Industrial Semantics-Aware Digital Twins: A Hybrid Graph Matching Approach for Asset Administration Shells

Ariana Metović*, Nicolai Maisch, Samed Ajdinović,
Armin Lechler, Andreas Wortmann and Oliver Riedel
Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW),
University of Stuttgart, Seidenstr.36, 70174 Stuttgart, Germany
ariana.metovic@isw.uni-stuttgart.de
*Corresponding author

***Abstract*—Although the Asset Administration Shell (AAS) standard provides a structured and machine-readable representation of industrial assets, their semantic comparability remains a major challenge, particularly when different vocabularies and modeling practices are used. Engineering would benefit from retrieving existing AAS models that are similar to the target in order to reuse submodels, parameters, and metadata. In practice, however, heterogeneous vocabularies and divergent modeling conventions hinder automated, content-level comparison across AAS. This paper proposes a hybrid graph matching approach to enable semantics-aware comparison of Digital Twin representations. The method combines rule-based pre-filtering using SPARQL with embedding-based similarity calculation leveraging RDF2vec to capture both structural and semantic relationships between AAS models. This contribution provides a foundation for enhanced discovery, reuse, and automated configuration in Digital Twin networks.**

***Index Terms*—Digital Twins, Asset Administration Shell, Semantic Graph Matching, RDF2vec, Graph Embeddings**

I. Introduction

The increasing digitization of industrial production is fundamentally transforming how assets and processes are managed. A key enabler of this transformation is the Digital Twin, which establishes a dynamic connection between physical assets and their digital counterparts, enabling monitoring, analysis, and control across the product lifecycle [1]. Various approaches and data models have emerged in order to provide the foundation for digital twins, such as the Asset Administration Shell (AAS) [2]. The AAS is a standardized metamodel that provides a machine-interpretable description of assets within Industry 4.0 environments, thereby ensuring interoperability across the manufacturing industry and beyond [3].

In practice, however, AAS models often rely on different vocabularies and modeling conventions. Although these models may be syntactically valid, they frequently lack semantic alignment, making automated comparison and reuse difficult.

An industrial use case that often involves significant manual effort and potential duplication arises when modeling a new product. Since each product must be represented individually, efficiently identifying and reusing existing models and assets can greatly simplify engineering tasks. Reusing available submodels, parameters, and metadata not only reduces engineering effort but also ensures consistency across products. However, even with semantic annotations, identifying appropriate AAS models within a large pool of assets can still be challenging. The resulting heterogeneity increases manual integration efforts for example, when equipment from multiple suppliers must be combined into a unified production system.

In the fields of Semantic Web and Machine Learning, promising technologies such as ontologies and graph-based [4] or vector-based representations [5] offer potential for improving AAS discovery. This paper proposes a hybrid graph-matching approach that combines rule-based semantic pre-filtering with vector-based similarity computation to enable more accurate matching of heterogeneous AAS models. While graph-based discovery (e.g., using query languages such as SPARQL) captures the structural composition of AAS models vector-based methods provide semantic comparability. Because brownfield AAS models often lack uniform structures and vocabularies, combining both perspectives enables flexible and robust discovery across heterogeneous AAS networks and repositories. This approach not only improves the accuracy of matching relevant AAS instances but also enhances their reusability and interoperability within Digital Twin networks.

The remainder of this paper is structured as follows. Section II introduces the fundamental concepts and terminology relevant to Digital Twins and semantic modeling. Section III reviews the state of the art in semantic matching, as well as graph- and vector-based similarity methods, and highlights the existing research gaps. Section IV presents the proposed hybrid graph matching approach, followed by the modeling and learning objectives in Section V and the discussion of testing pipeline and weaknesses in Section VI. Section VII concludes the paper and outlines directions for future work.

II. Fundamentals

In this section the formal structure of the AAS and the function of Resource Description Framework (RDF) as a standard model for data interchange on the Web is outlined.

