Automated performance monitoring of HVAC components by artificial intelligence

Maximilian Both [a], *Nicolai* Maisch [a], *Björn* Kämper [a], *Alina* Cartus [a], *Jochen* Müller [a], *Christian* Diedrich [b]

[a] Institute of Building Services Engineering, TH Köln, University of Applied Sciences, Cologne, Germany, {maximilian_alexander.both; nicolai.maisch; bjoern_klaus.kaemper; acartus; jochen.mueller}@th-koeln.de.

[b] Institute of Automation Technology, Otto von Guericke University Magdeburg, Magdeburg, Germany, christian.diedrich@ovgu.de.

Abstract. Energy management systems are an important tool for increasing the energy efficiency of buildings. However, the widespread availability of such systems is offset by the high complexity and high costs of implementation, as well as a lack of data. By using standardized digital twins of technical components, these obstacles can be addressed. In combination with homogeneous semantics of the digital twins and standardized interfaces as uniform access points to the information, the implementation of an energy management system can be simplified. If all technical components of a building have the same information technology structure in the form of digital twins and make their standardized information uniformly available for query, simple query rules can be implemented. These enable the automated integration of the information into an energy management system. However, given the large number of different manufacturers of the technical components, agreement on a common semantic standard in particular seems unlikely. Studies show that methods from the field of Natural Language Processing can be used to process heterogeneous semantics. Agreement on a common vocabulary is no longer necessary. Instead, different semantics can be used and matched to a target vocabulary. In order to use semantic matching in Industrie 4.0 environments, it must be provided as an Industrie 4.0 service. The service provides a translation mechanism from a foreign vocabulary to one's own. For this purpose, a standardized Industrie 4.0 interface consisting of two operations is specified. This interface is implemented prototypically as an API to show how it can be used. The specified interface can be used within the digital twins to process heterogeneous semantics and map them to its own. Extending the Industrie 4.0 approach from homogeneous to heterogeneous semantics can help simplifying the implementation of energy management systems. Simpler implementation lowers the barriers to the use of such systems, which in turn can lead to their higher availability.

Keywords. Industrie 4.0 Interfaces, Natural Language Processing, Energy management systems, Digital twins
DOI: https://doi.org/10.34641/clima.2022.144

1. Introduction

Increasing energy efficiency in every sector of industry is an important aspect of meeting the agreed climate targets of Germany and other countries [1]. In order to identify and leverage energy efficiency potentials, energy management measures, in particular energy management systems (EMS), energy audits, Eco-Management and Audit Schemes (EMAS) and environmental management systems can be used. Although these systems are generally considered useful to increase the energy efficiency of buildings, they are not yet widely used. Studies show that the percentage of companies with EMSs is between 18 and 23% [2]. Barriers to the use of such systems include the high complexity of these, high costs, and a lack of data to identify potential savings [2]. In order to increase the spread of EMS, these obstacles must be overcome: the complexity of the systems must be reduced, the costs lowered and the data basis expanded. Concepts from the areas of Industry 4.0 (I4.0) and Internet of Things (IoT) are ideal for this purpose. The basis of these two concepts is the digital representation of assets. If the digital representation, in the form of a digital twin (DT), is based on a uniform standard (e.g. [3, 4]), the information can be accessed automatically. For this purpose, the DT can also provide standardized interfaces via which its information can be accessed by other DTs. [5]. In addition, a standardized semantics of the DTs' information content is necessary in order to be able to process the information in an automated way [4]. The standardized characteristics can be stored in digital

