8

IoT Things to Service Matchmaking at the Edge

Nisrine Bnouhanna[1] and Rute C. Sofia[2]

[1]Technical University of Munich, Germany
[2]fortiss Research Institute of the Free State of Bavaria for Software Intensive Systems and Services – affiliated with the Technical University of Munich, Germany
E-mail: bnouhanna.nisrine@gmail.com; sofia@fortiss.org

Abstract

This chapter debates on the use of Machine Learning (ML) to support edge-based semantic matchmaking to handle a large-scale integration of IoT data sources with IoT platforms. The chapter starts by addressing the interoperability challenges currently faced by integrators, the role of ontologies in this context. It continues with a perspective on semantic matchmaking approaches, and ML solutions that can best support a cognitive matchmaking. The chapter then covers a use case and pilots that are being developed with a new open-source middleware, TSMatch, in the context of the Horizon 2020 EFPF project, for the purpose of environmental monitoring in smart manufacturing.

Keywords: Machine learning, semantic technologies, matchmaking.

8.1 Introduction

Manufacturing environments are becoming increasingly digitized to improve the overall process and business efficiency. Sensors and actuators (Internet of Things devices) are heavily integrated into manufacturing environments,

and interconnected to edge and cloud via heterogeneous communication protocols, such as the Message Queuing Telemetry Transport (MQTT) [1] or the Open Platform Communications Unified Architecture (OPC UA) [2], [3]. The integration and discovery of sensors and their interconnection to the edge or cloud implies heavy human intervention, being error prone. Moreover, as devices within an devices within an industrial environment are often acquired from different vendors, interoperability at different levels (device, protocol, domain, etc.) [4] is a significant issue in IoT.

Semantic sensor technologies [4], such as the Web of Things (WoT), provide a way to define IoT devices via expressive, uniform descriptions of metadata and data meaning, thus assisting in lowering the barrier of interoperability between IoT devices and IoT platforms.

Similarly, IoT services can be seen as a subset of web-based services, and, therefore they can be described via semantic technologies.

Therefore, describing semantically IoT devices and services is a methodology that provides a way to lower the interoperability barrier. However, semantic data annotation is still being tied to specific protocols. For instance, OPC UA specifies robotics specifications, which are not necessarily compatible with the specifications provided by other protocols.

Semantic matchmaking can assist in bringing this level of automation to IIoT. In this context, semantic matchmaking relates with using the meaning and information content provided by IIoT device descriptions (Things Descriptions (TD)) to match it with the meaning of offered IIoT services, e.g., environmental monitoring, abnormal pattern detection, etc.

Currently, semantic matchmaking requires the use of ontologies to discover semantic similarity between the two semantic descriptions – in the case of this work, Thing and Service – to detect the "semantic distance" between the two elements.

Ontologies are therefore a key component of semantic interoperability, as they provide the foundation and capability for devices to interpret and infer knowledge from datasets. However, the application of ontologies is complicated due to three major problems: i) fragmentation and vendor-lock; ii) cross-domain interoperability; iii) lack of open tools and application examples [5]. Fragmentation may be reduced via the development of information models that are universal, based on open standards, such as the ETSI Smart Applications Reference Ontology (SAREF) [6], and not based on specific protocols or vendors. Cross-domain interoperability requires a new approach to ontologies, in particular, the application of a universal language and a universal universal approach that can assist the mapping between domain-based

ontologies, such as that happening with the methodology of the Industrial Ontology Foundry-core (IOF-core)[1] for industry, and the core ontology for biology and biomedicine (COB)[2] for the biomedical domains. SAREF is a key reference in this context.

While ontologies provide an easier way to interpret data across IIoT infrastructures, the need for a higher degree of automation still persists.

The focus of this chapter, therefore, relates with a debate on the definition and use of semantic matchmaking in IIoT environments, to improve the overall interoperability. The main contributions of this chapter are as follows:

- To provide an understanding of semantic matchmaking and the underlying semantic technologies that are applied in the context of IIoT environments.
- To explain the current challenges and propose guidelines to circumvent them.
- To explain how semantic matchmaking can be applied in the context of edge−cloud environments.

