv:PredictedEvent ;
semts:title "Identified Person" ;
semts:description "Knowledge group containing person detection results including
     class label and bounding boxes" ;
semts:knowledgeGenerationEntity :yolo-detection-method ;
semts:groupKnowledge :person-detected-class, :person-bboxes .

:person-detected-class a semts:SegmentPrediction ;
semts:knowledgeConcept semts-dv:InferredClassID ;
semts:title "Detected Class" ;
semts:description "Class label assigned by YOLO object detection" ;
semts:knowledgeGenerationEntity :yolo-detection-method ;
semts:hasValue :person-class-value .

:person-class-value a semts:EmbeddedValue ;
semts:valueString "PERSON" ;
semts:hasDatatype qudt:StringEncodingType .

:person-bboxes a semts:SegmentPrediction ;
semts:knowledgeConcept semts-dv:InferredAnnotation ;
semts:title "Person BBoxes" ;
semts:description "Bounding box coordinates for detected person" ;
semts:knowledgeGenerationEntity :yolo-detection-method ;
semts:hasValue :person-bboxes-values .

:JSON a qudt:Datatype ;
rdfs:label "JSON" .

:person-bboxes-values a semts:ReferencedValue ;
semts:title "YOLO Bounding Boxes" ;
dct:description "JSON file containing bounding box coordinates for detected
    person" ;
dcat:distribution :dist-yolo-bboxes-person-04480-04520 ;
```

```
semts:hasDatatype :JSON .

:dist-yolo-bboxes-person-04480-04520 a dcat:Distribution ;
dct:description "YOLO bounding box coordinates for detected person in JSON
    format" ;
dct:format "application/json" ;
dcat:accessURL <file:///data/yolo_results/person_bboxes_04480_04520.json> .


#    Paddle Hit Event Segment - Time Index (~5.1 seconds)

:idx-hit-05100-05150 a time:Interval ;
time:hasBeginning :hit-start ;
time:hasEnd :hit-end .

:hit-start a time:Instant ;
time:inXSDDateTime "2025-09-15T08:00:05.100Z"^^xsd:dateTime .

:hit-end a time:Instant ;
time:inXSDDateTime "2025-09-15T08:00:05.150Z"^^xsd:dateTime .


#    Paddle Hit Detection Segment

:seg-paddle-hit-05100-05150 a semts:TimeSeriesSegment ;
dct:title "Paddle velocity subsegment @ ~5.1s" ;
dct:description "Paddle velocity segment where a hit event was detected at
    approximately 5.1 seconds" ;
semts:isPartOf :seg-recording1 ;
semts:segmentIndex :idx-hit-05100-05150 ;
semts:segmentDimension :dim-paddle ;
dcat:distribution :dist-recording1 ;
semts:segmentKnowledge :paddle-hit-knowledge .


#    Paddle Hit Knowledge Group

:paddle-hit-knowledge a semts:KnowledgeGroup ;
semts:knowledgeConcept semts-dv:PredictedEvent ;
semts:title "Paddle Hit" ;
semts:description "Knowledge group containing paddle hit detection results
    including classification and velocity measurement" ;
semts:knowledgeGenerationEntity :paddle-hit-method ;
semts:groupKnowledge :paddle-hit-class, :paddle-hit-velocity .

:paddle-hit-class a semts:SegmentPrediction ;
semts:knowledgeConcept semts-dv:InferredClassID ;
semts:title "Hit Classification" ;
semts:description "Classification of the detected paddle event as a HIT" ;
```

```
semts:knowledgeGenerationEntity :paddle-hit-method ;
semts:hasValue :paddle-hit-class-value .

:paddle-hit-class-value a semts:EmbeddedValue ;
semts:valueString "HIT" ;
semts:hasDatatype qudt:StringEncodingType .

:paddle-hit-velocity a semts:SegmentDataCharacteristic ;
semts:knowledgeConcept semts-dv:MeanNAbsoluteMax ;
semts:title "Mean of n absolute maxima (paddle velocity)" ;
semts:description "Peak velocity measurement during the paddle hit event" ;
semts:knowledgeGenerationEntity :paddle-hit-method ;
semts:hasValue :paddle-hit-velocity-value .

:paddle-hit-velocity-value a semts:EmbeddedValue ;
semts:valueString "3.8" ;
semts:hasDatatype qudt:DoublePrecisionType ;
semts:hasUnit unit:M-PER-SEC .


#    Quality Measures

:yolo-confidence-measure a semts:KnowledgeQualityMeasure ;
semts:qualityMeasureConcept :ConfidenceScoreConcept ;
semts:title "YOLO Detection Confidence" ;
semts:description "Confidence score for the person detection" ;
semts:hasValue :yolo-confidence-value .

:ConfidenceScoreConcept a semts:QualityMeasureConcept ;
skos:prefLabel "Confidence Score"@en ;
rdfs:comment "Probability-based confidence measure for predictions"@en .

:yolo-confidence-value a semts:EmbeddedValue ;
semts:valueString "0.92" ;
semts:hasDatatype qudt:DoublePrecisionType .


# Link quality measure to knowledge

:person-detected-class semts:knowledgeQualityMeasure :yolo-confidence-measure .
```

Grinding Beads Example

```
@prefix :           <http://example.org/grinding-bead#> .
@prefix semts:      <https://w3id.org/semts/ontology#> .
@prefix rdf:        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:        <http://www.w3.org/2001/XMLSchema#> .
@prefix time:       <http://www.w3.org/2006/time#> .
@prefix unit:       <http://qudt.org/vocab/unit/> .
@prefix qudt:       <http://qudt.org/schema/qudt/> .
@prefix skos:       <http://www.w3.org/2004/02/skos/core#> .
@prefix mls:        <http://www.w3.org/ns/mls#> .
@prefix dcat:       <http://www.w3.org/ns/dcat#> .
@prefix dct:        <http://purl.org/dc/terms/> .
@prefix rdfs:       <http://www.w3.org/2000/01/rdf-schema#> .
@prefix semts-sv:   <https://w3id.org/semts/vocabulary/scenario-knowledge#> .
@prefix semts-dv:   <https://w3id.org/semts/vocabulary/data-knowledge#> .
@prefix mlso-algo:  <http://example.org/mlso-algo#> .
@prefix mlso-em:    <http://example.org/mlso-em#> .

# External Vocabulary Concept Typing

mlso-algo:LSTM a semts:KnowledgeGenerationConcept ;
skos:prefLabel "LSTM"@en ;
rdfs:comment "Long Short-Term Memory recurrent neural network"@en .

mlso-em:mean_absolute_error a semts:QualityMeasureConcept ;
skos:prefLabel "Mean Absolute Error"@en ;
rdfs:comment "Average of absolute differences between predictions and
    observations"@en .

# Main Time Series Segment

:time-interval a time:Interval ;
time:hasBeginning [
a time:Instant ;
time:inXSDDateTime "2023-01-01T00:00:00Z"^^xsd:dateTime
] ;
time:hasEnd [
a time:Instant ;
time:inXSDDateTime "2023-01-31T23:59:59Z"^^xsd:dateTime
] .