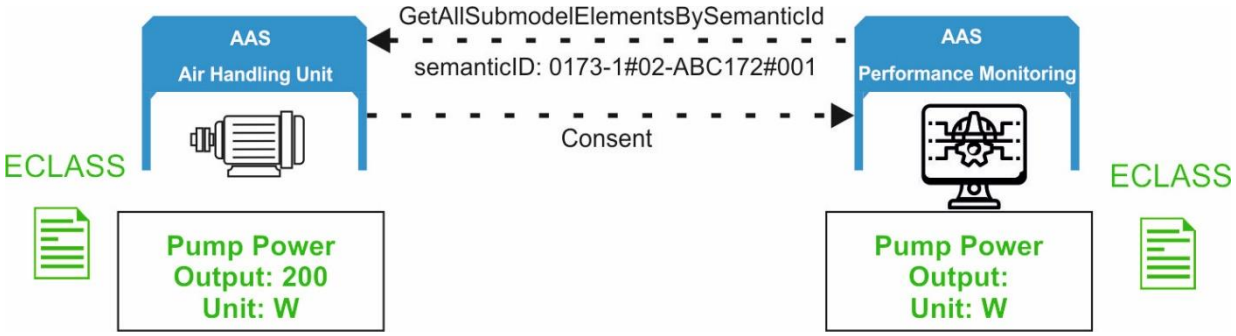

Fig. 1 - Interaction between two Digital Twins

repositories, e.g. [6–8], making them retrievable and available at any time.

An example of an interaction between an EMS and the DT of a pump is shown in Figure 1. Here, the energy management application also has a DT. Specifically, both DTs follow the concept of the asset administration shell (AAS) [4] and are thus based on the same standard. Both AAS have the standardized interface "GetAllSubmodelElementsBySemanticId". Via this interface the EMS application can query the AAS of the pump whether it has the property with the standardized semanticID. This is from the ECLASS standard [6] and is the ID of the "Pump Power Output" property. Since the semantics of the pump's AAS is also based on the ECLASS standard, it can return the requested property and can be integrated into the EMS.

If the three prerequisites (standardized DT, interfaces, semantics) are present, an EMS can be easily set up. Within the EMS, automated rules can be implemented that check new components of a building for their standardized information. One such rule is the query for the "Pump Power Output" and the "Manufacturer Name" of a pump. If these two pieces of information are available in standardized form for all pumps in a building and can be queried via the same interface, they can be automatically integrated into the building's EMS. A manual examination of the information household of components is no longer necessary. This in turn leads to a reduction in complexity (automated creation of the EMS), reduces costs (less manual engineering) and results in a better data basis (standardized information of the DTs).

However, if one of the three conditions is not met, the automated creation of an EMS becomes more difficult (Figure 2).

The EMS and the heat pump both have an AAS and the standardized interface "GetAllSubmodelElementsBySemanticId". However, while the AAS of the EMS is based on the ECLASS standard, the AAS of the heat pump uses a manufacturer-specific vocabulary: the two AASs are based on heterogeneous semantics. Therefore, the ECLASS semanticID is not recognized and the value of the requested property cannot be returned. Thus, an automated query and integration of energy-relevant values into an EMS is not possible. Instead, the semantics of the properties would have to be analyzed and manually linked to an EMS. This in turn leads to increased complexity and rising costs.

One way to solve the problem of heterogeneous semantics is to use Semantic Matching (SM) [9, 10]. Here, methods from the field of artificial intelligence, specifically natural language processing (NLP), are used to map heterogeneous semantics to each other.

This paper designs interfaces that AAS can use to implement an SM service. The interfaces are first specified in a technology-neutral way and then as an API. Additionally, the specified interfaces are implemented prototypically to show how they can be integrated into an AAS. By means of the interfaces it is also possible to process heterogeneous semantics. This further simplifies the implementation of EMS.

