Knowledge Reinjection Policies and Machine Learning Integration in CWM-Based Complex Data Warehouses

Fabrice Razafindraibe, Jean Christian Ralaivao, Angelo Raherinirina and Hasina Rakotonirainy
University of Fianarantsoa, Madagascar

Keywords: Common Warehouse Metamodel (CWM), Knowledge Reinjection, Data Warehouse Architecture, Ontology Integration, Machine Learning, Semantic Interoperability, Knowledge Management, Metadata Management, Knowledge Discovery, Ontological Reasoning.

Abstract: This article discusses the growing complexity of data warehousing systems and the need for enhanced frameworks that can effectively manage simultaneously metadata and knowledge. While the Common Warehouse Metamodel (CWM) provides a standardized method for metadata management, its semantic limitations hinder its use in complex environments. To overcome these shortcomings, the paper proposes an extended CWM framework that incorporates ontologies, machine learning, and knowledge re-injection policies. This new framework introduces additional layers and components, such as a 'learning package' and advanced knowledge mapping, to improve semantic interoperability, adaptability and usability. The research also explores the integration of hybrid AI systems that use both inductive and deductive methods to facilitate knowledge discovery and improve decision making.

1 INTRODUCTION AND RESEARCH QUESTIONS

The increasing complexity of data warehouse projects requires robust architectures capable of effectively managing and exploiting the knowledge embedded in data. Traditional data warehouse systems excel at handling structured data but struggle to meet the semantic depth and dynamic requirements of modern organizations. The Common Warehouse Metamodel (CWM) provides a standardized framework for metadata management, promoting interoperability and governance. However, its limited semantic expressiveness poses challenges in representing complex relationships and domain-specific rules, which are essential for advanced knowledge integration and decision-making.

Recent advances in ontology-based modeling and Artificial Intelligence (AI), particularly machine learning, have created new opportunities to improve knowledge discovery, representation, and utilization within data warehouses. Ontologies provide a structured approach to capturing domain knowledge and semantic relationships, while machine learning techniques facilitate adaptive, data-driven insights. By integrating these methodologies into CWM-based architectures, we aim to address long-standing challenges in metadata and knowledge management, such as heterogeneity, scalability, and performance optimization.

This paper addresses the challenge of integrating knowledge reinjection policies and machine learning into CWM-based data warehouses to enhance metadata management. To achieve this, we explore the following key research questions: 1. *How can the CWM framework be extended to improve semantic expressiveness and adaptability*? 2. *What methodologies best support the integration of machine learning for knowledge reinjection and metadata enhancement*?

To answer these questions, we propose an extended CWM framework incorporating ontologies, machine learning, and systematic knowledge reinjection policies. This new approach introduces additional layers and components—such as a Learning Package and advanced knowledge mapping mechanisms—to enhance semantic interoperability, adaptability, and usability. The research also explores the integration of hybrid AI systems, combining inductive and deductive learning methods to facilitate knowledge discovery and improve decision-making.

The rest of the paper is structured as follows: • Section 2 reviews the state of the art in knowledge integration, ontology-based systems, and machine learning applications in data warehouses. • Section 3 describes the proposed methodology and framework, detailing the CWM extension and knowledge reinjection policies. • Section 4 presents results.

By addressing these aspects, this work contributes

Razafindraibe, F., Ralaivao, J. C., Raherinirina, A. and Rakotonirainy, H.
Knowledge Reinjection Policies and Machine Learning Integration in CWM-Based Complex Data Warehouses.
DOI: 10.5220/0013352600003929

In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 1*, pages 301-308
ISBN: 978-989-758-749-8; ISSN: 2184-4992

to the evolution of knowledge-driven data warehouse architectures, enhancing their ability to manage complex and heterogeneous data.

2 STATE OF THE ART AND RELATED WORKS

2.1 Incorporation of Ontologies in Data Warehouse

Ontology-based modeling has gained significant attention as a means to enhance semantic expressiveness in data warehouses. Ontologies provide a structured representation of domain knowledge, enabling improved metadata management, semantic interoperability, and knowledge reasoning.