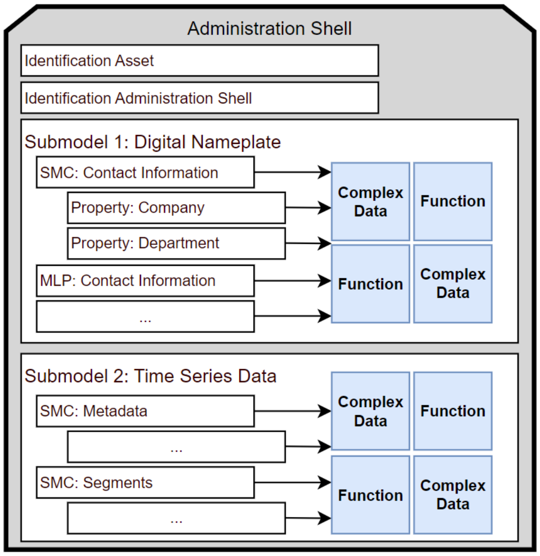

Fig. 1. Example of the basic Structure of an Asset Administration Shell based on [3]

A. Asset Administration Shell

The AAS is a metamodel standard for digitally representing industrial assets. The basic structure of the AAS consists of information about both the asset and the AAS itself (e.g., unique identification details), as well as submodels that contain more detailed information depending on the specific use cases [3]. (Fig. 1) shows an example AAS featuring a digital nameplate and time series data, with corresponding submodel templates available from the IDTA[1].

Serialized in standard formats such as JSON or XML, or in the custom AASX format [6], the AAS can be flexibly distributed, modified, and extended. Each information element of the AAS (such as name or referenced submodels) and its submodels (such as nameplate or energy usage) is identified using unique identifiers and can be semantically described using an internal concept description or references to standardized vocabularies (e.g., IEC CDD [7] or ECLASS [8]). These annotations can be used to discover assets with specific attributes such as finding all pumps on a shop floor or identifying all assets older than five years.

B. Knowledge Graphs and Resource Description Framework

A knowledge graph is a technique for structuring and linking information through explicit relationships between entities, forming a modular and extensible graph with nodes and edges [9]. An ontology is an explicit specification of a conceptualization [10]. A knowledge graph organizes data through nodes and relationships, while an ontology provides the formal vocabulary and rules that define the meaning of nodes and relations. The W3C standard RDF is used as a technical implementation for describing and interlinking data in the Semantic Web [11]. RDF describes information in form of triples that combine to form graphs and enable a flexible, extensible representation [12]. An RDF statement has the following structure:

$$\langle \text{subject} \rangle \quad \langle \text{predicate} \rangle \quad \langle \text{object} \rangle$$

An RDF statement expresses a relationship between two resources, the subject and the object. The relationship is directed from the subject to the object and the predicate is describing the relationship [11]. RDF can be serialized in various formats, such as RDF/XML, N-Triples, JSON-LD or Turtle [13]. SPARQL is a query language for RDF data, used for querying and manipulating RDF graph content on the Web or in an RDF store. It supports SELECT queries (return variable bindings), ASK queries (e.g. boolean) and CONSTRUCT queries (construct new RDF graphs from a query result) [14].

III. State of The Art

A. RDF Technology for AAS

Besides the serialization as JSON or XML, the AAS may be represented as an RDF graph [15], enabling semantic reasoning on the AAS metamodel. [16] proposed a Python package for converting AAS data into RDF format[2]. It also provides a graphical user interface for editing and showing converted AAS. However, there remain challenges in the conversion, such as the lack of support for ordered elements in the RDF representation of an AAS [16]. Other AAS tools such as Eclipse BaSyx [17], which supports AAS hosting, or AAS4J [18], which enables AAS-based development in Java, feature RDF only as experimental plugins[3] or planned extensions[4]. The platform metaphactory[5] provides an application layer on top of RDF knowledge graphs that integrate ontologies and standard dictionaries (e.g., IEC CDD, ECLASS) together with AAS. It offers a SPARQL-based discovery stack and a user interface for exploring entities and relations. An integrated AI service enables natural-language questions that are translated into SPARQL, making semantic querying accessible to non-native SPARQL users [19].