:wear-time-series a semts:TimeSeriesSegment ;
dct:title "Grinding Bead Wear Data" ;
dct:description "Hourly measurements of grinding bead wear in an industrial mill
    " ;
semts:segmentIndex :time-interval ;
semts:segmentDimension :dim-wear-rate, :dim-temperature ;
semts:segmentKnowledge :avg-wear-rate,
```

```
:wear-variability ,
:expert-observation ;
dcat:distribution :wear-ts-dist .

:wear-ts-dist a dcat:Distribution ;
dct:title "Grinding bead wear data (Jan 2023)" ;
dcat:accessURL <http://example.org/data/grinding-bead-wear-2023-01.csv> ;
dct:format "text/csv" .

:dim-wear-rate a semts:DataDimension ;
semts:title "Wear Rate" ;
semts:description "Measured wear rate of grinding beads in mm per hour" ;
semts:dimensionPosition "0"^^xsd:int ;
semts:hasUnit unit:MilliM-PER-HR ;
semts:hasDatatype qudt:DoublePrecisionType .

:dim-temperature a semts:DataDimension ;
semts:title "Temperature" ;
semts:description "Operating temperature of the mill in Celsius" ;
semts:dimensionPosition "1"^^xsd:int ;
semts:hasUnit unit:DEG_C ;
semts:hasDatatype qudt:DoublePrecisionType .

# Dedicated Refill Event Segment (Single Point in Time)

:refill-instant a time:Instant ;
time:inXSDDateTime "2023-01-15T14:00:00Z"^^xsd:dateTime .

:refill-segment a semts:TimeSeriesSegment ;
dct:title "Grinding Bead Refill Event" ;
dct:description "Single point event marking the refill of grinding beads" ;
semts:segmentIndex :refill-instant ;
semts:segmentDimension :dim-wear-rate, :dim-temperature ;
semts:segmentKnowledge :refill-event ;
semts:isPartOf :wear-time-series ;
dcat:distribution :refill-ts-dist .

:refill-ts-dist a dcat:Distribution ;
dct:title "Refill event record" ;
dcat:accessURL <http://example.org/data/grinding-bead-refill-2023-01-15T14.csv>
    ;
dct:format "text/csv" .

# Refill Event Knowledge Generation

:refill-logging a semts:KnowledgeReasoning ;
semts:title "Refill Event Logging" ;
semts:description "Manual logging of grinding bead refill event by operator" ;
semts:knowledgeGenerationConcept [
```

```
a semts:KnowledgeGenerationConcept ;
skos:prefLabel "Manual Event Entry"
] ;
semts:used :refill-input ;
semts:generated :refill-output .

:refill-input a semts:KnowledgeGenerationInput ;
semts:title "Input for Refill Event Logging" ;
semts:description "Refill segment context for event logging" ;
semts:inputDataReference :refill-segment .

:refill-output a semts:KnowledgeGenerationOutput ;
semts:title "Refill Event Output" ;
semts:description "Logged refill event information" ;
semts:outputKnowledge :refill-event ;
semts:wasDerivedFrom :refill-input .

:refill-event a semts:ScenarioKnowledge ;
semts:title "January Refill Event" ;
semts:description "Refill of 50kg new grinding beads at specific timestamp" ;
semts:knowledgeConcept semts-sv:CategorizedEvent ;
semts:knowledgeGenerationEntity :refill-logging ;
semts:hasValue [
a semts:EmbeddedValue ;
semts:valueString "50" ;
semts:hasDatatype qudt:DoublePrecisionType ;
semts:hasUnit unit:KiloGM
] .

# Statistical Analysis Method

:statistical-analysis a semts:KnowledgeGenerationMethod ;
semts:title "Statistical Analysis of Wear Data" ;
semts:description "Basic statistical analysis of grinding bead wear measurements
    " ;
semts:knowledgeGenerationConcept [
a semts:KnowledgeGenerationConcept ;
skos:prefLabel "TSFresh"
] ;
semts:used :stats-input ;
semts:generated :stats-output .

:stats-input a semts:KnowledgeGenerationInput ;
semts:title "Input for Statistical Analysis" ;
semts:description "Raw wear rate data for statistical analysis" ;
semts:inputDataReference :wear-time-series .

:stats-output a semts:KnowledgeGenerationOutput ;
semts:title "Statistical Analysis Results" ;
```

```
semts:description "Statistical measures derived from wear rate data" ;
semts:outputKnowledge :avg-wear-rate,
:wear-variability ;
semts:wasDerivedFrom :stats-input .

# Data Characteristics

:avg-wear-rate a semts:SegmentDataCharacteristic ;
semts:title "Average Wear Rate" ;
semts:description "Average wear rate of grinding beads over the entire period" ;
semts:knowledgeConcept semts-dv:MeanValue ;
semts:knowledgeGenerationEntity :statistical-analysis ;
semts:hasValue [
a semts:EmbeddedValue ;
semts:valueString "0.042" ;
semts:hasDatatype qudt:DoublePrecisionType ;
semts:hasUnit unit:MilliM-PER-HR
] .

:wear-variability a semts:SegmentDataCharacteristic ;
semts:title "Wear Rate Variability" ;
semts:description "Standard deviation of wear rate measurements" ;
semts:knowledgeConcept semts-dv:StandardDeviation ;
semts:knowledgeGenerationEntity :statistical-analysis ;
semts:hasValue [
a semts:EmbeddedValue ;
semts:valueString "0.008" ;
semts:hasDatatype qudt:DoublePrecisionType ;
semts:hasUnit unit:MilliM-PER-HR
] .

# Correlation Analysis + Multi-Segment Modeling

:correlation-analysis a semts:KnowledgeGenerationMethod ;
semts:title "Correlation Analysis Method" ;
semts:description "Method for calculating correlation between time series
    dimensions" ;
semts:knowledgeGenerationConcept [
a semts:KnowledgeGenerationConcept ;
skos:prefLabel "Correlation Analysis"
] ;
semts:used :corr-input ;
semts:generated :corr-output .

:corr-input a semts:KnowledgeGenerationInput ;
semts:title "Input for Correlation Analysis" ;
semts:description "Temperature and wear rate data for correlation analysis" ;
semts:inputDataReference :wear-time-series .
```

```
:corr-output a semts:KnowledgeGenerationOutput ;
semts:title "Correlation Analysis Results" ;
semts:description "Calculated correlation between temperature and wear rate" ;
semts:outputKnowledge :temp-wear-correlation ;
semts:wasDerivedFrom :corr-input .

:temp-wear-correlation a semts:SegmentRelation ;
semts:title "Temperature-Wear Correlation" ;
semts:description "Analysis of correlation between temperature and wear rate" ;
semts:knowledgeConcept semts-dv:Correlation ;
semts:knowledgeGenerationEntity :correlation-analysis ;
semts:hasValue [
a semts:EmbeddedValue ;
semts:valueString "0.72" ;
semts:hasDatatype qudt:DoublePrecisionType
] .

# Univariate segments participating in the multi-segment relation

:wear-segment a semts:TimeSeriesSegment ;
dct:title "Wear Rate Segment (Univariate)" ;
dct:description "Wear rate time series over Jan 2023" ;
semts:segmentIndex :time-interval ;
semts:segmentDimension :dim-wear-rate ;
semts:multiSegmentKnowledge :temp-wear-correlation ;
semts:isPartOf :wear-time-series ;
dcat:distribution :wear-segment-dist .