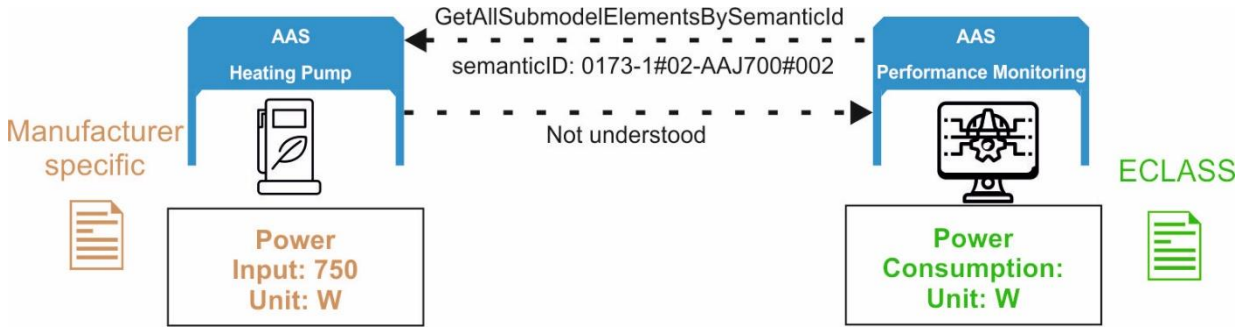

Fig. 2 - Failed Interaction of two Digital Twins

2. Background

I4.0 interfaces and NLP methods are used to implement an SM service.

2.1 The Asset Administration Shell

The transformation from rigid value chains to flexible, highly dynamic and globally networked value networks characterizes the current efforts of organizations towards I4.0. Various fields of action are being addressed for this purpose, one of the central ones being the topic of interoperability [11]. In the field of I4.0, the concept of the AAS has established itself as the basis for interoperability. The AAS is the digital representative of an asset in the digital world [12]. Here, an asset can be any entity that has value for an organization [13]. The information model of the AAS defines the structure of how information of different assets must be arranged[4]. The composition of AAS and asset is called I4.0-component. The central building blocks of the AAS are submodels, which represent the properties and functionalities of the assets and their contents in the form of submodel elements (SE) [4]. Submodels represent, for example, identification, design, or configuration. In addition to the specified structure through the information model, a unique semantics is required for the interaction of I4.0-components [12, 14]. This is achieved if the submodels are available in standardized form and their semantics are also uniformly standardized. The semantics of submodels or SEs can be uniquely described by semantic IDs to locally stored or online available repositories or ontologies.

Various initiatives standardize submodels for technical components, e.g. pumps or drives [15, 16]. The integration of the submodels in vocabularies available online [6–8] enables the unambiguous identification of the submodels and their SEs and thus paves the way for unambiguous semantics. In the current I4.0-approach, interoperability is achieved through a uniform structure (information model AAS), semantics (homogeneous language space) and the provision of services based on standardized interfaces.

2.2 Industrie 4.0 Interfaces

The access to the information of an AAS is realized by interfaces, provided by the AAS. To enable automated access to the interfaces of different AAS, these interfaces are standardized. According to [5] the I4.0-service model describes four levels that are passed through during the standardization of interfaces. First, at a technology-neutral level, the interfaces and associated operations are described in a general textual form. The second level describes how these general descriptions can be applied in different technologies (e.g., HTTP/REST, OPC UA, MQTT). The third level is called the implementation level and includes the implementation of the interfaces using a concrete language such as Python or Java. The last level describes the runtime level, i.e., the concrete implementation of the interfaces in an I4.0-environment. [5]

In [5] the levels 1 and 2 of different interfaces are described, e.g. "GetSubmodel". This interface can be called to retrieve a specific submodel of an AAS. The general interfaces can be provided in different services. A service consists of different interfaces [13]. A basic distinction is made between "Infrastructure Services" and "Application relevant Software Services". Infrastructure services refer to general services that can be reused in concrete application services. Asset-related services, which are provided by AAS, represent a subcategory of application services. [17]

2.3 Semantic Matching for Industrie 4.0 Administration Shells with heterogenous semantics

Currently, the I4.0 research approach focuses on homogeneous semantics to achieve semantic interoperability and interoperability based on it. If AAS with heterogeneous semantics are used, interoperability cannot be guaranteed [4]. To extend this approach to heterogeneous semantics, a method is developed that can automatically map heterogeneous semantics to each other. This method is called semantic matching (SM) [10]. The basis of SM are methods from the NLP domain. The goal of NLP is to make computers understand human language and realize interactions based on it [18]. For this purpose, Language Models (LM) (e.g. [19–24] are used. The training of these takes place in a two-stage process. In the first step, the models are trained on large amounts of general text (pretraining). In the second step (fine tuning) the models are trained for specific NLP tasks (e.g. text classification or question answering). Based on these LMs and the specific NLP task of paraphrase identification [25] heterogeneous semantics can be processed.