Several studies have explored ontology integration into data warehouses (Falquet et al., 2011; Olsina, 2021). These approaches aim to bridge the gap between structured data models and semantic knowledge representation. For instance, (Zayakin et al., 2021) proposed an ontology-based methodology for analytical platforms in data-intensive domains, emphasizing the traceability of metamodel evolution to maintain semantic relevance. Similarly, (Antunes et al., 2022) reviewed various ontology classifications applicable to data warehouses, highlighting their potential for enhancing data governance and metadata structuring.

However, despite their benefits, these ontology-based approaches often lack dynamic knowledge reinjection mechanisms. They primarily focus on static ontology integration, without addressing how newly acquired knowledge can be reinjected into metadata management systems dynamically. This limitation reduces their adaptability in environments requiring continuous knowledge updates.

2.2 Combining Machine Learning and Ontology

The integration of machine learning (ML) and ontologies is a growing area of research, aimed at enhancing knowledge discovery and automated reasoning. (Ghidalia et al., 2024) conducted a systematic review of hybrid AI systems combining inductive learning (ML-based inference) with deductive reasoning (ontology-driven approaches). Their work categorizes hybrid AI applications into: • **Learning-Enhanced Ontologies**: Using ML to automate ontology construction and maintenance. • **Semantic Data Mining**: Leveraging ontological knowledge to enhance ML-based data mining. • **Learning and Reasoning Systems**: Combining symbolic reasoning with ML to emulate human-like cognitive processes.

These studies demonstrate the potential of hybrid AI approaches for adaptive knowledge management. However, they do not provide a formalized method for metadata management in CWM-based architectures. Most existing works focus on either ML-driven knowledge extraction or ontology-based reasoning, but lack a comprehensive framework for continuous knowledge reinjection and metadata evolution within data warehouses.

2.3 CWM Extensions

The CWM is a widely adopted standard for metadata management in data warehouses, providing a structured framework for interoperability, governance, and data lineage tracking. While CWM offers a robust foundation, its limited semantic expressiveness hinders its application in complex knowledge-driven environments.

Several studies have proposed CWM extensions to enhance knowledge discovery and management (Gomes et al., ; Midouni et al., ; da Silva et al., ; Tavac and Tavac,). Among them, (Thavornun, 2015) introduced an RDF-based metadata management approach for knowledge discovery, extending the Core (Object layer) and Transformation and Data Mining (Analysis layer) packages with three key classes: **Evidence** – Extracted knowledge patterns; **UserAction** – User interactions for adaptive learning; **DataProperty** – Relationships between extracted knowledge elements.

Recent research has explored CWM extensions to incorporate ontology-based capabilities. For instance, (Ralaivao et al., 2024) proposed extending CWM with a new knowledge layer (2), introducing: 1. **Acquired and Explicit Knowledge** – Capturing technical knowledge and operational statistics. 2. **Domain Knowledge and Ontology** – Structuring specialized domain-specific metadata. 3. **Metadata and Knowledge Mapping** – Managing transformations and semantic linkages within CWM.

(Ralaivao et al., 2024) proposed **Core** package extension to enable compatibility with the ODM metamodel. The addition of the class **AnnotatedElement** is proposed, would inherit directly from **ModelElement** and serves as the parent class for **Ontology** and **OntologyElement**, as depicted in Figure 1. This proposed extension aims to enhance the CWM metamodel's capacity for knowledge representation and integration, particularly in data-intensive domains.

While these efforts enhance semantic metadata representation, they do not fully address the integra-