:wear-segment-dist a dcat:Distribution ;
dct:title "Univariate wear rate series (Jan 2023)" ;
dcat:accessURL <http://example.org/data/grinding-bead-wear-rate-2023-01.csv> ;
dct:format "text/csv" .

:temp-segment a semts:TimeSeriesSegment ;
dct:title "Temperature Segment (Univariate)" ;
dct:description "Temperature time series over Jan 2023" ;
semts:segmentIndex :time-interval ;
semts:segmentDimension :dim-temperature ;
semts:multiSegmentKnowledge :temp-wear-correlation ;
semts:isPartOf :wear-time-series ;
dcat:distribution :temp-segment-dist .

:temp-segment-dist a dcat:Distribution ;
dct:title "Univariate temperature series (Jan 2023)" ;
dcat:accessURL <http://example.org/data/grinding-bead-temperature-2023-01.csv> ;
dct:format "text/csv" .

# Forecast Method (Including Refill Event Knowledge)
```

```
:forecast-interval a time:Interval ;
time:hasBeginning [
a time:Instant ;
time:inXSDDateTime "2023-02-01T00:00:00Z"^^xsd:dateTime
] ;
time:hasEnd [
a time:Instant ;
time:inXSDDateTime "2023-02-28T23:59:59Z"^^xsd:dateTime
] .

:forecast-method a semts:KnowledgeGenerationMethod ;
semts:title "Wear Rate Forecast Method" ;
semts:description "Method for predicting future wear rates considering refill
    events" ;
semts:knowledgeGenerationConcept mlso-algo:LSTM ;
semts:used :forecast-input ;
semts:generated :forecast-output .

:forecast-input a semts:KnowledgeGenerationInput ;
semts:title "Input for Wear Forecast" ;
semts:description "Historical wear data and refill event information for
    generating forecast" ;
semts:inputDataReference :wear-time-series ;
semts:inputKnowledge :avg-wear-rate,
:wear-variability,
:refill-event .

:forecast-output a semts:KnowledgeGenerationOutput ;
semts:title "Wear Forecast Output" ;
semts:description "Forecasted wear rates for the next period considering refill
    impact" ;
semts:outputKnowledge :wear-forecast ;
semts:wasDerivedFrom :forecast-input .

:forecast-segment a semts:ArtificialTimeSeriesSegment ;
dct:title "Forecast Segment" ;
dct:description "Predicted wear values for February 2023" ;
semts:segmentIndex :forecast-interval ;
semts:segmentDimension :dim-wear-rate ;
semts:segmentKnowledge :wear-forecast ;
dcat:distribution :forecast-ts-dist .

:forecast-ts-dist a dcat:Distribution ;
dct:title "Forecasted wear rates (Feb 2023)" ;
dcat:accessURL <http://example.org/data/grinding-bead-wear-forecast-2023-02.csv>
    ;
dct:format "text/csv" .

:wear-forecast a semts:SegmentPrediction ;
```

```
semts:title "Wear Rate Forecast" ;
semts:description "Prediction of wear rate for February 2023 accounting for
    refill event" ;
semts:knowledgeConcept semts-dv:Forecast ;
semts:knowledgeGenerationEntity :forecast-method ;
semts:knowledgeQualityMeasure :forecast-accuracy ;
semts:hasValue [
a semts:ReferencedValue ;
dct:title "Forecasted wear rates for February 2023" ;
dcat:distribution :forecast-ts-dist ;
semts:hasDatatype qudt:DoublePrecisionType ;
semts:hasUnit unit:MilliM-PER-HR
] .

:forecast-accuracy a semts:KnowledgeQualityMeasure ;
semts:title "Forecast Accuracy" ;
semts:description "Mean Absolute Error of the forecast" ;
semts:qualityMeasureConcept mlso-em:mean_absolute_error ;
semts:hasValue [
a semts:EmbeddedValue ;
semts:valueString "0.005" ;
semts:hasDatatype qudt:DoublePrecisionType ;
semts:hasUnit unit:MilliM-PER-HR
] .

# Expert Observation Knowledge Generation

:expert-reasoning a semts:KnowledgeReasoning ;
semts:title "Domain Expert Analysis" ;
semts:description "Expert analysis of grinding bead wear patterns based on
    operational experience" ;
semts:knowledgeGenerationConcept [
a semts:KnowledgeGenerationConcept ;
skos:prefLabel "Expert Observation"
] ;
semts:used :expert-input ;
semts:generated :expert-output .

:expert-input a semts:KnowledgeGenerationInput ;
semts:title "Input for Expert Analysis" ;
semts:description "Wear time series data reviewed by domain expert" ;
semts:inputDataReference :wear-time-series .

:expert-output a semts:KnowledgeGenerationOutput ;
semts:title "Expert Analysis Output" ;
semts:description "Domain expert observation on wear patterns" ;
semts:outputKnowledge :expert-observation ;
semts:wasDerivedFrom :expert-input .
```

```
:expert-observation a semts:ScenarioKnowledge ;
semts:title "Expert Observation on Wear Patterns" ;
semts:description "Observation from domain expert on grinding bead wear patterns
    " ;
semts:knowledgeConcept semts-sv:Observation ;
semts:knowledgeGenerationEntity :expert-reasoning ;
semts:hasValue [
a semts:EmbeddedValue ;
semts:valueString "Wear rate increases non-linearly when temperature exceeds 60
    degree celsius" ;
semts:hasDatatype qudt:StringEncodingType
] .
```

Blow Molding Example

```
@prefix : <http://example.org/blowmolding#> .
@prefix semts: <https://w3id.org/semts/ontology#> .
@prefix semts-dv: <https://w3id.org/semts/vocabulary/data-knowledge#> .
@prefix semts-sv: <https://w3id.org/semts/vocabulary/scenario-knowledge#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix qudt: <http://qudt.org/schema/qudt/> .

# Distributions

:BlowMoldingDataDistribution a dcat:Distribution ;
dcat:accessURL <file:///data/blowmolding_sensor_data.parquet> ;
dct:format "application/parquet" .

:SegmentationDistribution a dcat:Distribution ;
dcat:accessURL <file:///data/segmentation.csv> ;
dct:format "text/csv" .

:TrainedModelDistribution a dcat:Distribution ;
dcat:accessURL <file:///models/transformer_autoencoder.pt> ;
dct:format "application/octet-stream" .

# Time Indices

:MainSeriesInterval a time:Interval ;
time:hasBeginning :MainSeriesStart ;
time:hasEnd :MainSeriesEnd .

:MainSeriesStart a time:Instant ;
time:inXSDDateTime "2024-01-01T00:00:00"^^xsd:dateTime .

:MainSeriesEnd a time:Instant ;
time:inXSDDateTime "2024-12-31T23:59:59"^^xsd:dateTime .

:Cycle1Interval a time:Interval ;
time:hasBeginning :Cycle1Start ;
time:hasEnd :Cycle1End .

:Cycle1Start a time:Instant ;
time:inXSDDateTime "2024-01-01T00:00:00"^^xsd:dateTime .