The developed approach is based on the idea that two SEs can have different names and definitions, though having the same semantic content, like the example in Figure 2. In the applied model (DistilBERT-SE), name and definition of two SEs are passed through forming sentence embeddings [26] first. In the form of vectors, these embeddings contain the semantic information of the SEs' names and definitions. Using cosine similarity, the similarity of the sentence embeddings is checked and output whether the two SEs are paraphrases, i.e. whether they have the same semantic meaning. In initial studies, this approach was investigated using heterogeneous pump SEs as an example. The results with an accuracy of 94% show that this approach provides first promising results [10].

To improve the results, the DistilBERT [22] model was additionally trained on domain-specific literature [27]. Furthermore, additional attributes (e.g. unit) of SEs in AAS of pumps and HVAC systems

were used to classify the semantics of SEs. A decision tree was implemented for this purpose. The combination of the decision tree and the LM "Eng_DistilBERT-SE" results in the overall model "MetaEng-DistilBERT-SE" (Figure 3), which is described in more detail in [9].

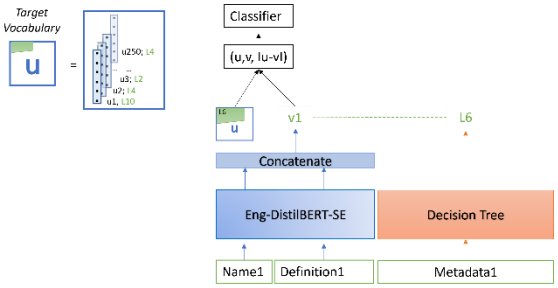

Fig. 3 – The MetaEng-DistilBERT-SE model

This model will be used in the further course of the implementation of the interface.

3. Industrie 4.0 Interface for Semantic Matching

This paper specifies the "Semantic Matching" interface. For this purpose, the first three levels of the I4.0-service model are developed. In addition, it is shown how this can be used within an EMS to enable the automated integration of semantically heterogeneous information into such an EMS.

3.1 Technology-neutral specification

The SM interface is composed of two interface operations. The first operation is called "PostAssetAdministrationShellEmbeddings" (PAASE) (Appendix A). The input to the operation is a component's own AAS. The operation is called from its own AAS and therefore has no output parameter that is returned to another AAS. With the help of this operation, the content of an AAS, specifically the individual SEs of it, passes through the model "MetaEng-DistilBERT-SE" (Figure 4).

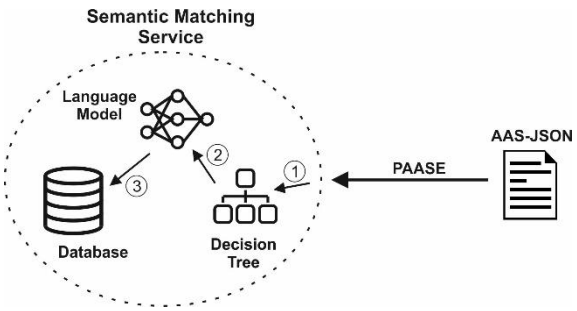

Fig. 4 - Operation PostAssetAdministration-ShellEmbeddings

First, the metadata is processed in the decision tree and assigned to one of the pre-trained classes. The attributes of the SE from Table 1 are first mapped to the parent classes (Watt to Power, Real Measure to Real) and then assigned to class 28 by the decision tree. This class acts as the index of the database to be created. In parallel, the name and definition of the SE run through the "Eng-DistilBERT-SE" model and the sentence embeddings are formed. These embeddings are then concatenated and written to the matching index of the database together with the semantic ID of the SE. This database is initialized and created when the operation is called.