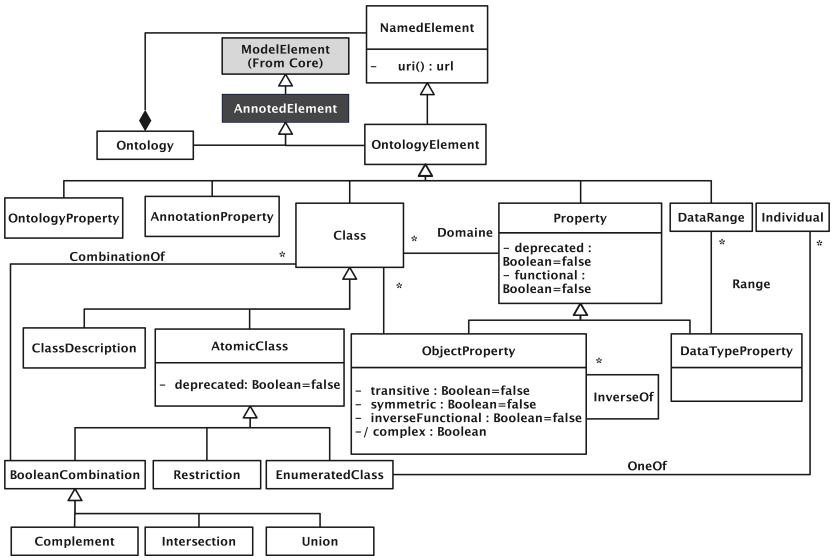

Figure 1: Elements structuring the ODM metamodel and extending **Core** Package.

tion of dynamic knowledge reinjection policies.

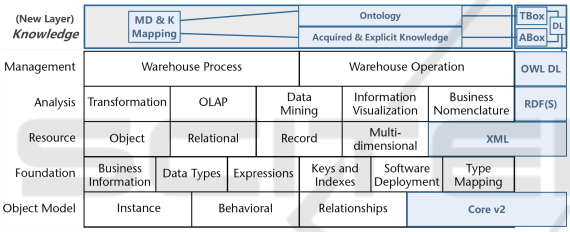

Figure 2: New layer integrated in CWM metamodel.

3 METHODOLOGY AND CONTRIBUTION

3.1 Research Methodology

This research addresses knowledge integration, ontology enhancement, and machine learning incorporation into CWM. While CWM supports classical data mining, it lacks dynamic knowledge reinjection and adaptive metadata management. Our approach follows three key steps:1. **Extending CWM** – Defining a structured knowledge reinjection policy to ensure continuous metadata evolution (Ralaivao et al., 2024). 2. **Integrating AI-driven Metadata Management** – Leveraging ontology-based machine learning for adaptive semantic enrichment. 3. **Evaluating the Framework** – Assessing performance through benchmarks and real-world use cases. Key challenges include metamodel complexity, heterogeneous data integration, and business-aligned reinjection policies. Future research should refine automated knowledge discovery and reinjection to enhance adaptability.

3.2 Contribution of this Paper

This paper extends CWM with a structured knowledge reinjection process and AI-driven enhancements, addressing limitations in existing ontology and machine learning-based approaches. • **AI-Enhanced CWM** – Introducing a Learning Package for adaptive metadata enrichment. • **Structured Knowledge Reinjection** – Ensuring continuous, validated knowledge updates. • **Optimized Query Processing** – Leveraging reinjected knowledge to improve semantic search and decision support.

This hybrid AI and ontology-based framework enables scalable, real-time metadata management with enhanced semantic expressiveness.

4 RESULTS

4.1 Knowledge Reinjection Policy

4.1.1 General Reinjection Framework

Unlike the traditional approach of simply extracting knowledge from data, this research aims to take the knowledge derived from the data, reclassify it, validate it and then recode it in the form of actionable constraints. These constraints are intended to be used at the data level, particularly in the form of enriched metadata, to optimise data management and decision-making processes.

For instance, in the healthcare domain, the reinjection process detects inconsistencies in patient data and updates metadata dynamically. Suppose an AI model identifies a new correlation between symptoms and diagnoses. The system classifies this as new knowledge, validates it with existing ontologies, and reinjects it into the metadata layer for future queries.