:Cycle1End a time:Instant ;
time:inXSDDateTime "2024-01-01T00:05:30"^^xsd:dateTime .
```

```
:Cycle2Interval a time:Interval ;
time:hasBeginning :Cycle2Start ;
time:hasEnd :Cycle2End .

:Cycle2Start a time:Instant ;
time:inXSDDateTime "2024-01-01T00:05:31"^^xsd:dateTime .

:Cycle2End a time:Instant ;
time:inXSDDateTime "2024-01-01T00:11:00"^^xsd:dateTime .

# Data Dimensions

:DimTemperature a semts:DataDimension ;
semts:title "Temperature" ;
semts:dimensionPosition "0"^^xsd:int .

:DimPressure a semts:DataDimension ;
semts:title "Pressure" ;
semts:dimensionPosition "1"^^xsd:int .

:DimCycleStartSignal a semts:DataDimension ;
semts:title "Cycle Start Binary Signal" ;
semts:description "Binary signal indicating cycle start condition" ;
semts:dimensionPosition "2"^^xsd:int .

:DimCycleEndSignal a semts:DataDimension ;
semts:title "Cycle End Binary Signal" ;
semts:description "Binary signal indicating cycle end condition" ;
semts:dimensionPosition "3"^^xsd:int .

:DimVibration a semts:DataDimension ;
semts:title "Vibration" ;
semts:dimensionPosition "4"^^xsd:int .

# Main Time Series Segment (Multivariate)

:BlowMoldingSensorData a semts:TimeSeriesSegment ;
dct:title "Blow Molding Machine Sensor Data" ;
dct:description "Multivariate time series from blow molding machine sensors" ;
semts:segmentIndex :MainSeriesInterval ;
semts:segmentDimension :DimTemperature, :DimPressure, :DimCycleStartSignal,
:DimCycleEndSignal, :DimVibration ;
dcat:distribution :BlowMoldingDataDistribution ;
semts:segmentKnowledge :CycleDefinitionKnowledge .

# Scenario Knowledge - Cycle Definition via Binary Signals

:CycleDefinitionValue a semts:EmbeddedValue ;
```

```
semts:valueString "(CycleStartSignal == 1) AND (CycleEndSignal == 0) marks cycle
     start; (CycleStartSignal == 0) AND (CycleEndSignal == 1) marks cycle end" .

:CycleDefinitionKnowledge a semts:ScenarioKnowledge ;
semts:title "Machine Cycle Definition" ;
semts:description "Defines how the combination of two binary signals indicates
    the start and end of a machine cycle" ;
semts:knowledgeConcept semts-sv:Condition ;
semts:hasValue :CycleDefinitionValue ;
semts:knowledgeGenerationEntity :ExpertAnnotationEntity .

# Knowledge Generation - Expert Annotation (for Scenario Knowledge)

:ExpertAnnotationConcept a semts:KnowledgeGenerationConcept ;
skos:prefLabel "Expert Annotation" .

:ExpertAnnotationEntity a semts:KnowledgeReasoning ;
semts:title "Domain Expert Annotation" ;
semts:description "Expert-provided knowledge about cycle definition" ;
semts:knowledgeGenerationConcept :ExpertAnnotationConcept ;
semts:used :ExpertAnnotationInput ;
semts:generated :ExpertAnnotationOutput .

:ExpertAnnotationInput a semts:KnowledgeGenerationInput ;
semts:inputDataReference :BlowMoldingSensorData .

:ExpertAnnotationOutput a semts:KnowledgeGenerationOutput ;
semts:outputKnowledge :CycleDefinitionKnowledge ;
semts:wasDerivedFrom :ExpertAnnotationInput .

# Knowledge Generation - Segmentation Method

:SegmentationConcept a semts:KnowledgeGenerationConcept ;
skos:prefLabel "Time Series Segmentation" .

:SegmentationMethod a semts:KnowledgeGenerationMethod ;
semts:title "Cycle-Based Segmentation" ;
semts:description "Segments the multivariate time series into machine cycles
    based on binary signal conditions" ;
semts:knowledgeGenerationConcept :SegmentationConcept ;
semts:used :SegmentationInput ;
semts:generated :SegmentationOutput .

:SegmentationInput a semts:KnowledgeGenerationInput ;
semts:title "Segmentation Input" ;
semts:inputDataReference :BlowMoldingSensorData ;
semts:inputKnowledge :CycleDefinitionKnowledge .

:SegmentationOutput a semts:KnowledgeGenerationOutput ;
```

```
semts:title "Segmentation Output" ;
semts:outputKnowledge :SegmentationResult ;
semts:wasDerivedFrom :SegmentationInput .

# Segmentation Result (Single Knowledge Instance)

:ArrayDatatype a qudt:Datatype ;
rdfs:label "Array" .

:SegmentationResultValue a semts:ReferencedValue ;
dcat:distribution :SegmentationDistribution ;
semts:hasDatatype :ArrayDatatype .

:SegmentationResult a semts:DataKnowledge ;
semts:title "Segmentation Result" ;
semts:description "Array of time tuples defining machine cycle segment
    boundaries derived from binary signal conditions" ;
semts:knowledgeConcept semts-dv:Segmentation ;
semts:hasValue :SegmentationResultValue ;
semts:knowledgeGenerationEntity :SegmentationMethod .

# Cycle Segments (referencing Segmentation Result)

:CycleSegment1 a semts:TimeSeriesSegment ;
dct:title "Machine Cycle 1" ;
dct:description "First machine cycle segment" ;
semts:segmentIndex :Cycle1Interval ;
semts:segmentDimension :DimTemperature, :DimPressure, :DimCycleStartSignal,
:DimCycleEndSignal, :DimVibration ;
dcat:distribution :BlowMoldingDataDistribution ;
semts:isPartOf :BlowMoldingSensorData ;
semts:multiSegmentKnowledge :SegmentationResult .

:CycleSegment2 a semts:TimeSeriesSegment ;
dct:title "Machine Cycle 2" ;
dct:description "Second machine cycle segment" ;
semts:segmentIndex :Cycle2Interval ;
semts:segmentDimension :DimTemperature, :DimPressure, :DimCycleStartSignal,
:DimCycleEndSignal, :DimVibration ;
dcat:distribution :BlowMoldingDataDistribution ;
semts:isPartOf :BlowMoldingSensorData ;
semts:multiSegmentKnowledge :SegmentationResult .

# Knowledge Generation

:AutoencoderTraining a semts:KnowledgeGenerationMethod ;
semts:title "Transformer Autoencoder Model Training" ;
semts:description "Trains a transformer autoencoder on cycle segments for
    representation learning" ;
```

```
semts:knowledgeGenerationConcept mlso-algo:Autoencoders ;
semts:used :TrainingInput1, :TrainingInput2 ;
semts:generated :TrainingOutput .

:TrainingInput1 a semts:KnowledgeGenerationInput ;
semts:title "Training Input (Cycle 1)" ;
semts:description "First cycle segment used for training the transformer
    autoencoder" ;
semts:inputDataReference :CycleSegment1 ;
semts:inputKnowledge :SegmentationResult .