B. RDF Stores and Querying

Within Semantic Web technologies, extensive RDF stores comprising numerous RDF triples are established[6]. This infrastructure supports complex querying, information discovery, and reasoning through SPARQL querying [20]. The main components of an RDF store are the repository and the middleware to continuously communicating with the underlying repository [21]. Various implementations of RDF stores are suitable as a backbone of semantic applications that need to store and process large amount of RDF data in a reliable manner and

[1] https://industrialdigitaltwin.org/en/content-hub/submodels

[2] GitHub of the converter: https://github.com/mhrimaz/py-aas-rdf

[3] GitHub branch Experimental/Adapter/RDF: https://github.com/eclipse-basyx/basyx-python-sdk/tree/Experimental/Adapter/RDF

[4] GitHub Eclipse AAS4J: https://projects.eclipse.org/projects/dt.aas4j

[5] https://metaphacts.com/metaphactory

[6] Including Ontotext GraphDB, OpenLink Virtuoso, Apache Jena Fuseki, Eclipse RDF4J, Stardog or Amazon Neptune

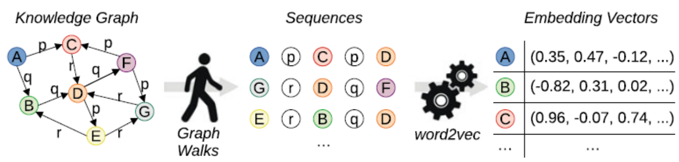

Fig. 2. Overall workflow of RDF2vec [29]

that most of them provide support for important requirements such as data protection, information privacy and security [22].

[23] and [24] present SPARQL benchmark studies on real-world datasets, demonstrating its scalability and efficiency for querying complex data.

C. Comparability of RDF Graphs and Data

By representing information as semantic statements, RDF stores link nodes, allowing them to be placed in relationships. A variety of techniques exist to compare such graphs, ranging from structural matching based on the formal structure of the graph(s) over semantics-aware alignment based on semantic annotations up to embedding-based similarity.

[25] proposes a graph-based method for efficient duplicate detection in RDF datasets by comparing subgraphs for similarity and aggregating these to detect duplicates. While conceptually related, the approach focuses on data cleansing rather than semantic querying or graph-based knowledge retrieval.

[26] utilizes Graph Edit Distance (GED), a method that measures the minimum cost to transform one (sub)graph into another [27], and enriches the costs with predicate/type similarity to reflect RDF labels, therefore combining structural and semantic similarities.

[28] introduces a formal framework for pairwise entity comparison in RDF graphs, generating explicit similarity or difference queries instead of producing a global similarity score suitable for large-scale retrieval.

D. Vectorial Matching Approaches

With the rise of machine learning, particularly language models, the embedding-based approach RDF2vec [29] was introduced. It generates vector representations from random walks or triple sequences, such that semantic and structural proximity in the RDF graph is reflected as geometric closeness in the embedding space (Fig.2). In RDF2vec, extracted sequences of entities and relations from the graph are then fed into the Word2vec algorithm, producing embedding vectors for all entities encountered in these walks. Graph similarity is then computed using distances between these vectors.

In the field of Industry 4.0, there are ongoing efforts to semantically match information contained within AAS, or even across different data models, using natural language processing techniques to achieve semantic interoperability [30]. By generating vector representations of AAS and their data (embeddings), semantic similarity can be quantified [31]. However, this approach primarily handles textual representations of AAS, which can pose limitations when analyzing large numbers of instances. [32] presented a Retrieval-Augmented