Tab. 1 -Example for an SE from an AAS

AAS Metamodel	**AAS Pump**
Submodel Element	Property
Category	VARIABLE
Qualifier	Operation
Unit	Watt
Data type	REAL_MEASURE
Preferred name	Pump power output
Definition	measured useful mechanical power transferred to the fluid during its passage through the pump
Semantic ID	0173-1#02-ABC172#001

The second interface operation is called "GetAllSubmodelElementsBySemanticIdAndSemanticInformation" (GASEIBSIASI) (Appendix B). The goal of this operation is to return the matching SE of its own AAS (Figure 5).

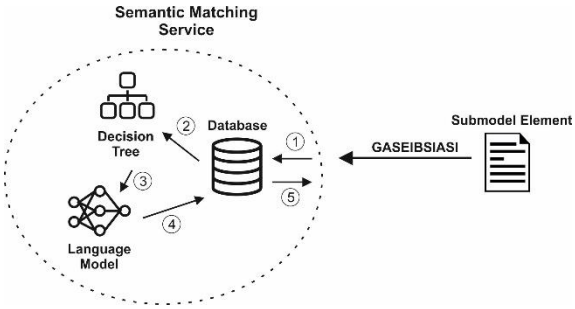

Fig. 5- Operation GASEIBSIASI

The input parameters of this operation are the attributes of the SE to be matched. First, the semantic ID is used to check whether the two AASs use a homogeneous language space. The semantic ID of the SE to be matched is first compared with all semantic IDs in the database of the SEs of the own AAS. If a semantic ID returns a match with the semantic ID to be matched, this SE is returned as matching. Matching based on the other attributes does not need to be performed since the same vocabulary is used. If no match is found, the attributes are passed to the "MetaEng-DistilBERT-SE" model (step 2), classified and the sentence embeddings are formed. According to the class formed on the basis of the meta

information, all embeddings of the own AAS belonging to this class are returned from the database.

Each of these sentence embeddings is then compared to the requested SE using cosine similarity. The SE with the highest similarity is classified as a paraphrase and returned as an output parameter along with the corresponding cosine similarity.

3.2 Technology-specific and implementation level

In the technology-specific level, the generally specified interface SM and the two operations are mapped to a specific API. In this paper, HTTP/REST is used as the technology. The implementation of the interface and the associated operations is carried out using Python. The Python framework FastAPI [28] is used to implement the API. The API provided via FastAPI for calling the two operations is shown in Figure 6.

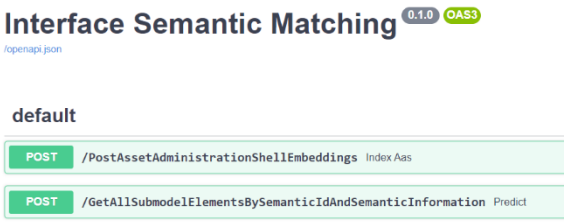

Fig. 6 – Interface Semantic Matching

When an AAS is passed to the PAASE operation, the model is first run and the results are stored in a database [29].

When the operation "GASEIBSIASI" is called, it is first checked with the database whether a common language space is used. If not, the SM is subsequently performed and the result is returned to the requesting AAS. An example of such a call is given in Figure 7.

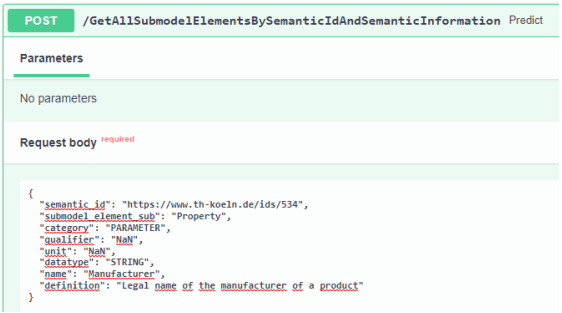

Fig. 7 – Calling the operation GASEIBSIASI

An own SE can be defined via the mask. The necessary attributes for calling the operation must be filled in here. If, as in the example, no unit is available, the value can be left empty, this is intercepted via the function. This request is made to the API and processed. The response is shown in Figure 8.