:TrainingInput2 a semts:KnowledgeGenerationInput ;
semts:title "Training Input (Cycle 2)" ;
semts:description "Second cycle segment used for training the transformer
    autoencoder" ;
semts:inputDataReference :CycleSegment2 ;
semts:inputKnowledge :SegmentationResult .

:TrainingOutput a semts:KnowledgeGenerationOutput ;
semts:title "Training Output" ;
semts:description "Trained autoencoder model" ;
semts:outputKnowledge :TrainedAutoencoder ;
semts:wasDerivedFrom :TrainingInput1, :TrainingInput2 .

# Trained Model Instance

:TrainedModelValue a semts:ReferencedValue ;
dcat:distribution :TrainedModelDistribution .

:TrainedAutoencoder a semts:ModelInstance ;
semts:title "Trained Transformer Autoencoder" ;
semts:description "Transformer autoencoder model trained on machine cycle
    segments" ;
semts:knowledgeConcept semts-dv:ModelInstance ;
semts:hasValue :TrainedModelValue ;
semts:knowledgeGenerationEntity :AutoencoderTraining .

:CycleSegment1 semts:multiSegmentKnowledge :TrainedAutoencoder .
:CycleSegment2 semts:multiSegmentKnowledge :TrainedAutoencoder .

# Knowledge Generation - Threshold Determination

:ThresholdDeterminationConcept a semts:KnowledgeGenerationConcept ;
skos:prefLabel "Anomaly Detection Threshold Determination" .

:ThresholdDeterminationMethod a semts:KnowledgeGenerationMethod ;
semts:title "Threshold Determination for Anomaly Detection" ;
semts:description "Determines an optimal threshold value using the trained model
     for later anomaly detection" ;
```

```
semts:knowledgeGenerationConcept :ThresholdDeterminationConcept ;
semts:used :ThresholdInput ;
semts:generated :ThresholdOutput .

:ThresholdInput a semts:KnowledgeGenerationInput ;
semts:title "Threshold Determination Input" ;
semts:description "Trained model used to determine anomaly detection threshold"
    ;
semts:inputDataReference :CycleSegment1 ;
semts:inputKnowledge :TrainedAutoencoder .

:ThresholdOutput a semts:KnowledgeGenerationOutput ;
semts:title "Threshold Determination Output" ;
semts:outputKnowledge :AnomalyDetectionThreshold ;
semts:wasDerivedFrom :ThresholdInput .

# Threshold Knowledge (for Anomaly Detection)

:ThresholdValue a semts:EmbeddedValue ;
semts:valueString "0.85" .

:AnomalyDetectionThreshold a semts:ScenarioKnowledge ;
semts:title "Anomaly Detection Threshold" ;
semts:description "Reconstruction error threshold determined from the trained
    transformer autoencoder for anomaly detection" ;
semts:knowledgeConcept semts-sv:Threshold ;
semts:hasValue :ThresholdValue ;
semts:knowledgeGenerationEntity :ThresholdDeterminationMethod .