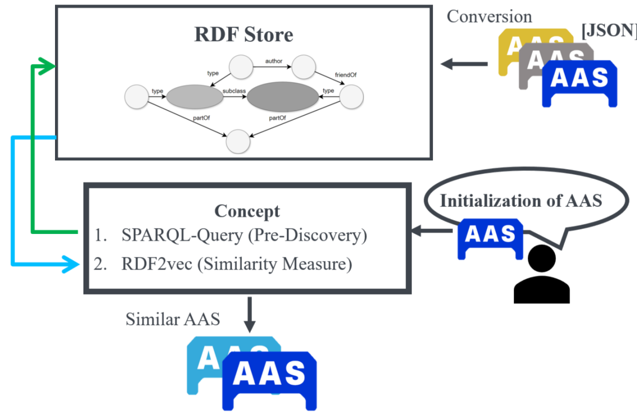

Fig. 3. Concept of the the proposed Hybrid Graph Matching Approach

Generation (RAG) [33]–based semantic matching algorithm that bridges multiple industrial information models. By utilizing vector representations of instances, the study demonstrated how language models can identify semantic similarities to automatically integrate heterogeneous data entities.

E. Research Gaps

Although RDF has been used for AAS discovery, methods for automatically comparing AAS across large RDF stores are still lacking. Current systems expose rich metadata but do not support automated content-level comparisons. SPARQL queries return explicit structural matches but cannot reveal similarities between conceptually related entities expressed with different vocabularies or naming conventions. Semantic matching approaches based on embeddings perform well for similarity-based matching tasks and are able to capture only local graph structure but may have computational limitations in large, complex data stores and are restricted to textual information. By combining advantages of both technologies semantic matching and discovery can be improved. In summary, RDF-based discovery can be enhanced with vector-based representations to leverage semantic features for more effective AAS discovery. We therefore propose a hybrid method to quantify similarity between AAS graphs/subgraphs and automatically retrieve comparable AAS models. This integrates

1) the structural relationships of the RDF graph,
2) the semantic meanings captured by ontologies, and
3) synonymy and terminological variation,

thereby providing a robust foundation for content-level comparability of AAS in RDF.

IV. Architecture

The method combines pre-filtering using SPARQL over RDF with vector-space similarity via RDF2vec, enabling identification of AAS models that are semantically similar to a target while leveraging the strengths of both paradigms (Fig. 3). The objective is to enable constraint handling and context-sensitive semantics.

While RDF's structured querying enables navigation through the AAS composition, vector-based semantic comparisons can resolve similar data such as "power input" and "electric power". In terms of AAS modeling, two AAS instances may both implement the same submodels, but additionally describe related properties possibly using different labels. Because they have a similar set of submodels and their contexts are semantically aligned, the query algorithm will most likely match.

A. Data Storage and pre-discovery via SPARQL

Existing AAS instances and templates are provided in JSON and converted into RDF using the mapping tool [16]. The resulting graphs are persisted in a data store and exposed via SPARQL endpoints, yielding a queryable representation of all AAS and enabling pre-discovery as the concept's first step.

In industrial scenarios, the AAS used as input for a workflow is incomplete: it may lack a specific submodel (e.g. Timeseries) or standardized parameters (e.g. a torque parameter defined according to ECLASS or IEC CDD), while still containing essential information about the asset (e.g. Digital Nameplate). The goal is not to compare this partial AAS with all available AAS, but only with candidates structurally fulfilling missing requirements. For this reason, SPARQL is used as an explicit constraint mechanism rather than a similarity measure. Missing information is expressed as SPARQL constraints (e.g. required submodels, properties). This yields a filtered AAS set, reducing computation and avoiding irrelevant matches while ensuring only structurally valid candidates proceed to semantic scoring.

For instance, during the engineering process a user's AAS is missing the Timeseries submodel. The user wants to find existing AAS instances that include this submodel so that missing parameters can be reused. Listing 1 presents a pseudo-SPARQL query that implements this filtering pattern and returns only AAS containing the required submodel. The result is a set of plausible candidates that are passed to the next stage.

Listing 1. Pseudo-SPARQL Query

```
SELECT AAS
WHERE {
    ?aas hasSubmodel ?sm .
    ?sm hasIdShort TimeSeriesData
}
```

B. Vectorization via RDF2vec

After SPARQL pre-discovery, we embed the filtered AAS graphs with RDF2vec to obtain vector representations suitable for similarity search. On the filtered graphs, we train or reuse RDF2vec embeddings to obtain vector representations of entities. RDF2vec extends the DeepWalk principle to RDF data by performing random or breadth-first graph walks over entities and relations [34]. The resulting sequences are then treated analogously to text sentences, see Fig. 2, and embedded using neural language models. In this way, entities occurring in similar structural contexts receive similar vector representations, enabling a quantitative comparison of AAS based on semantic structure.