The proposed knowledge reinjection process involves several key strategies:

1. **Optimization of Query Structure** The characteristics of a variable—such as its probability distribution and parameters—can enhance query performance in a data warehouse. By leveraging these characteristics, you can refine the order in which predicates are applied within query selections. This strategic application can lead to significant improvements in query efficiency; 2. **Uncovering Hidden Functional Dependencies** The process also involves the discovery of non-explicit functional dependencies within large volumes of data. These dependencies, often overlooked, can be leveraged to optimize the design and organization of the data warehouse schema, ensuring a more efficient and effective structure.; 3. **Enriching Knowledge and Data Structures** The validated knowledge, once refined, becomes accessible to Complex Data Warehouse administrator. This knowledge can be used to: • **Enrich the Ontology**: Identify and add new concepts, uncover relationships between existing concepts, and challenge or refine existing ontological links; • **Reformulate Queries**: Use newly discovered knowledge to revise queries, incorporating newly identified concepts, more relevant calculation rules, and optimized query formulations; • **Reorganize Complex Data Warehouse Structures**: Introduce new hierarchies or reorganize schema dimensions, update CWM's "Business Nomenclature" layer, or develop new transformation and deduction rules to reflect new insights; • **Adjust Administration Parameters**: Optimize the administration of the Complex Data Warehouse, including adjusting ETL scheduling, process management, and aligning processes with the "Warehouse Process" and "Warehouse Operation" layers within CWM, ensuring smoother and more effective operations.

Once the knowledge has been validated, it is annotated and supported by the "**Knowledge**" layers of the extended CWM, in particular the "**MD & K Mapping**" package. Depending on the annotation attached to each piece of knowledge, the latter is responsible for transcribing it and distributing it to the various layers concerned (as new knowledge, obsolete knowledge or modified knowledge).

4.1.2 Reinjection Process

The knowledge reinjection process (Figure 3) follows a structured workflow to ensure that new knowledge is effectively classified, validated, and integrated into the system. It consists of the following steps: 1. **Knowledge Extraction**: Identify new patterns from the data warehouse using machine learning techniques. 2. **Validation**: Compare extracted knowledge with existing ontologies to ensure consistency. 3. **Metadata Update**: Integrate new insights into the metadata management layer. 4. **Query Optimization**: Adapt the query engine to leverage enriched metadata.

Each new knowledge entry is categorized as either confirmed, tentative, or rejected, ensuring robust control over metadata evolution.

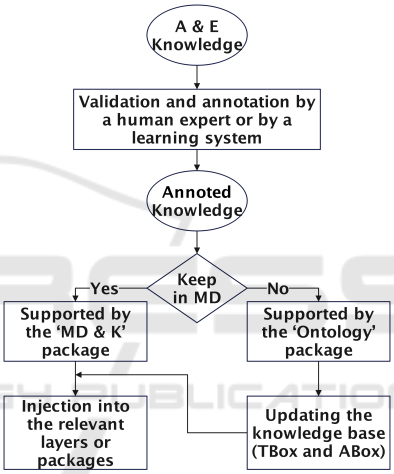

Figure 3: Knowledge reinjection procedure.

If it is decided to transform the knowledge into metadata, the '**MD & K**' layer is responsible for this transformation and for injecting it into the layer(s) concerned. If it is decided to keep it in the form of knowledge, the same '**MD & K**' layer is responsible for transmitting it to the '*Ontology*' layer. This in turn decides whether to update the **Tbox** or **Abox** bases of the ontology in DL. This then updates the layers concerned in the same way as the metadata. Note that in this second case, it is mainly the new layers resulting from the CWM extension that are most involved.

4.2 CWM Learning Integration

We propose to take the classifications defined by (Ghidalia et al., 2024) and transform them into the concept of classes for possible integration into the CWM metamodel. And we propose to extend the Analysis layer by adding a **Learning** package (Figure

4). This new package will work in conjunction with the **Knowledge** layer to improve knowledge management and governance.

To further clarify the role of machine learning within our extended CWM framework, we categorize its functions into three main strategies:

Table 1: ML Strategies for Knowledge Reinjection.

Strategy	Objective	Algorithms
Knowledge Discovery	Extract patterns and new insights from raw data.	Decision Trees, Random Forest, Autoencoders
Ontology Enrichment	Automatically update and refine ontology structures.	Word2Vec for semantic similarity, Graph Neural Networks
Reinjection Optimization	Decide when and how to reinject new knowledge.	Reinforcement Learning, Bayesian Networks

By formalizing these strategies, our framework ensures that machine learning contributes effectively to metadata enhancement and knowledge management.