:BlowMoldingSensorData semts:segmentKnowledge :AnomalyDetectionThreshold .
```

E Knowledge Propagation Algebra: Formal Validation Sketch

This appendix provides sketch-level[1] confirmation of the correctness claims made in Section 5.4.1 for the knowledge propagation algebra. We establish three guarantees:

- **Termination:** every propagation expression reduces to a result in finitely many steps; no rewrite sequence runs forever.
- **Confluence:** the result of reduction is unique (up to the set laws), independent of the order in which rules are applied.
- **Soundness and Completeness:** on its computational fragment, the rewrite system produces exactly the knowledge states permitted by the axioms of Section 5.4.1, and every valid knowledge state is reachable.

Throughout, we work in the standard framework of *rewriting modulo equations*: the rewrite system $\mathcal{R}$ operates modulo the equational theory E_{AC}, consisting of associativity and commutativity of set union. The remaining set identities, which are the unit law $K \cup \varnothing = K$ and idempotence $K \cup K = K$, are *collapse equations*. They cannot soundly be placed in the modulo part when termination is established with AC-compatible path orderings, and they are therefore *oriented as rewrite rules* (R0a and R0b below). This combination yields exactly the intended behaviour of finite sets: the order of elements is irrelevant (modulo AC), while empty summands and duplicates are actively normalised away. It directly mirrors RDF triple stores, where duplicate triples collapse and ordering is absent by design. In the Maude mechanisation, set union is declared

[1] The formal proof sketches presented in Appendix E were developed with the assistance of AI-based language models (cf. Declaration of Generative AI). Results were subsequently tested mechanically using the Maude rewriting engine. Note that they only constitute proof sketches.

with the `assoc` and `comm` attributes together with `id:` $\varnothing$, i.e., built-in matching modulo ACU, which realises R0a, and idempotence (R0b) is added as an explicit equation. This is equationally equivalent to the R/E_{AC} presentation analysed below.

Formally, $t \to_{R/E_{\mathrm{AC}}} t'$ holds iff there exist terms $t_0 =_{E_{\mathrm{AC}}} t$ and $t'_0 =_{E_{\mathrm{AC}}} t'$ such that $t_0 \to_R t'_0$ by an ordinary rewrite step. All statements about termination, normal forms, and confluence are to be read modulo E_{AC}.

Rewrite System and Equational Theory

The equational theory in the modulo part consists of associativity and commutativity of union:

$$E_{\mathrm{AC}}: \qquad (K_1 \cup K_2) \cup K_3 \;=\; K_1 \cup (K_2 \cup K_3), \qquad K_1 \cup K_2 \;=\; K_2 \cup K_1.$$

The rewrite system $\mathcal{R}$ orients the set-normalisation laws, the accessor equations of Axiom A1, the hierarchy and scope equations of Axiom A3, the feeding axioms A5/A6, and the provenance accessors:

(R0a) $K \cup \varnothing \;\to\; K$

(R0b) $K \cup K \;\to\; K$

(R1) $\mathrm{inSeg}(\mathrm{mkIn}(s, K)) \;\to\; s$

(R2) $\mathrm{inKnow}(\mathrm{mkIn}(s, K)) \;\to\; K$

(R3) $\mathrm{outKnow}(\mathrm{mkOut}(r, K)) \;\to\; K$

(R4) $\mathrm{members}(\mathrm{mkGroup}(M)) \;\to\; M$

(R5) $\mathrm{feed}(O, \mathrm{mkIn}(s, K)) \;\to\; \mathrm{mkIn}\big(s,\; K \cup \mathrm{filterScope}(s, \mathrm{outK}(O))\big)$

(R6) $\mathrm{feedFlat}(O, \mathrm{mkIn}(s, K)) \;\to\; \mathrm{mkIn}\big(s,\; K \cup \{\, a \in \mathrm{flatten}(\mathrm{outK}(O)) \mid s \in \mathrm{scopeOf}(a) \,\}\big)$

(R7a) $\mathrm{feed}(\varnothing, i) \;\to\; i$

(R7b) $\mathrm{feedFlat}(\varnothing, i) \;\to\; i$

(R8) $\mathrm{atomsOf}(a) \;\to\; \{a\} \qquad (a \in \mathrm{KAtom})$

(R9) $\mathrm{atomsOf}(\mathrm{mkGroup}(M)) \;\to\; \bigcup_{x \in M} \mathrm{atomsOf}(x)$

(R10) $\mathrm{flatten}(K) \;\to\; \bigcup_{x \in K} \mathrm{atomsOf}(x)$

(R11) $\text{outK}(O) \;\to\; \bigcup_{o \in O} \text{outKnow}(o)$

(R12) $\text{filterScope}(s, K) \;\to\; \{\, k \in K \mid s \in \text{scopeOf}(k) \,\}$

(R13) $\text{scopeOf}(\text{mkGroup}(M)) \;\to\; \bigcup_{x \in M} \text{scopeOf}(x)$

(R14) $\text{runOf}(\text{mkOut}(r, K)) \;\to\; r$

(R15) $\text{used}(\text{mkRun}(e, I, n)) \;\to\; I$

(R16) $\text{derivedFrom}(\text{mkOut}(r, K)) \;\to\; \text{used}(r)$

Remarks.

- *Constructor patterns.* R5 and R6 apply only when the second argument is in constructor form $\text{mkIn}(s, K)$; by the no-junk part of Axiom A1 every ground KGI normalises to this form, so the rules are exhaustive on ground data.
- *Set normalisation.* R0a and R0b have the AC symbol $\cup$ at the root. Following Peterson–Stickel, their AC-extensions (e.g., $K \cup K \cup Z \to K \cup Z$) are implicitly included so that rewriting modulo AC is coherent (see, e.g., [BN98]). In Maude, R0a is realised by the `id:` attribute and R0b by an explicit equation.
- *Two-level status of* scopeOf. On atoms, scopeOf is interpretation-given: each concrete SemTS instance supplies a finite table of ground equations $\text{scopeOf}(a) = S_a$ with S_a a constructor set term. On groups, scopeOf is *defined* by R13 (structural recursion through members, exactly like R9). The two cases are pattern-disjoint, since KAtom and KGroup are disjoint sorts.
- *Comprehensions and big unions as schemas.* The right-hand sides of R6 and R9–R13 use big-union and set-comprehension notation; these are rule *schemas*. In the executable specification they are implemented by structural recursion over the set constructors $\varnothing$, singleton, and $\cup$ (with AC matching); the resulting first-order equations are orientable and overlap-free by the same arguments given below and are covered by the Maude validation (Section 5.4.1).
- *No equations on* feed/feedFlat. Neither operator is declared associative, commutative, or idempotent; all such properties (Proposition P1) are *derived* from R5–R7 together with the set laws.
- *Derived abbreviations.* propagate and propagateFlat are definitional abbreviations

involving the loose symbol eval; they are expanded by definition, not by $\mathcal{R}$. segs is likewise a definitional abbreviation; no axiom or rule depends on it, but it could be oriented as $\text{segs}(I) \to \bigcup_{i \in I}\{\text{inSeg}(i)\}$ (a homomorphism analogous to R11, with segs $\succ$ inSeg) without affecting any result.

- *Provenance.* R16 realises the *coarsest* conforming choice of derivation traceability, $\text{derivedFrom}(o) = \text{used}(\text{runOf}(o))$. An interpretation may instead supply finer-grained ground data satisfying $\varnothing \neq \text{derivedFrom}(o) \subseteq \text{used}(\text{runOf}(o))$; this replaces R16 by a finite table of ground equations and affects none of the results below.

Well-Formedness Invariants

Lemma E.1 (Acyclicity of groups)**.** *In the initial (term-generated) model, the membership relation on knowledge groups is acyclic: no group is reachable from itself by iterated* members*, i.e., $\neg(g \in^{+} g)$ for every group g (Axiom A2).*

Proof. Define the rank of a ground constructor term of sort Know by

$$\text{rank}(a) = 0 \;\; (a \in \text{KAtom}), \qquad \text{rank}(\text{mkGroup}(M)) = 1 + \max_{x \in M} \text{rank}(x),$$

which is well-defined because every ground constructor term is finite and every group is nonempty. The set laws (E_{AC}, R0a, R0b) only normalise the set notation inside M and neither add nor remove members, so rank is invariant on $=_{E_{\text{AC}}}$-classes and under set normalisation. If $x \in \text{members}(g)$ then $\text{rank}(x) < \text{rank}(g)$; hence rank strictly decreases along the transitive closure $\in^{+}$. A cycle $g \in^{+} g$ would yield $\text{rank}(g) < \text{rank}(g)$, a contradiction. □

Acyclicity is thus *structural*: it holds for every term that can be built at all, and does not need to be checked at runtime. Group extensionality is exactly the injectivity of mkGroup and is part of Axiom A1.

Termination

We show that $\to_{R/E_{\text{AC}}}$ admits no infinite reduction sequence. The proof uses a recursive path ordering compatible with the AC axioms of union (AC-RPO, [Rub02]), induced by a precedence on the function symbols.