V. Challenges and Implementation

This chapter outlines the primary research objectives of the RDF2vec method and the process for determining the similarity threshold for the resulting AAS.

A. RDF2vec method

The approach leads to the following learning objectives and challenges for RDF2vec in knowledge graphs [35]:

- RDF2vec aims to preserve the topological structure of an RDF graph within a continuous vector space. By translating graph walks into linear sequences, it enables neural models to capture neighborhood relations between entities. The risk hereby is a focus on structural context while semantic similarities are neglected. For instance, two AAS using alternative property names may represent equivalent semantics, yet RDF2vec embeddings could diverge if the syntactic structures differ [35].
- RDF graphs contain literal values such as numbers and dates. Standard RDF2vec models primarily operate on entities and relations, often neglecting literals such as numerical values or textual attributes, which are essential in AAS contexts for describing quantitative parameters such as voltage, current, or torque. Neglecting these literals can lead to incomplete semantic representations. There has been previous work on that topic [36], [37].

Addressing these challenges is one of the key learning objectives of this work, as well as the opportunities resulting from it after the implementation.

B. Similarity metric and threshold

Another aspect that is to be investigated is the choice of the similarity metric for the resulting vectors. The similarity between the query AAS and each candidate is computed by using the standard cosine similarity measure [38]. The similarity between two entities e_1 and e_2 is calculated as the cosine similarity between the vectors V_1 and V_2 [38]:

$$\mathrm{sim}(e_1, e_2) = \frac{V_1 \cdot V_2}{\|V_1\| \cdot \|V_2\|} \tag{1}$$

Other distance measures can be substituted without changing the overall pipeline. Some works report better results when using other distance metrics, for example Euclidean distance [39] or Manhattan distance [40]. Euclidean distance measures the straight-line distance between two points or vectors in an n-dimensional space. After defining the metric, a fitting threshold determining the number of output AAS must be defined.

$$\mathrm{EuclideanDistance} = d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

Let G_q be the query AAS that the user in Fig. 3 wants to fill with more data and $\mathcal{G} = \{G_i\}$ the candidate set obtained after SPARQL pre-filtering in the first step of the concept. A similarity index $s(G_q, G_i)$ as the cosine similarity between vector representations of G_q and G_i with V_1 and V_2 being identical for $s = 1$ is computed. Three decision policies regarding the output of a *similar AAS* are considered:

- Threshold policy: Return all G_i with $s(G_q, G_i) \geq t$.
- Top-k policy: Return the k most similar AAS.
- Hybrid policy: Return $\{G_i | s(G_q, G_i) \geq t\}$, forming set R_t. If $|R_t| < k$, the set is filled up to k by decreasing s.

The method to compute the variables k and t are to be determined by existing studies or empirically. Depending on the metric selection, it is to be investigated whether a mapping from the original domain to the target decision scale of $[0, 1]$ is necessary, as for example the cosine similarity is defined in the interval $[-1, 1]$.

VI. Discussion

A. Testing Pipeline

To ensure verifiability and reproducibility, a structured testing pipeline is defined. First, a toy dataset with a small number of AAS instances including varied vocabularies, renamed properties, and partial template implementations will be constructed. This dataset enables a concrete step-by-step walkthrough of the hybrid pipeline:

1) Transformation of an AAS to an RDF graph,
2) SPARQL constraint filtering,
3) Generation of RDF2vec walks,
4) Ranking of retrieved AAS by similarity.

Second, the embedding model used in the experiments can either be pre-trained once on the entire AAS repository or trained specifically for the toy dataset. Embedding training is therefore not part of the evaluation itself, but part of the reproducibility setup. In practical deployments, embeddings would be trained once upfront and reused throughout subsequent testing phases. Parameters that are essential for the reproduction pipeline are graph walk configurations, embedding parameters and similarity thresholds.