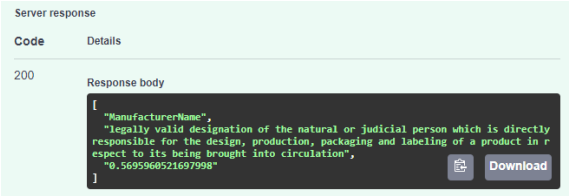

Fig. 8 – Response of the server

The semantics of the requested AAS is based on the ECLASS standard. The SE defined in the mask, however, is not based on any standard. Therefore, a homogeneous language space does not exist. After the implemented model has been run through, the appropriate SE is returned as the response.

The example shows that as a request the name of the manufacturer was asked. Although the requested AAS is based on a different vocabulary and the name and definition do not match the requested name and definition, the matching SE was still returned. Accordingly, the model processed the request correctly and interpreted the semantics of the requested SE correctly.

The example shows the basic flow of the SM. The following chapter describes how this can be integrated into an I4.0-environment in further developments and made available for an automated EMS.

3.3 Integration of the Semantic Matching Service into an I4.0-environment

The interface is deployed as a Docker Image. Docker Images can be downloaded from Docker Hub and used as a Docker Container as one application on one system. All installations required for this application are included on the image. [30]

The SM interface has been implemented as a Docker image so that it can be easily used in an I4.0 environment (Figure 9).

The AAS of the heat pump provides the endpoint for API access to the SM interface to other AASs. The EMS's AAS then sends a request, as in Section 3.2 to the heat pump's AAS, requesting the SE. This AAS has implemented the Docker image as a Docker container within its runtime environment and can thus access the operations of the container. The simple implementation of the container makes it possible to implement this interface within AAS.

If several AAS are implemented by different components in a building, an EMS can now simply be set up. If, for example, energy-relevant variables such as power consumption, power demand, etc. are to be integrated from each component, the individual AAS of the components can be queried for these variables. If positive feedback is received about the existence of these variables, they can be integrated into the EMS. Time-consuming examinations of all SEs of each AAS for the appropriate energy-relevant variables are no longer necessary. The setup of an EMS in a building

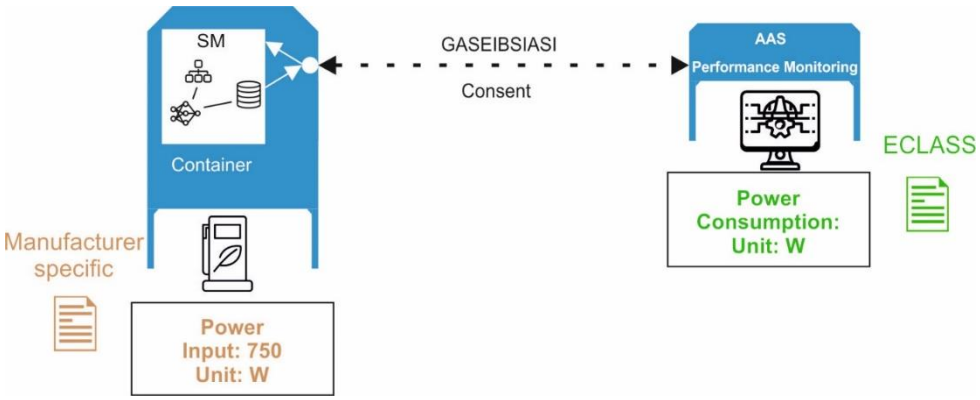

Fig. 9 - Interaction EMS with heat pump based on heterogeneous sematics

is significantly simplified by the SM service.