Definition E.1 (Operator precedence)**.** *Let* $\succ$ *be the strict total precedence*

$$\text{feed} \succ \text{feedFlat} \succ \text{flatten} \succ \text{atomsOf} \succ \text{filterScope} \succ \text{scopeOf} \succ \text{outK}$$
$$\succ \text{outKnow} \succ \text{derivedFrom} \succ \text{used} \succ \text{runOf} \succ \text{inSeg} \succ \text{inKnow} \succ \text{members},$$

*with all data constructors (*mkIn, mkOut, mkGroup, mkRun, $\cup$, $\varnothing$, *singleton) ranked strictly below all defined symbols. On atoms,* scopeOf *is interpretation-given ground data and contributes only ground constructor right-hand sides; it participates in the precedence as a defined symbol through its group case (R13). The function* kout *of Axiom A4 is interpretation-given, has no rewrite rules, and does not participate.*

Theorem E.2 (Termination)**.** *The relation* $\to_{R/E_{\mathrm{AC}}}$ *is terminating on ground terms.*

Proof. Let $\succ_{\mathrm{rpo}}$ be the AC-RPO induced by the precedence of Definition E.1. It suffices to show $\ell \succ_{\mathrm{rpo}} r$ for every rule $\ell \to r$ of $\mathcal{R}$ (including the AC-extensions of R0a/R0b) and for every ground instance of the interpretation-given scope table; AC-RPO is then a reduction ordering compatible with E_{AC} [Rub02], which rules out infinite reductions modulo AC.

Set normalisation (R0a, R0b). In both rules the right-hand side K is a proper subterm of the left-hand side, so orientation follows from the subterm property. The AC-extensions $K \cup \varnothing \cup Z \to K \cup Z$ and $K \cup K \cup Z \to K \cup Z$ are likewise decreasing: under AC-RPO, terms rooted in the same AC symbol are compared by the multiset extension on their flattened argument lists, and the right-hand multisets $\{K, Z\}$ are proper sub-multisets of $\{K, \varnothing, Z\}$ and $\{K, K, Z\}$, respectively.

Accessors (R1–R4, R14, R15). Each right-hand side (s, K, M, r, I) is a proper subterm of the left-hand side; orientation follows from the subterm property.

Propagation (R5). The left-hand side is rooted in feed, the maximal symbol. The right-hand side is rooted in the constructor mkIn, which is below feed, so it suffices that $\mathrm{feed}(O, \mathrm{mkIn}(s, K)) \succ_{\mathrm{rpo}} u$ for every argument u of the right-hand side. The arguments are s, K (proper subterms) and $\mathrm{filterScope}(s, \mathrm{outK}(O))$, whose defined symbols filterScope and outK are below feed in the precedence and whose arguments s, O are proper subterms of the left-hand side. Hence $\ell \succ_{\mathrm{rpo}} r$.

Flat propagation (R6). Analogous to R5: the right-hand side mentions the defined symbols flatten, outK, and through the comprehension, scopeOf, all of which are strictly

below feedFlat, applied to proper subterms of the left-hand side. (Note that scopeOf is a defined symbol; the required inequality feedFlat $\succ$ scopeOf holds by Definition E.1.)

Empty batches (R7a, R7b). The right-hand side i is a proper subterm; subterm property.

Hierarchy (R8–R10) and group scope (R13). R8 replaces a defined symbol applied to an atom by a constructor term built from a proper subterm. R9 and R13 are structural recursions: every recursive call atomsOf(x), resp. scopeOf(x), is applied to a member $x \in M$, a proper subterm of mkGroup(M), and $\cup$ is a constructor below the defined symbol; under AC-RPO each instance of the (schematic) right-hand side is strictly smaller. R10 is the same argument one level up, with flatten $\succ$ atomsOf.

Output collection (R11) and scope filtering (R12). R11 uses outK $\succ$ outKnow with each outKnow(o) applied to an element $o \in O$, a proper subterm. R12 uses filterScope $\succ$ scopeOf; the membership tests $s \in$ scopeOf(k) apply scopeOf to elements $k \in K$, proper subterms of the left-hand side.

Provenance (R16). derivedFrom $\succ$ used by the precedence, and the argument r is a proper subterm of the left-hand side.

Ground scope table. Each interpretation-given equation scopeOf(a) $\to S_a$ replaces a defined symbol by a ground constructor term; since every defined symbol exceeds every constructor in the precedence, these instances are decreasing as well.

The argument is based on the precedence and the subterm relation and is in particular independent of the semantic acyclicity established in Lemma E.1. All rules being strictly decreasing under an AC-compatible reduction ordering, $\to_{R/E_{\mathrm{AC}}}$ terminates. □

Confluence

Theorem E.3 (Confluence)**.** *The relation $\to_{R/E_{\mathrm{AC}}}$ is confluent on ground terms. Together with Theorem E.2, every ground term of the computational fragment has a unique normal form modulo AC.*

Proof. By Theorem E.2 the system terminates, so by Newman's lemma it suffices to establish local confluence, which for rewriting modulo AC reduces to the joinability of all critical pairs of $\mathcal{R}$ (including the AC-extensions of R0a/R0b) modulo E_{AC} [BN98].

Set normalisation (R0a, R0b). These are the only rules whose left-hand sides are rooted in the AC symbol $\cup$. Their critical pairs are the standard ones of set normalisation and are joinable. Representative cases: $(K \cup K) \cup K$ reduces by either application of R0b to $K \cup K$ and then to K; $\varnothing \cup \varnothing$ reduces to $\varnothing$ by R0a and by R0b alike; $(K \cup K) \cup \varnothing$ reduces via R0b to $K \cup \varnothing$ and then via R0a to K, and via R0a to $K \cup K$ and then via R0b to K. The resulting normal forms are the duplicate-free, $\varnothing$-free unions (canonical representations of finite sets modulo AC).

Overlap 1: R5 versus R7a. Both rules apply to $\mathrm{feed}(\varnothing, \mathrm{mkIn}(s, K))$. Via R7a the term reduces to $\mathrm{mkIn}(s, K)$. Via R5 it reduces to $\mathrm{mkIn}(s, K \cup \mathrm{filterScope}(s, \mathrm{outK}(\varnothing)))$; R11 reduces $\mathrm{outK}(\varnothing)$ to the empty union $\varnothing$, R12 reduces $\mathrm{filterScope}(s, \varnothing)$ to $\varnothing$, and R0a reduces $K \cup \varnothing$ to K. Both paths join at $\mathrm{mkIn}(s, K)$.

Overlap 2: R6 versus R7b. Identical to Overlap 1 with $\mathrm{flatten}(\varnothing)$ reducing to the empty union via R10, the comprehension over the empty set yielding $\varnothing$, and R0a closing the join.

Nested reduction 1 (iterated feeding). For $\mathrm{feed}(O_2, \mathrm{feed}(O_1, \mathrm{mkIn}(s, K)))$, the inner redex must be contracted before R5 matches the outer feed; the result is $\mathrm{mkIn}(s, (K \cup F_1) \cup F_2)$ with $F_j = \mathrm{filterScope}(s, \mathrm{outK}(O_j))$ normalised. Any alternative scheduling of the auxiliary reductions (R11, R12, set normalisation) yields the same term modulo AC, since union is compared modulo AC and the auxiliary redexes are at disjoint positions.

Nested reduction 2 (commutativity of batches). $\mathrm{feed}(O_1, \mathrm{feed}(O_2, i))$ and $\mathrm{feed}(O_2, \mathrm{feed}(O_1, i))$ normalise to $\mathrm{mkIn}(s, K \cup F_2 \cup F_1)$ and $\mathrm{mkIn}(s, K \cup F_1 \cup F_2)$, respectively, which are equal modulo AC.

Nested reduction 3 (idempotence of feeding). $\mathrm{feed}(O, \mathrm{feed}(O, \mathrm{mkIn}(s, K)))$ normalises to $\mathrm{mkIn}(s, K \cup F \cup F)$, which the AC-extension of R0b rewrites to $\mathrm{mkIn}(s, K \cup F)$, the normal form of a single feeding step. Idempotence thus enters as an explicit rewrite step (R0b), not as an identification in the modulo theory.

Flat analogues. The corresponding nested reductions for feedFlat are joinable by the same arguments, with R10/R13 in place of R11/R12 where applicable.

Remaining overlaps. All rules except R0a/R0b are left-linear and have pairwise distinct defined root symbols, so root overlaps between distinct defined symbols are impossible. R8 and R9 are sort-disjoint (KAtom versus KGroup); likewise, R13 is pattern-disjoint from the interpretation-given ground equations for scopeOf on atoms. R14, R15, and