The chapter is organized as follows. After this introductory section, Section 8.2 provides background on semantic matchmaking. Section 8.3 presents a specific applicability case derived from the application of an open-source semantic matchmaking middleware, TSMatch. Section 8.4 explains the current challenges faced when applying semantic matchmaking between IoT Things and services. Section 8.5 debates on proposals to further support an intelligent, adaptive, and semantic matchmaking for IIoT. Section 8.6 concludes the document.

8.2 Semantic Matchmaking and Current Approaches

In general, semantic matchmaking refers to the mapping between two concepts, entities, or descriptions focusing on how similar the semantic meaning of the matched concepts is, while identifying the relationship between them [6].

Semantic matchmaking is used in various fields such as web services [7], [8], information retrieval [9], [10], and in various vertical domains such as vertical domains such as smart cities [11] or health [12]. A typical application of semantic matchmaking in the context of web services

[1] https://industrialontologies.org
[2] https://obofoundry.org/ontology/cob.html

is service composition, where several existing web services are combined using semantic matchmaking between the services' descriptions, to provide enriched descriptions of services. There are also several scenarios where semantic matchmaking is applied to retrieve information. The most common is web search engines and recommendation systems. Semantic matchmaking has also been used in vertical domains as an interoperability solution to enable communication between various entities.

Semantic matchmaking is especially useful for large-scale IoT environments, since the large number of connected devices/Things need to effectively communicate between each other and with other services, to reach its full potential. However, applying semantic matchmaking to IoT needs to be adjusted to the specific conditions and needs of IoT environments, e.g., the diversity of attributes for sensors, the different units applied, etc. An additional challenge to address in the context of IoT environments is the matching of devices/Things across different vertical domains, as fine-grained matching is required. For example, semantic matchmaking should be able to differentiate between a temperature sensor of an environment and a temperature sensor of a machine and accordingly match the adequate sensor to the service request.

There are several methods and technologies used to achieve semantic matchmaking depending on the scenario, requirements, and the type of entities to be matched. Overall, semantic matchmaking can be categorized in three main approaches: knowledge-based, statistical, and hybrid [19].

Knowledge-based approach refers to using a predefined knowledge containing statements such as rules, facts, and constraints to provide a semantic match between entities. A common example of such an approach is ontology-based matching, where reasoning is used to find similarities and relations between the semantic description entities.

Knowledge-based semantic matchmaking approaches tend to be accurate and provide fine-grained matching since they are based on pre-built and expert knowledge and models combined with logical reasoning. However, they may lead to false negatives caused by the limitations of the knowledge used. For example, if two concepts are semantically synonymous but defined differently in their terminological definitions, the similarity between the two is not captured and a reasoner would fail to find the match between the two concepts. Moreover, knowledge-based semantic matchmaking approaches are complex, require long design time, demand high maintenance to keep the knowledge-base up to date, and are associated with long processing time [16], [17].

A **statistical approach** is based on analyzing the frequency of occurrence of certain terms used in the semantic descriptions of the entities to be matched and use statistical tools such as lexical resources (e.g., WordNet[3])/distance measures (e.g., cosine similarity) or ML (e.g., clustering techniques [13]) to define their semantic similarity. Statistical-based semantic matchmaking is considered to be less complex compared to the knowledge-based approach; it also tends to require less processing time due to less complex computation. However, for applications that require fine-grained matching, statistical-based approaches are usually not suitable since the processing and transformation of the data causes loss of semantic information, thus leading to more generic matching.

Hybrid approach refers to semantic matchmaking solutions that combine both knowledge-based and statistical approaches to mitigate the advantages and disadvantages of both approaches [14], [15]. Hybrid approaches may assist in overcoming the disadvantages of both the knowledge-based and statistical categories. However, identifying the effective way to combine and take advantage of both techniques remains a challenge.