4.2.1 The *MLModel* Component

MLModel represents the abstract concept unifying the three components and uses the **AnnotedElement** concept from extended Core Package. It defines common attributes like *reasoningType* and *knowledgeBase* and methods like **dataIntegration()**, **trainModel()**, and **infer()**.

Properties are characterised as follows: 1. **reasoningType** influences how knowledge is processed to derive new insights or verify existing information. It shapes the system's reasoning strategy, tailored to its specific application domain and desired outcomes. Common reasoning types can be : Deductive, Inductive, Abductive, Ontological or Hybrid reasoning. 2. **knowledgeBase** serves as the factual and logical foundation that enables the AI system to comprehend its domain and reach informed conclusions. It ensures the system has access to reliable and organized information. The components of a KnowledgeBase can be : $TBox$, $ABox$, Logical Rules or Data Sources.

The methods are designed as follows: 1. **dataIntegration()** method unifies and prepares disparate data sources to create a cohesive dataset for analysis and learning. It links raw data with ontological knowledge, ensuring semantic alignment and data consistency. 2. **trainModel()** method trains a machine learning model using integrated and pre-processed data. It uses domain-specific knowledge from ontologies to enhance the training process, ensuring that the model effectively learns patterns and relationships relevant to the target domain. 3. **infer()** method applies the trained model to new data, performing inference tasks by integrating knowledge from both the ontology and the learned model. It generates predictions and explanations based on data patterns and ontological rules.

Classes like **LearningEnhancedOntology**, **SemanticDataMining**, and **LearningAndReasoningSystem** are abstracted under **MLModel**, with relevant relationships to supervised, unsupervised, and neural techniques.

4.2.2 The *LearningEnhancedOntology* Component

The **LearningEnhancedOntology** component automates the extraction and classification of ontological concepts. For instance, in an e-commerce system, this module can dynamically classify new product categories based on customer interactions and reinject this knowledge into the search engine's metadata to enhance recommendations. It includes attributes like ontologyType and ruleset.

1. **ontologyType** defines the specific type of ontology used in the system. Examples include domain ontology, task ontology and parent ontology. This classification helps the system to understand the focus and scope of the ontology and facilitates the adaptation of learning processes to the type of ontology. In the education domain, the ontology type could be called "curriculum ontology", which includes courses, prerequisites and learning objectives. 2. **ruleset** defines the logical rules and constraints that govern the ontology. It provides a deductive framework for reasoning and knowledge inference, while ensuring compliance with domain-specific rules. 3. **conceptualMapping** indicates whether an ontology supports connections between various knowledge structures, such as schemas or ontologies. This capability facilitates interoperability and data integration, allowing for semantic alignment across multiple data sources and systems. In a multilingual educational platform, conceptual mapping links equivalent concepts across different language-specific ontologies.

1. **ontologyCreation()** method automates the development of new ontologies by extracting concepts, relationships and properties from existing data. This streamlines the design process and minimises manual effort, ensuring that the resulting ontologies are data-driven and accurately represent the domain. It can extract data from both structured and unstructured sources, identifying key concepts and relationships using machine learning algorithms such as clus-