R16 have unique constructor patterns and overlap with no other rule. Sequential dependencies - R11 producing outKnow-redexes for R3, and R16 producing a used-redex for R15 - create nested, position-disjoint redexes rather than superpositions and are therefore not critical. Since no rule other than R0a/R0b has $\cup$ at the root, all remaining interactions with the AC theory occur strictly below variable positions and induce no further critical pairs.

All critical pairs are joinable, so the system is locally confluent; with termination, it is confluent, and normal forms are unique modulo AC. □

Correspondence with the Set-Theoretic Semantics

The specification has two layers. The *computational fragment* $\Sigma_c \subseteq \Sigma$ consists of the data constructors (mkIn, mkOut, mkGroup, mkRun, $\cup$, $\varnothing$, singleton, the base activity constants, segment constants, and atom constants) together with the defined symbols of R0–R16; it carries *initial* semantics. The symbols eval and generated, the knowledge-output function kout, and scopeOf on atoms are *loosely specified* observers: they carry no rewrite rules, are supplied by the interpretation, and are constrained by Axiom A4. The accessor used has a rewrite rule only on run instances. On base activities it is likewise interpretation-given (cf. the remark on provenance operators in Section 5.4.1). The correspondence result below is accordingly stated for the computational fragment; this is not a weakness but the standard treatment of loose function symbols in algebraic specification.

Theorem E.4 (Soundness and completeness on the computational fragment)**.** *Let* $\mathcal{A}_{\text{set}}$ *be a set-theoretic algebra that interprets each sort by its intended carrier (*$\Phi(X)$ *by finite subsets,* KGI *by* $\text{Seg} \times \Phi(\text{Know})$*,* KGO *by* $\text{KGE} \times \Phi_+(\text{Know})$*,* KGroup *by* $\Phi_+(\text{Know})$ *modulo Lemma E.1, runs by* $\text{KGE} \times \Phi_+(\text{KGI}) \times \mathbb{N}$*), each constructor by the corresponding tupling or set operation, each defined symbol of R0–R16 by its set-theoretic counterpart, and the loose symbols by arbitrary functions satisfying Axiom A4. Then for every such conforming interpretation, the denotation map*

$$[\![-]\!] \;:\; T_{\Sigma_c}/(R \cup E_{\text{AC}}) \;\longrightarrow\; \mathcal{A}_{\text{set}}$$

restricted to constructor-reachable equivalence classes is a well-defined isomorphism onto the data part of $\mathcal{A}_{\text{set}}$*. Moreover, Axiom A4 is satisfiable, and every conforming interpretation satisfies Axioms A1–A6 and Proposition P1.*

Proof. (1) Well-definedness. By Theorems E.2 and E.3, every ground Σ_c-term has a unique normal form modulo AC. Every defined symbol of Σ_c applied within its constructor domain is eliminated by reduction: accessors by R1–R4, R14, R15; propagation by R5–R7; hierarchy and scope by R8–R10, R13 together with the ground scope table; output collection and filtering by R11–R12; derivation by R16; and set notation is canonicalised by R0a/R0b. Normal forms are therefore constructor terms with duplicate-free, $\varnothing$-free unions, on which the denotation is computed directly. Two kinds of terms are excluded. Terms containing the loose symbols eval or generated lie outside Σ_c. Applications of used to a base activity, and hence of derivedFrom to an output whose run argument is a base activity, match no rule. Their classes contain no constructor term and therefore fall outside the constructor-reachable fragment. Both kinds are interpreted directly rather than by normalisation which is why the theorem is restricted as stated.

(2) Injectivity. Distinct constructor normal forms denote distinct values, by induction on the term structure: segments, atoms, and base activities are pairwise distinct generators; finite sets in canonical form are equal iff they are equal modulo AC; mkIn, mkGroup are injective by A1; mkOut is compared componentwise on the generating run and the carried knowledge; and runs are separated by the injectivity of mkRun together with the distinctness of base activities from run terms.

(3) Surjectivity. Every element of the data part of $\mathcal{A}_{\text{set}}$ is the denotation of a constructor term: finite sets are built from $\varnothing$, singletons, and $\cup$; inputs, outputs, groups, and runs from their constructors. The no-junk clauses of A1 guarantee that the constructor-reachable subalgebra exhausts the constructed sorts; the restriction to it excludes junk by construction.

(4) Homomorphism and satisfiability of A4. Interpret evaluation by

$$\text{eval}^{\mathcal{A}}(e, I, n) \;:=\; \Big(\text{mkRun}(e, I, n),\; \big\{\, \text{mkOut}(\text{mkRun}(e, I, n), K) \;\big|\; K \in \text{kout}(e, I) \,\big\},\; n{+}1\Big).$$

For *every* choice of the ground function kout the postconditions of A4 hold: $\text{used}(e') = I$ by A1/R15; $\text{generated}(e') = O$ is well-defined because the run e' uniquely determines (e, I, n) and hence O; $\text{derivedFrom}(o) = \text{used}(\text{runOf}(o)) = I$ is a nonempty subset of I; freshness holds by the injectivity of mkRun and mkOut; and $\text{outK}(O) = \bigcup \text{kout}(e, I)$ depends only on (e, I) (knowledge-level determinism). In particular, the constant choice $\text{kout}(e, I) = \{\{k_0\}\}$ for a fixed atom k_0 yields a model, confirming that the axiom system

is non-vacuously satisfiable.

Remark (necessity of output identity). With a content-extensional output constructor $\mathrm{mkOut} : \Phi_+(\mathrm{Know}) \to \mathrm{KGO}$, outputs are identified by their knowledge alone. Then $I_1 = \{\mathrm{mkIn}(s, \varnothing)\}$ and $I_2 = \{\mathrm{mkIn}(s, \{k_0\})\}$ are disjoint input batches whose evaluations under the constant kout produce the *same* output term o; derivation traceability would force $\mathrm{derivedFrom}(o) \subseteq I_1 \cap I_2 = \varnothing$, a contradiction. Likewise, a run constructor depending only on (e, n) makes used over-determined. Run-indexed constructors are therefore a consistency requirement of the specification, not an implementation convenience; they mirror PROV, where entities are individuated by IRIs over and above their content.

(5) Remaining axioms. Under the denotation, A1 holds by construction of the carriers; A2 by Lemma E.1; A3 by R13 together with the ground scope table; A5 and A6 by R5–R7 and the set laws; and the laws of Proposition P1 reduce to identities of finite sets (unit, idempotence, commutativity, associativity of union, and distributivity of outK, flatten, and filterScope over unions), all of which hold in $\mathcal{A}_{\mathrm{set}}$ and are realised by R0–R16 modulo AC. □

Summary of Formal Guarantees

1. **Termination and canonical results.** Every propagation expression of the computational fragment reduces in finitely many steps to a unique normal form modulo AC; set notation is actively canonicalised (no duplicates, no empty summands) by the oriented rules R0a/R0b (Theorems E.2 and E.3).

2. **Order and batching independence.** Commutativity, idempotence, and batch decomposition of feeding (Proposition P1) are *derived* consequences of the rewrite rules and the set laws and not additional assumptions on feed or feedFlat.

3. **Structural acyclicity.** Knowledge hierarchies cannot contain cycles; this holds for every constructible term (Lemma E.1) and requires no runtime checks.

4. **Freshness and computable provenance.** Runs and outputs carry their full construction data (activity, input batch, counter), so distinct evaluations yield distinct run instances and distinct output entities even when the produced knowledge coincides; the provenance accessors used, runOf, and derivedFrom are total,

well-defined, and computable by rewriting (R14–R16), while output *knowledge* remains deterministic in the activity and its inputs (Axiom A4).

5. **Faithful set semantics.** On the computational fragment, the rewrite system is isomorphic to the intended set-theoretic semantics (Theorem E.4); the loosely specified symbols (eval, kout, generated, scopeOf on atoms) are constrained but not fixed by the axioms, so every conforming implementation realises the same data semantics.