8.3 TSMatch, an Example of Semantic Matchmaking for IIoT

TSMatch [22] is an open-source middleware[4] that supports semantic matchmaking between IoT data sources (Things) and IoT services. TSMatch contributes to solve the challenge of semantic interoperability, by providing an automated matchmaking solution between IoT devices and IoT services, which relies on semantic technologies. The proposed solution is based on the following two assumptions: i) each IoT device has a semantic description; ii) each IoT service can be described semantically based on an ontology.

The TSMatch middleware has been developed and applied in industrial environmental pilots (TRL6) in the context of the **Horizon 2020 European Connected Factory Platform for Agile Manufacturing (EFPF)** project[5], and a demonstrator is available and interconnected to the EFPF data spine via the fortiss IIoT Lab.

[3] https://wordnet.princeton.edu/

[4] https://git.fortiss.org/iiot_external/tsmatch

[5] https://www.efpf.org/

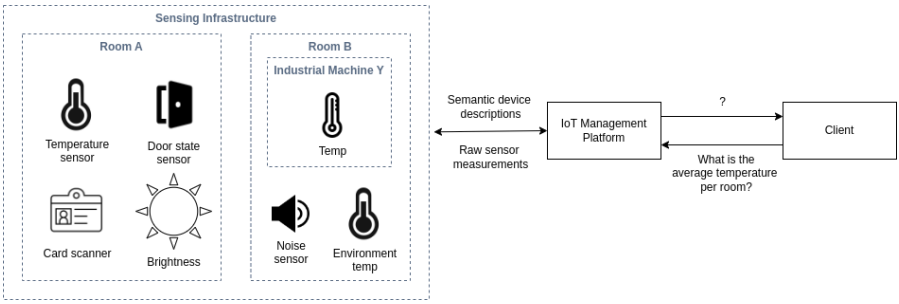

Figure 8.1 Example of a smart facility interconnected to an IoT platform.

As an example that may assist in understanding the operation of semantic matchmaking, let us consider Figure 8.1 standing for a smart factory shopfloor, integrating multiple sensors.

In the shopfloor, machines from different vendors, with coupled sensors attached on them, are expected. Moreover, sensors are also used to monitor the environment, e.g., CO_2, temperature, humidity, etc. Employees of this facility are using wearable devices, tablets, and smart phones, which also have sensing capabilities. In this scenario, different IoT platforms have been acquired to different vendors. Therefore, each platform considers different semantic standards to support an interoperable data exchange. Data exchange is supported by a data bus across the factory, and the different platforms rely on specific communication protocols to exchange data, e.g., OPC UA, MQTT Sparkplug, etc. Different services, e.g., data analytics tooling, environmental monitoring services, and certification services, are interconnected to the data spine via software-based connectors that have been specifically devised for this purpose, by the different vendors, or by an integrator.

Some of these services run on the so-called edge (close to the field-level devices, e.g., on the shopfloor) and others run on the cloud. On this scenario, the semantic matchmaking process can occur on the cloud or on the edge. Placing the matchmaking on the edge is expected to lower latency and also reduce energy consumption, as most of the data processing (including aggregation) is performed closer to the end-user.

TSMatch (rf. to Figure 8.2) aims at providing this type of support, being developed to run as an edge-based service. Following a client-server approach and consisting of multiple containerized microservices, TSMatch comprises a server-side, **the TSMatch engine**, and a **TSMatch client**. The TSMatch engine is composed of two main functional blocks and several interfaces:

- **Semantic matchmaking**. Performs semantic matchmaking between IoT Things descriptions (stored on a database) and ontologies. The result is a set of enriched data nodes, which are also stored in a database.
- **Data aggregation**. Sensor data aggregator.
- **Ontology interface**. Provides support for ontologies to be imported into TSMatch.
- **Connectors**. Different connectors, e.g., Mosquitto to RabbitMQ; HTTP/REST, etc.

The input and output of the matchmaking and data aggregation processes are stored on a local Neo4J database, storing Things descriptions, service descriptions, ontologies, and new data nodes (aggregated Things based on a category, e.g., temperature measurement).