B. Weaknesses and failure modes

The proposed approach relies on several assumptions that introduce potential weaknesses.

1) Usability issues: SPARQL filtering requires a level of structural knowledge about missing components of the incomplete AAS, typically a name or identifier of a submodel or its key properties. A central challenge is the high variability of modeling practices across manufacturers. It limits applicability in scenarios where users cannot express structural constraints. Applying a filter then may preemptively exclude semantically valid AAS instances from the feasible set.

2) Vocabulary and modeling diversity: SPARQL filtering reduces complexity but does not eliminate the risk of irrelevant or marginally related candidates when vocabularies deviate strongly. Embedding-based similarity reduces terminological variation, but the combined structural–semantic retrieval may still fail when assets are semantically close yet modeled using substantially different structural patterns. This may lead receiving low similarity scores for similar AAS. Another challenge is harmonizing vocabularies such as ECLASS and IEC CDD, which will be addressed through standardization. The development of the harmonization should be monitored and integrated in the outlook[7].

3) Scalability and computational cost: Embedding-based similarity requires generating RDF2Vec embeddings over potentially large AAS knowledge graphs. Depending on the number of AAS instances, the size of their submodels, and the graph walk parameters, the computational cost can become significant. Pre-filtering via SPARQL reduces the candidate set, but the full workflow may still face scalability constraints in industrial-scale repositories with thousands of heterogeneous AAS [41]. It is therefore advisable to consider and evaluate different RDF2Vec implementations [29]. While the approach is expected to work for small and medium-sized repositories, the threshold at which embedding generation or similarity computation becomes inefficient has not yet been empirically determined and remains an open research question.

VII. Conclusion and Outlook

This paper introduced a hybrid graph matching approach to enhance semantic discovery among Digital Twin representations based on the AAS. The proposed method combines rule-based pre-filtering using SPARQL with embedding-based similarity computation via RDF2vec, thus integrating symbolic reasoning and statistical learning into a unified framework. This fusion enables the comparison of AAS instances not only by structural similarity but also by semantic context, addressing limitations of purely syntactic or purely embedding-based techniques enabling re-usage of existing AAS instances in industry applications like modeling a new product.

By leveraging SPARQL, domain-specific constraints and ontology-based semantics can be incorporated directly into the matching process, ensuring that only contextually relevant candidates are considered. RDF2vec embeddings capture the relationships between entities, allowing for context-aware similarity scoring across vocabularies and modeling conventions. This concept demonstrates the potential of combining rule-based and data-driven querying for Digital Twin interoperability. The proposed approach contributes toward reducing manual integration effort, supporting asset discovery, and enabling knowledge reuse across manufacturers and domains.

Future work will focus on several directions. First, the approach will be empirically validated using larger and more diverse AAS datasets. Second, the modeling objectives regarding the RDF2vec method will be explored to improve representation and understanding of numerical properties in AAS, as well as ensuring the capture of semantically related information. Lastly, case studies will lead to investigation of the best fitting similarity parameters.

[7]https://github.com/eclipse-esmf/esmf-semantic-aspect-meta-model/issues/340

Acknowledgements

This Project is supported by the Federal Ministry for Economic Affairs and Energy (BMWE) on the basis of a decision by the German Bundestag – CompAAS – KK5311208RG4.

Partly funded by the Federal Ministry for Economic Affairs and Energy (BMWE) – Factory-X – 13MX001L (https://factory-x.org) and growING – 13IPC036G.

References

[1] M. Soori, B. Arezoo, and R. Dastres, "Digital twin for smart manufacturing, A review," *Sustainable Manufacturing and Service Economics*, vol. 2, no. 100017, p. 100017, Apr. 2023.

[2] J. Zhang, C. Ellwein, M. Heithoff, J. Michael, and A. Wortmann, "Digital twin and the asset administration shell," *Software and Systems Modeling*, vol. 24, no. 3, pp. 771–793, 2025.