4. Conclusions

The results extend the current I4.0-research approach to the interaction of semantically homogeneous to semantically heterogeneous AAS. For this purpose, NLP methods were used to implement an automated SM on a target vocabulary. The SM service was specified as an I4.0-interface and is composed of two operations that are called when the interface is used. The interface was expressed and prototyped as an HTTP/REST API. This demonstrated that the interface can be used to process heterogeneous semantics. However, for the interface to be used by AAS, standardization is necessary. For this purpose, the interface will be brought into the corresponding I4.0 working groups to perform a standardization.

EMS are an important factor to increase the energy efficiency of buildings. In order to simplify the

implementation of these and reduce costs, the use of DTs of technical components must be pushed further. In this context, the presented SM service represents a possibility to process heterogeneous semantics of components and thus reduces a potential conflict in the widespread use of EMS. In this paper, the AAS was used as the digital representation. In order to cover further specifications of DTs the SM service must be adapted to their structure.

The datasets generated during the current study are available in the Labor GART repository, https://github.com/thcologne-gart

5. Acknowledgement

The authors gratefully acknowledge financial support from the KSB Foundation in the project Automatic interaction of semantically heterogeneous Industrie 4.0 Asset Administration Shells by means of generic translation mechanisms based on methods of Natural Language Processing (1.1359.2020.1).

6. Appendices

Appendix A

Operation Name	PostAssetAdministrationShellEmbeddings	
Explanation	Takes all submodel elements of the submodels of an Administration Shell, forms embeddings and writes them to a database	
Name	**Type**	**Description**
Input Parameter		
aas	AssetAdministraionShell	AssetAdministration Shell object

Appendix B

Operation Name	GetAllSubmodelElementsBySemanticIdAndSemanticInformation	
Explanation	Returns all submodel elements of the Submodel with a specific Semantic-Id or based on Semantic Matching	
Name	**Type**	**Description**
Input Parameter		
Semantic ID	String	Identifier of the semantic definition
Preferred name	String	Preferred Name of the Submodel Element
Definition	String	Definition of the Submodel Element
Data Type	DataTypeIEC61360	Data type of the Submodel Element (i.e. String, Real etc.)
Submodel Element	String	Subtype of the Submodel Element (i.e. Property, Capability, File etc.)
Category	String	Category of the Submodel Element (Constant, Parameter, Variable)
Qualifier	String	Qualifier of the Submodel Element
Unit	String	Unit of the Submodel Element (i.e. Pascal, Kelvin etc.)
Output Parameter		
Payload	Submodel Element	Matched Submodel Element
Cosine Similarity	Float	The calculated cosine similarity as a measure of the accuracy of matching

References

[1] Walter Kahlenborn, Sibylle Kabisch, Johanna Klein, Ina Richter, Silas Schürmann. Energy Management Systems in Practice: ISO 50001: A Guide for Companies and Organisations; 2012.

[2] Clemens Rohde, Patrick Plötz, Lisa Nabitz, *et al.* Branchen- und unternehmensgrößenbezogene Ermittlung von Klimaschutzpotenzialen (Schwerpunkt KMU) durch verstärkte Umsetzung von Energiemanagementmaßnahmen in der Wirtschaft: Abschlussbericht; 2018.

[3] Web of Things (WoT) Thing Description: W3C Recommendation 9 April 2020; 2020 [cited 2021 November 30] Available from: URL: https://www.w3.org/TR/wot-thing-description/.

[4] Federal Ministry for Economic Affairs and Energy (BMWi). Details Of the Administration Shell: Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 2020.

[5] Federal Ministry for Economic Affairs and Energy (BMWi). Details Of the Administration Shell: Part 2 - Interoperability at Runtime – Exchanging Information via Application Programming Interfaces 2021.

[6] ECLASS e.V. ECLASS: ECLASS - Standard für Stammdaten und Semantik für die Digitalisierung [cited 2021 November 30] Available from: URL: https://www.eclass.eu/index.html.