The end-user interacts with the TSMatch engine via an Android app (TSMatch client). Moreover, TSMatch relies on the following external components:

- **An MQTT broker.** TSMatch currently relies on an MQTT broker based on Mosquitto as message bus. The TSMatch client and engine interconnect to the Thing discovery: IoT Thing discovery is supported via Coaty.io[6].
- **Service registry**. Holds a set of service descriptions. Currently holds environment monitoring service specification examples based on OWL and WSDL, which the user can select via the TSMatch client.

8.3.1 Setup

The TSMatch operation considers two phases. During setup, TSMatch performs discovery of existing IoT devices via the coaty.io open-source middleware, and ontologies can also be imported.

For discovery, it is assumed that IoT devices have an integrated coaty agent or are interconnected to a hub that holds such agent. Therefore, when an IoT device boots up or becomes active after a period of inactivity, it publishes its TD via coaty. These TDs are stored on the local database co-located to the TSMatch engine. The semantic matchmaking module subscribes to Thing discovery events. When a new TD is received, or when there is a change in the TD, then the semantic matchmaking process computes the new data elements

[6] https://coaty.io/

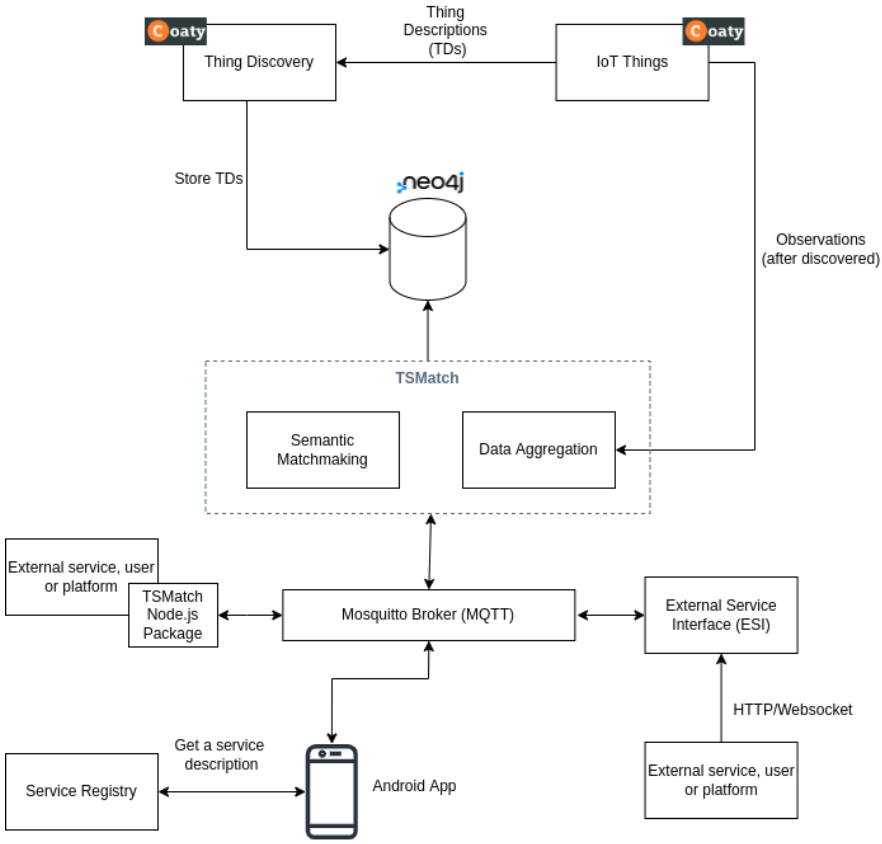

Figure 8.2 High-level perspective of the TSMatch architecture.

as described in the next subsection and stores the new data nodes on the local database (graphDB).