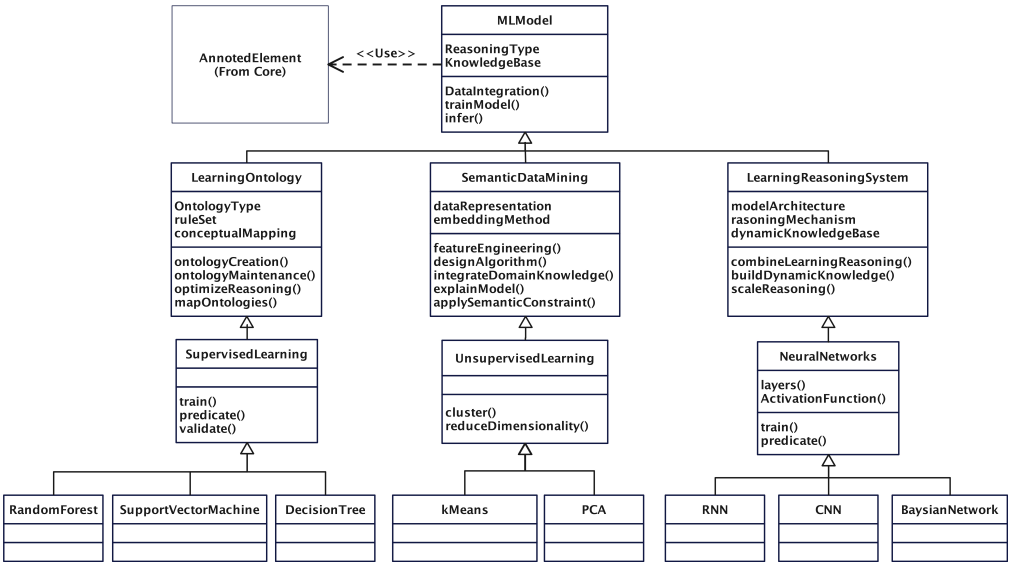

Figure 4: Extensible Learning Package for extending CWM Analysis layer.

tering and neural networks. The extracted knowledge is then formalised in ontology formats such as OWL or RDF. 2. **ontologyMaintenance()** function updates and refines the ontology to ensure its relevance and accuracy over time. It keeps the ontology synchronised with changes in the domain or data, supporting the dynamic evolution of knowledge structures. It can monitor domain changes or new data input and automatically add, modify or delete concepts and relationships. Updates are validated using domain-specific rules or expert feedback. 3. **optimizeReasoning** improves the reasoning process by incorporating machine learning techniques, resulting in faster and more accurate reasoning. It also reduces the computational complexity of ontology-based reasoning, allowing reasoning engines to adapt to real-time applications. The processes can be : • **Identify bottlenecks**: Recognise the limitations of traditional reasoning methods, such as long inference times. • **Apply Machine Learning Models**: Use models such as Recursive Reasoning Networks or heuristic optimizations to increase efficiency. • **Validate results**: Ensure accuracy by comparing results against established ontological rules. This approach can improve the system's ability to predict the optimal course sequence for students based on their learning history. 4. **mapOntologies()** effortlessly aligns disparate ontologies, paving the way for seamless data integration and interoperability. It fuses knowledge from different sources to present a unified view, while resolving any annoying semantic inconsistencies. First things first: identify common or equivalent concepts across ontologies. We use sophisticated algorithms such as Random Forest or Word2Vec to align concepts and their relationships. Once we've mapped them, we validate the relationships to ensure everything is consistent and accurate.

4.2.3 The *SemanticDataMining* Component

SemanticDataMining ocuses on feature engineering, algorithm design, domain knowledge integration, and model explanation. It includes attributes like dataRepresentation and embeddingMethod.

1. **dataRepresentation** defines how data is structured before it enters the machine learning pipeline. It ensures that the data is compatible with both the data mining algorithms and the associated ontological framework. This process determines whether to use raw data, semantic embeddings or ontologically annotated data. In short, effective data representation is critical to harnessing the power of machine learning technologies. A use case could be "Learning Activity Features with Ontological Annotations". 2. **embeddingMethod** transforms data into a format rich in semantic knowledge. This process enables data mining algorithms to effectively utilize ontological information. By integrating these techniques, we can enhance the capabilities of our data analysis and improve decision-making processes. We propose "Word2Vec for Learning Object Metadata" as an example. 3. **semanticConstraints** refers to the application of semantic rules during the data mining process. This feature ensures that machine learning models follow the specific rules and relationships outlined by the underlying ontology, thereby maintaining domain relevance and integrity.