[7] IEC. IEC Common Data Dictionary [cited 2021 November 30] Available from: URL: https://cdd.iec.ch/cdd/iec61360/iec61360.nsf/TreeFrameset?OpenFrameSet.

[8] buildingSMART. buildingSMART Data Dictionary [cited 2021 November 30] Available from: URL: https://www.buildingsmart.org/users/services/buildingsmart-data-dictionary/.

[9] Both M, Cartus A, Maisch N, Müller J, Diedrich C. Interoperability of semantically heterogeneous digital twins through Natural Language Processing methods. CLIMA 2022.

[10] Both M, Müller J, Diedrich C. Automatisierte Abbildung semantisch heterogener I4.0-Verwaltungsschalen durch Methoden des Natural Language Processing. at - Automatisierungstechnik 2021; 69(11): 940–51 [https://doi.org/10.1515/auto-2021-0050]

[11] Federal Ministry for Economic Affairs and Energy (BMWi). 2030 Vision for Industrie 4.0: Shaping Digital Ecosystems Globally; October 2019.

[12] VDI. Language for I4.0 components: Structure of messages. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA); 2020 04.2020.

[13] VDI/VDE-Gesellschaft. Industrie 4.0

Begriffe/Terms: VDI Statusreport; April 2019.
[14] Federal Ministry for Economic Affairs and Energy (BMWi). Position Paper: Interoperability – Our vision for Industrie 4.0: Interoperable communication between machines within networked digital ecosystems; November 2019.
[15] Both M, Müller J. Digitization of pumps – Industry 4.0 submodels for liquid and vacuum pumps. 4th International Rotating Equipment Conference 2019.
[16] ZVEI - Zentralverband Elektrotechnik und Elektronikindustrie e. V. Antrieb 4.0 – Vision wird Realität: Merkmale, Daten und Funktionen elektrischer Antriebssysteme in Industrie 4.0 für Hersteller, Maschinenbauer und Betreiber; November 2018.
[17] Functional View of the Asset Administration Shell in an Industrie 4.0 System Environment: Discussion Paper; April 2021.
[18] Jurafsky D, Martin JH. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. 2. ed., Pearson internat. ed. Upper Saddle River, NJ: Prentice Hall Pearson Education Internat 2009.
[19] Devlin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding; 2018 Oct 11.
[20] Clark K, Luong M-T, Le V Q, Manning CD. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators; 2020 Mar 23.
[21] Liu Y, Ott M, Goyal N, *et al.* RoBERTa: A Robustly Optimized BERT Pretraining Approach; 2019 Jul 26.
[22] Sanh V, Debut L, Chaumond J, Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter; 2019 Oct 2.
[23] Lewis M, Liu Y, Goyal N, *et al.* BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension; 2019 Oct 29.
[24] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I. Language Models are Unsupervised Multitask Learners Alec; 2019.
[25] Socher R, Huang EH, Pennington J, Ng AY, Manning CD. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In: Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection; 2011. Red Hook, NY, USA: Curran Associates Inc; 801–9.
[26] Reimers N, Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks; 2019 Aug 27.
[27] Lo K, Wang LL, Neumann M, Kinney R, Weld D. S2ORC: The Semantic Scholar Open Research Corpus. In: Jurafsky D, Chai J, Schluter N, Tetreault J, editors. S2ORC: The Semantic Scholar Open Research Corpus. Stroudsburg, PA, USA: Association for Computational Linguistics; 4969–83.
[28] FastAPI: FastAPI framework, high performance, easy to learn, fast to code, ready for production [cited 2021 December 3] Available from: URL: https://fastapi.tiangolo.com/.
[29] Elasticsearch: Das Kernstück des kostenlosen und offenen Elastic Stack [cited 2021 December 3] Available from: URL: https://www.elastic.co/de/elasticsearch/.
[30] Docker Docs [cited 2021 December 3] Available from: URL: https://docs.docker.com/.