8.3.2 Runtime

Before running the semantic matchmaking algorithm, the TD files as well as the names of the ontology elements are pre-processed to clean up the data, i.e., to perform tokenization, remove punctuation, etc. After the pre-processing step, the TDs are passed to one of the selected semantic matchmaking algorithms. The algorithm matches the given TDs to the ontology elements. Then a relation is created in the graph database between each TD and the ontology nodes that it is being compared with.

```
...
"sensor": {
    "name": "LightIntensity",
    "description": "Light intensity in Lux",
    "type": "object",
    "readOnly: "true",
    "properties": {
        "LightIntensity": {
            type": "number",
            "readOnly": true
        }
    }
    ...
}
...
```

Figure 8.3 Description of a service enriched with information from the FIESTA-IoT ontology.

An example of a TD description is provided in Figure 8.3, where the "name" and "description" attributes (first level attributes) will be used by the algorithm to perform a check against ontologies' categories. The algorithm performs multiple interactions based on a depth search on the respective provided ontology/ies.

The matchmaking relies on a Natural Language Processing (NLP) neural network-based approach that has been compared with i) a statistical approach, based on sentence similarity; iii) a clustering-based approach. The NLP neural network model-based approach (W2VEC) has been tested and shown to achieve better results in comparison to a clustering approach derived from K-means, and to a cosine similarity approach [23][7,8].

After matches are found, the relations between the nodes representing sensor descriptions and specific categories of ontologies are created.

The third step on the algorithm concerns data aggregation. Upon receiving a service request, the data aggregation module checks for the aggregated TD nodes on the database, subscribes (via MQTT) to data from the respective

[7] https://github.com/fiesta-iot/ontology.

[8] https://git.fortiss.org/iiot_external/tsmatch/-/tree/master/dataset/ontology

sensors, and performs data aggregation based on a simple average function. Based on the service example described in Figure 8.3, the algorithm discovers TDs stored in the database with a stored relation to "Illuminance," "Lux," "LightSensor," and "Environment."

Ontology nodes also on the database. It sends a response back to the requesting service (on the TSMatch client), providing information about the sensors, based on the TDs as shown in Figure 8.4. Moreover, the algorithm gets data from the respective IoT devices (via MQTT, as a subscriber) and then performs data aggregation (simple average) and periodically sends the results to the requesting service, as shown in Figure 8.5.

Figure 8.4 TSMatch client, interface that obtains the available descriptions of existing sensors in an infrastructure in real time.

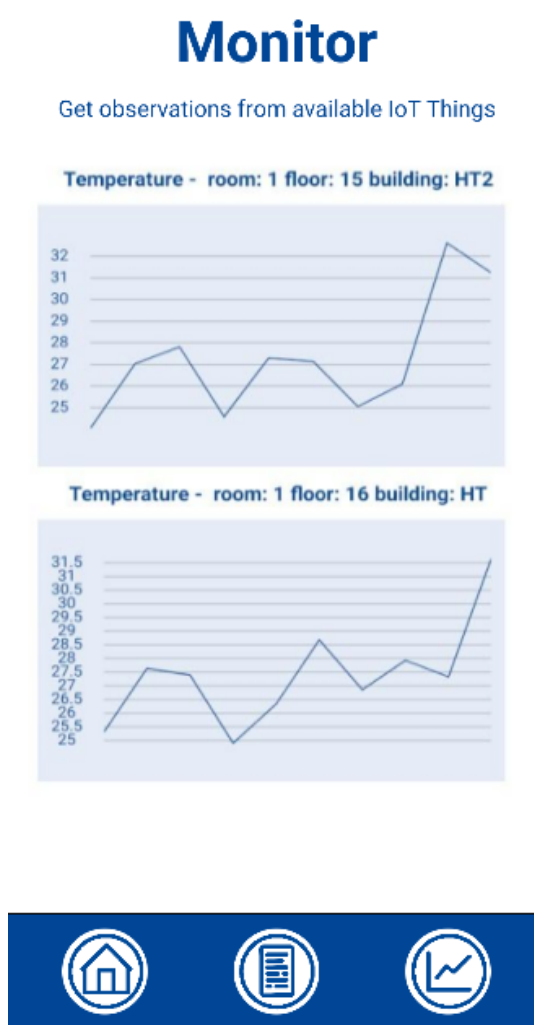

Figure 8.5 TSMatch, temperature monitoring, aggregated results derived from the TSMatch ML-based matchmaking process.