1. **featureEngineering()** function enhances, selects or extracts features from raw data by incorporating ontological knowledge. This process improves the relevance and representation of data for the machine learning pipeline. The types of Feature Engineering are : ***a)*** *Feature Augmentation*: Adds additional semantic features derived from the ontology ***b)*** *Feature Selection*: Identifies the most relevant features based on ontological rules ***c)*** *Feature Extraction*: Transform raw data into meaningful features using ontology-based embeddings. The feature engineering process includes the following steps : ***a)*** Extract raw features from the dataset. ***b)*** Using the ontology to add, reduce or transform features. ***c)*** Outputting a semantically enriched feature set. By following these steps, we can create a more effective and informative dataset for machine learning applications. 2. **designAlgorithm()** function designs or adapts data mining algorithms to incorporate ontological knowledge, aligning the logic and behaviour of the algorithm with domain-specific semantics. Types of Ontology-Enhanced Algorithms are like ***a)*** *Ontology-based decision trees*: These use decision nodes that exploit semantic relationships. ***b)*** *Ontology-based neural networks*: The layers or architecture of these networks are influenced by ontological hierarchies. ***c)*** *Ontology-based probabilistic graphical models*: In these models, relationships are driven by ontology constraints. Identifying the data mining problem (e.g. classification or clustering). Adapting an existing algorithm or creating a new one that incorporates ontology-driven rules. Validating the performance of the algorithm on semantically enriched data sets. 3. **integrateDomainKnowledge()** incorporates domain-specific knowledge from an ontology into the data mining pipeline. This approach ensures that machine learning models benefit from structured and contextual information. Steps to Implement: ***a)*** *Load Domain Ontology*: Import the relevant domain ontology associated with the dataset. ***b)*** *Identify Key Concepts*: Recognize essential concepts, relationships, and constraints that need to be integrated. ***c)*** *Apply Ontological Insights*: Utilize the identified insights to inform preprocessing, model training, and validation. This structured approach enhances the machine learning process by grounding it in relevant domain knowledge. 4. **explainModel()** improves model interpretability by linking predictions or decisions to ontological concepts and relationships. It increases the transparency and trustworthiness of machine learning models by providing domain-relevant justifications for model outputs. The steps are to analyse the model's predictions or decision paths, map these outputs to relevant ontological concepts and rules, and generate explanations that align the predictions with domain knowledge.

4.2.4 The *LearningReasoningSystem* Component

The **LearningReasoningSystem** leverages hybrid AI techniques to dynamically adjust its reasoning rules. For example, in a financial risk assessment system, it can detect fraudulent patterns by cross-referencing new transactional data with historical fraud cases, thereby refining its predictive accuracy in real-time.

It combines learning and reasoning capabilities with scalability and neuro-symbolic integration. It includes attributes like modelArchitecture and methods like dynamicAdaptation() and realTimeReasoning().

The properties are crucial for the efficiency and adaptability of hybrid AI. These properties include: 1. **modelArchitecture**: This defines the structure of the learning model used within the system. It could be neural networks, decision trees, or a hybrid approach. The *modelArchitecture* directly impacts how the system processes data and performs reasoning, ensuring it's well-suited for the application's complexity. For example, in a dynamic recommendation system, a "*Recurrent Neural Network with Ontology-Based Constraints*" architecture could optimize suggestions by considering semantic relationships between items. 2. **reasoningMechanism**: This specifies the reasoning framework used to infer knowledge or validate decisions. This could involve rule-based, probabilistic, or hybrid reasoning. The *reasoningMechanism* guides decision-making by leveraging domain-specific logic and combining data-driven learning with deductive reasoning. For instance, in a financial fraud detection system, a "*Probabilistic Reasoning with Bayesian Networks*" mechanism could evaluate suspicious transactions by considering the probabilities of fraudulent behavior. 3. **dynamicKnowledgeBase**: This indicates whether the system's knowledge base is dynamically updated with new information. A *dynamicKnowledgeBase* allows the system to adapt to evolving domains and learn in real-time. A medical diagnostic system that continuously updates its knowledge base with the latest research and treatment recommendations is a good example, ensuring more precise and relevant analyses. These properties a powerful and adaptable tool. It can evolve with changing contexts, improve the quality of decisions, and provide a deeper understanding of the domains it models.