[3] IDTA-Workstream "Specification of AAS", "Specification of the Asset Administration Shell Part 1: Metamodel," Germany.

[4] M. Liu, H.-Q. Guo, and J.-D. Su, "Semantic matching of ontology instances," in *2007 International Conference on Machine Learning and Cybernetics*. IEEE, 2007.

[5] H. Li, Y. Govind, S. Mudgal, T. Rekatsinas, and A. Doan, "Deep learning for semantic matching: A survey," *J. Comput. Sci. Cybern.*, vol. 37, no. 4, pp. 365–402, Oct. 2021.

[6] IDTA-Workstream "Specification of AAS", "Specification of the Asset Administration Shell Part 5: Package File Format (AASX)," Germany.

[7] International Electrotechnical Commission (IEC), "Common Data Dictionary – CDD." [Online]. Available: https://tc3.iec.ch/tc-activity/common-data-dictionary-cdd/

[8] ECLASS e.V., "An introduction to the standard." [Online]. Available: https://eclass.eu/en/eclass-standard/introduction

[9] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web*, vol. 8, pp. 489–508, 12 2016.

[10] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1042814383710083

[11] RDF Working Group, "Rdf 1.1 primer: Technical report," 2014. [Online]. Available: https://www.w3.org/TR/rdf11-primer/

[12] J. Pan, G. Vetere, J. M. Gomez-Perez, and H. Wu, *Exploiting Linked Data and Knowledge Graphs in Large Organisations*. Springer International Publishing, 01 2017.

[13] RDF Working Group, "Rdf 1.1 turtle: Technical report," 2014. [Online]. Available: https://www.w3.org/TR/turtle/

[14] SPARQL Working Group, "SPARQL 1.1 Overview: W3C Recommendation," 2013. [Online]. Available: https://www.w3.org/TR/sparql11-overview/

[15] S. Rongen, N. Nikolova, and M. van der Pas, "Modelling with AAS and RDF in Industry 4.0," *Computers in Industry*, vol. 148, p. 103910, 2023.

[16] M. Mohammad Hossein Rimaz, Christiane Plociennik, "Semantic asset administration shell for circular economy," *The 2nd International Workshop on Knowledge Graphs for Sustainability*, 2024.

[17] Eclipse BaSyx, "Eclipse basyx python sdk," 2024. [Online]. Available: https://basyx-python-sdk.readthedocs.io/en/latest/

[18] Eclipse Foundation, "Eclipse aas model for java," 2025. [Online]. Available: https://projects.eclipse.org/projects/dt.aas4j

[19] Mohammad Hossein Rimaz, "Smarter Digital Twins with metaphactory: AI, Knowledge Graphs and Asset Administration Shell for Industry 4.0," September 2025.

[20] T. Sagi, M. Lissandrini, T. B. Pedersen, and K. Hose, "A design space for RDF data representations," *The VLDB Journal*, vol. 31, no. 2, pp. 347–373, 2022.

[21] A. Hertel, J. Broekstra, and H. Stuckenschmidt, "RDF Storage and Retrieval Systems," in *Handbook on Ontologies*, S. Staab and R. Studer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 489–508.

[22] G. E. Modoni, M. Sacco, and W. Terkaj, "A survey of RDF store solutions," in *2014 International Conference on Engineering, Technology and Innovation (ICE)*. IEEE, 2014, pp. 1–7.

[23] M. Voigt, A. Mitschick, and J. Schulz, "Yet Another Triple Store Benchmark? Practical Experiences with Real-World Data," vol. 912, 01 2012.

[24] An Ngoc Lam, Brian Elvesæter, and Francisco Martin-Recuerda, Eds., *The Semantic Web: Evaluation of a Representative Selection of SPARQL Query Engines using Wikidata*. Springer Nature Switzerland, 2023.

[25] H. Jin, L. Huang, and P. Yuan, "K-Radius Subgraph Comparison for RDF Data Cleansing," in *Web-Age Information Management*, L. Chen, C. Tang, J. Yang, and Y. Gao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 309–320.