8.4 Semantic Matchmaking Challenges for IoT

As mentioned earlier, semantic matchmaking can be applied to various fields including IoT; however, with each field of application, new requirements and challenges are to be considered. Based on the scenario described earlier, the

objective is to semantically match the semantic description of IoT Things with Services, to achieve a finer data matching, and to be able to provide enriched data-based services. Besides providing semantic matchmaking between the IoT Things attributes (e.g., temperature attribute) and the service input signature, other aspects need to be considered, as follows:

- **Integration of functional and non-functional requirements**. To ensure a match that supports interoperability between the IoT Things and the services, non-functional requirements beyond security may be relevant to be integrated. An example of a non-functional requirement could be data compliance aspects, for instance.
- **Integration of user requirements**. Users in this scenario can be the manager of the facilities where the IoT Things are located. Such user requirements can be provided in the form of user preferences (e.g., preference in terms of non-functional service requirement, for instance, cost), but also in the form of quality of experience (QoE) feedback (e.g., level of satisfaction with the outcome of a specific match process). This way, the user can put constraints on the type of IoT data to be utilized and shared with services or specify the bandwidth constraints that may impact the frequency and data rate.
- **Integration of context-awareness**. The aim is to consider the surrounding context of Things (e.g., room temperature for a specific sensor installed on a machine in a room) together with Things attributes (e.g., temperature provided by the sensor itself). This would provide fine-grained and more accurate matching since it allows the differentiation between sub-types and application of IoT Things.
- **Integrate in the semantic approach design energy awareness and processing time reduction**. To meet far edge constraints, use semantic approaches that reduce the processing time (e.g., real-time requirements) and power consumption, among other aspects.
- **Interoperability across domains**. IoT is applied to various vertical domains. In many IoT scenarios, communication and integration across domains is required; hence, it is necessary to reduce the limitation associated with using a knowledge-based approach mainly due to the risk of having incomplete and complex models that require high maintenance.
- **Integration of a feedback loop to the user, to improve QoE**. Provide useful feedback to both the user and the service about the matching process (e.g., ranking of matches, information about the criteria selected for the match).

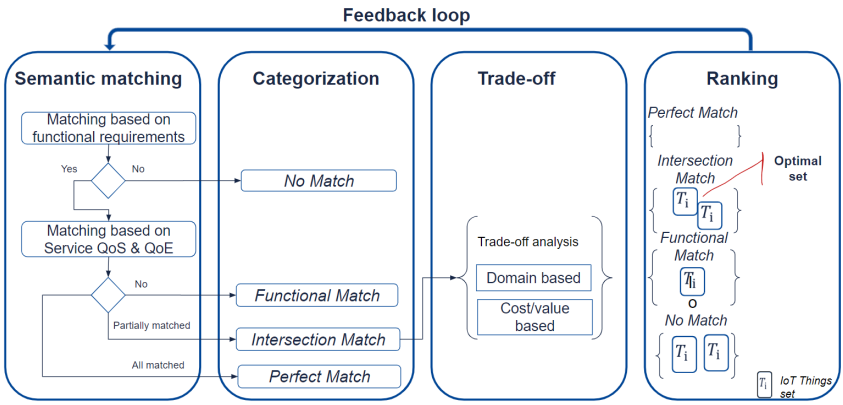

Figure 8.6 IoT Things to service semantic matchmaking approach.

- **Integration of a learning approach**. IoT solutions are constantly evolving; hence, building a matching solution that is able to learn from the previous matches to improve future matches is key.

8.5 Evolving Semantic Matchmaking at the Edge

To fulfill the requirements of semantic matchmaking between existing IoT Things and services at the edge, we propose the following five-step approach as illustrated in Figure 8.6, which are further described in the next subsections.