The **LearningReasoningSystem** combines learning models with reasoning engines to improve decision-making, adaptability, and scalability. It relies on four key methods:

1. buildDynamicKnowledge() integrates machine

learning with reasoning engines to create a hybrid system. Machine learning detects patterns, while reasoning engines refine predictions using rules or ontology-based logic. This improves accuracy and explainability. For example, in a tutoring system, it predicts student knowledge gaps and adjusts recommendations based on prerequisite rules. 2. **buildDynamicKnowledge()** keeps the knowledge base up to date by integrating new data and insights. The system continuously monitors input, updates validated information, and ensures consistency. In healthcare, it adds new treatments and diseases dynamically, ensuring diagnostic models stay current. 3. **scaleReasoning()** enhances the system's ability to process large and complex datasets efficiently. Optimization techniques, such as heuristic algorithms and parallel computing, improve reasoning performance. In urban planning, this method analyzes traffic data from sensor networks to suggest optimal routes. 4. **integrateNeuroSymbolicApproaches()** combines neural networks with symbolic reasoning for better decision-making. Neural networks handle unstructured data, while symbolic reasoning refines predictions for more reliable results. In legal document analysis, it classifies documents with AI and validates classifications using legal rules. These methods make the **LearningReasoningSystem** more adaptive, scalable, and capable of delivering precise and interpretable decisions.

5 CONCLUSION

The CWM framework offers a standardized approach to metadata management, yet its limited semantic expressiveness restricts its applicability in complex, heterogeneous environments. This paper proposes an extended CWM framework that incorporates ontologies, machine learning, and knowledge reinjection policies to address these limitations. By introducing new layers and components, such as the "Learning Package" and advanced knowledge mapping mechanisms, the framework enhances the semantic interoperability, adaptability, and usability of data warehouse systems. The research explores the integration of hybrid AI systems, combining inductive and deductive techniques, to improve knowledge discovery and decision-making. With the extension of the CWM mentioned in (Ralaivao et al., 2024) and the present extension, we obtain an extended CWM framework capable of discovering and managing knowledge. As a way forward, we propose experimentation and benchmarking on simple and knowledge-based queries. Future work will focus on refining the reinjection mechanism with adaptive learning techniques and evaluating its applicability across various domains, such as education and learning, healthcare, finance, and smart cities.

REFERENCES

Antunes, A. L., Cardoso, E., and Barateiro, J. (2022). Incorporation of ontologies in data warehouse/business intelligence systems-a systematic literature review. 2(2):100131.

da Silva, J., de Oliveira, A. G., do Nascimento Fidalgo, R., Salgado, A. C., and Times, V. C. Modelling and querying geographical data warehouses. 35:592–614.

Falquet, G., Métral, C., Teller, J., Tweed, C., Roussey, C., Pinet, F., Kang, M. A., and Corcho, O. (2011). An introduction to ontologies and ontology engineering. *Ontologies in Urban development projects*, pages 9–38.

Ghidalia, S., Narsis, O. L., Bertaux, A., and Nicolle, C. (2024). Combining machine learning and ontology: A systematic literature review. *arXiv preprint arXiv:2401.07744*.

Gomes, P., Farinha, J. T., and Trigueiros, M. J. A data quality metamodel extension to cwm. In *Asia-Pacific Conference on Conceptual Modelling*.

Midouni, S. A. D., Darmont, J., and Bentayeb, F. Approche de modélisation multidimensionnelle des données complexes : application aux données médicales,. In *Journées Francophones sur les Entrepôts de Données et l'Analyse en ligne*.

Olsina, L. (2021). The foundational ontology thingfo: Architectural aspects, concepts, and applicability. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, pages 73–99. Springer.

Ralaivao, C., Razafindraibe, F., Raherinirina, A., and Rakotonirainy, H. (2024). Cwm extensions for knowledge and metadata integration for complex data warehouse and big data. In *Proceedings of the 26th International Conference on Enterprise Information Systems, Angers, France*, volume 1, pages 329–336. SCITEPRESS - Science and Technology Publications.

Tavac, M. and Tavac, V. The general algorithm for the design of the mda transformation models. pages 171–176.

Thavornun, V. (2015). Metadata management for knowledge discovery. Master's thesis.

Zayakin, V. S., Lyadova, L. N., Lanin, V. V., Zamyatina, E. B., and Rabchevskiy, E. A. (2021). An ontology-driven approach to the analytical platform development for data-intensive domains. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, volume 1718, pages 129–149. Springer, Springer International Publishing.