[26] Weiguo Zheng, Lei Zou, Wei Peng, Xifeng Yan, Shaoxu Song, and Dongyan Zhao, "Semantic SPARQL Similarity Search Over RDF Knowledge Graphs," *Proceedings of the VLDB Endowment*, pp. 840–851, 2016.

[27] F. Serratosa, "Redefining the Graph Edit Distance," *SN Computer Science*, vol. 2, no. 6, 2021.

[28] A. Petrova, E. Sherkhonov, B. Cuenca Grau, and I. Horrocks, "Entity Comparison in RDF Graphs," in *The Semantic Web – ISWC 2017*, C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange, and J. Heflin, Eds. Cham: Springer International Publishing, 2017, pp. 526–541.

[29] H. Paulheim, P. Ristoski, and J. Portisch, *Embedding Knowledge Graphs with RDF2vec*. Cham: Springer International Publishing, 2023.

[30] J. Beermann, R. Benfer, M. Both, J. Müller, and C. Diedrich, "Comparison of different natural language processing models to achieve semantic interoperability of heterogeneous asset administration shells," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*. IEEE, Jul. 2023.

[31] Y. Xia, N. Jazdi, and M. Weyrich, "Automated generation of Asset Administration Shell: a transfer learning approach with neural language model and semantic fingerprints," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2022, pp. 1–4.

[32] S. Ajdinović, N. Maisch, R. Kimmel, T. Rasouli, A. Lechler, A. Wortmann, and O. Riedel, "AI-powered semantic matching and data harmonization for industrial applications with OPC UA & asset administration shells," in *2025 The 16th International Conference on Mechanical and Intelligent Manufacturing Technologies (ICMIMT)*. IEEE, May 2025, pp. 248–252.

[33] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[34] P. Ristoski and H. Paulheim, "Rdf2vec: Rdf graph embeddings for data mining," in *The Semantic Web – ISWC 2016*, P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, and Y. Gil, Eds. Cham: Springer International Publishing, 2016, pp. 498–514.

[35] D. T. T. Van and Y.-K. Lee, "A similar structural and semantic integrated method for RDF entity embedding," *Applied Intelligence*, vol. 53, no. 16, pp. 19 302–19 316, 2023.

[36] G. A. Gesese, R. Biswas, M. Alam, and H. Sack, "A survey on knowledge graph embeddings with literals: Which model links better literal-ly?" *Semantic Web*, vol. 12, pp. 1–31, 10 2020.

[37] A. Kristiadi, M. A. Khan, D. Lukovnikov, J. Lehmann, and A. Fischer, "Incorporating literals into knowledge graph embeddings," in *The Semantic Web – ISWC 2019*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, Eds. Cham: Springer International Publishing, 2019, pp. 347–363.

[38] P. Ristoski, J. Rosati, T. D. Noia, R. D. Leone, and H. Paulheim, "RDF2Vec: RDF graph embeddings and their applications," *Semantic Web*, vol. 10, no. 4, pp. 721–752, 2019. [Online]. Available: https://journals.sagepub.com/doi/abs/10.3233/SW-180317

[39] M. Bakhshizadeh, H. Maus, and A. Dengel, "Using semantic-based adaptive relevance prediction to enhance entity recommendation for personal knowledge assistance," in *Knowledge-aware and Conversational Recommender Systems Workshop (KaRS-2024), October 18-18, Bari, Italy*. Association for Computing Machinery, 2024.

[40] O. de Carvalho Júnior, R. Guimarães, A. Gillespie, N. Silva, and R. Gomes, "A new approach to change vector analysis using distance and similarity measures," *Remote Sensing, vol. 3, issue 11, pp. 2473-2493*, vol. 3, pp. 2473–2493, 11 2011.

[41] B. Gergin and C. Chelmis, "Large–Scale Knowledge Graph Embeddings in Apache Spark," in *2024 IEEE International Conference on Big Data (BigData)*, 2024, pp. 243–251.