1. **Hybrid semantic matchmaking**: After discovering the available IoT Things descriptions and receiving the service semantic request, use a hybrid semantic approach to match both the functional and non-functional requirements of the service with the IoT Things while considering user requirements (e.g., QoE) as constraints.
2. **Categorization**: Group IoT Things based on the semantic matchmaking results, to reflect the degree of matched requirements.
3. **Tradeoff**: In case of a partial match of functional and non-functional requirements of the service, use tradeoff analysis to optimize for a specific goal to support the identification of an optimal set of IoT Things.
4. **Ranking**: Considering the results of the categorization and the tradeoff analysis, rank IoT Things sets and highlight the match criteria and optimization goal to the user and the IoT service to support further decisions.

5. **Feedback loop**: Use previous matched results to learn and optimize the hybrid semantic matchmaking and tradeoff analysis.

8.5.1 Hybrid semantic matchmaking

Hybrid semantic matchmaking brings together the advantage of both knowledge-based and statistical categories. A novel approach also needs to fulfill the requirements described in Section 8.3. Such a hybrid approach is based on an algorithm that takes into consideration the novel aspects of integrating IoT service requirements, IoT user requirements, and IoT Things attributes.

Such an automated semantic matchmaking algorithm needs to be able to find similarities between criteria required by the service, the user, and the features of the existing IoT Things.

First, the main parameters to be matched for the service are the service functional requirements meaning the service requested IoT input data and the service QoS, for example, sampling rate, resolution, or delay of the IoT data. Second, the user requirements or QoE refer to the user criteria to accept/be satisfied with the provided functionality and integration between the IoT Things and the service. Such a user could be the manager of the environment hosting the IoT Things. For instance, a user can specify requirements regarding the network usage, IoT Things energy consumption in case of battery-operated IoT Things, or the type of data shared, e.g., public (i.e., services requesting such data can have access without permission), private (i.e., requires permission from the user), or restricted data (i.e., cannot be shared with external services). Finally, some of the key aspects of the IoT Things descriptions to be considered in the semantic matchmaking are as follows:

- The IoT Things observed property, for example, temperature, occupancy, and presence.
- The IoT Things observation type, which refers to the value type created based on environmental stimuli, for example, 27°C.
- The IoT Things feature refers to the specific feature we are observing or measuring its property, for example, environment, machine, human, etc.
- The IoT Things spatial property is the area of observation or location, for instance, a factory in a specific city.
- The IoT Things capabilities group the set of specifications that describe aspects of the provided observations such as range and accuracy.

Now that we identified the various concepts to be matched, the next step is to use hybrid semantic matchmaking to identify which IoT Things

fulfill the needed requirements. Thus, the main phases of the hybrid semantic matchmaking approach are illustrated in Figure 8.7.

As a first step, IoT Things are discovered using an IoT crawler or a web of things search engine (WoTSE). These tools can assist in automatically identifying existing IoT devices in an infrastructure, and in storing their TDs [18]. Then, relevant concepts for matching are retrieved from the descriptions, e.g., IoT Thing name, observed property, feature of interest, spatial property, capabilities, pre-processed, stored, accuracy, sampling rate, etc. This is an ongoing process to update, remove, or add IoT Things information to reflect the current state of the IoT environment.

The second step aims at organizing and grouping the IoT Things information. The goal is to match the incoming service requests with a subset of IoT Things relevant for the service instead of the full set to reduce processing time and computations. Different machine learning methods can be used for that purpose, for instance, clustering and categorization methods. Clustering algorithms are unsupervised learning techniques used to group similar data points without prior knowledge of the groups, while categorization algorithms are a supervised learning technique where a model is trained to predict the category/class based on labeled data. In IoT scenarios, the categories are not predefined and depend on the set of IoT Things available in an environment. Creating a fine list of IoT Things categories would be challenging and would need to be updated constantly to reflect the constant development of new

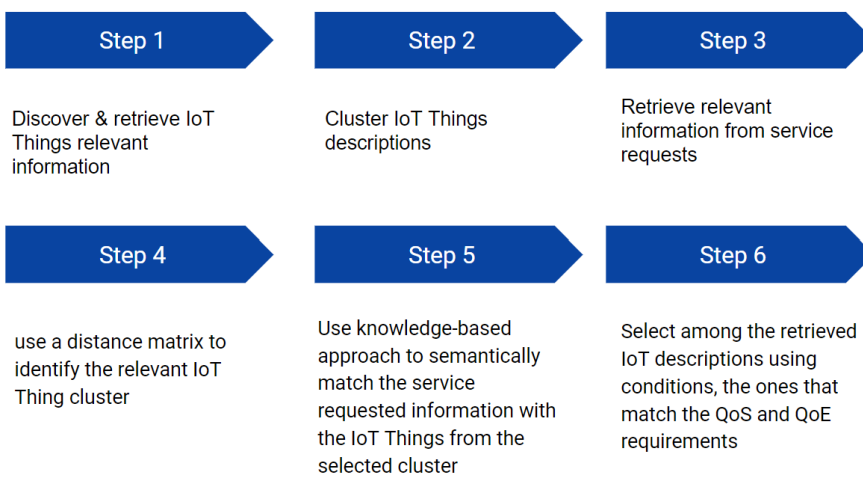

Figure 8.7 Steps of the proposed hybrid semantic matchmaking approach.

solutions. Thus, unsupervised learning techniques such as clustering methods seem to be more suitable for grouping a set of IoT Things without prior knowledge of the groups and classes available in the environment. Different clustering methods can be selected, such as centroid-based clustering (e.g., *k*-means), ensity-based clustering (e.g., Density-based Spatial Clustering of Applications with Noise (DBSCAN)), hierarchical clustering (e.g., agglomerative hierarchical clustering), or distribution-based clustering (e.g., Gaussian mixture model), to name a few.

Selecting the adequate unsupervised learning technique would need to consider the following requirements:

- IoT Things information is non-numerical data that may have different dimensions and incomplete information.
- Small number of IoT Things in the environment.
- Limited computation power on the edge.
- Varying number of clusters depending on the IoT Things available in the environment.

A potential unsupervised learning technique to investigate is hierarchical clustering since it is known to handle non-numerical data with different dimensions and incomplete data. It does not require a predefined number of clusters, can determine the number of clusters based on the data, and is able to work with a small amount of data. However, the main disadvantages of hierarchical clustering are its high computational complexity and lack of scalability, especially for large amounts of data. Hence, different clustering methods might be used depending on the scale of the IoT environment in consideration.

The third step relates to retrieving and pre-processing relevant information from the service, e.g., service description text, location, domain, service input, and required QoS. As the fourth step, the algorithm can use a distance (similarity) matrix to match the service-requested information (functional requirements) to the centroid of each cluster to identify the most relevant cluster of IoT Things. In step 5, the algorithm would apply a knowledge-based approach to semantically match the service-requested information with the selected cluster. Depending on the specific identified cluster domain, a specific ontology could be considered, or a standard (such as SAREF) may assist in interoperability across different domains/different ontologies.

Finally, a rule- and condition-based approach is used to identify if the selected IoT Things meet the required QoS requirements of the service and the user QoE. Thus, the input from the user and the requirements from the

service should be extracted based on which the set of rules are developed. Then, performance data from the IoT Things are analyzed against these rules to determine if they meet the service requirements. It is important that the IoT Things performance data are updated over time along with an ongoing evaluation to assure that these requirements are met over time.

The proposed hybrid semantic approach has the advantage of using clustering techniques and similarity matrix to reduce the problem space of the ontology matching, leading to less processing time while maintaining fine-grained matching. Moreover, existing standard ontologies can be automatically selected and used depending on the scenario, hence removing the need to develop and maintain a cross-domain ontology. Different concepts from different stakeholders are included in the matching process to ensure a comprehensive IoT Thing to service matching solution. However, the matching accuracy is influenced by the clustering algorithm, which may cause false positives and the selected ontology form the knowledge-based matching ; therefore, it is important to use the feedback loop to help automatically adjust the clustering parameters and select the adequate ontology to improve the precision.

8.5.2 Categorization

Based on the semantic matchmaking, the next stage is to categorize and group the IoT TDs based on the level of identified matches. Therefore, we propose the following categories, which can be derived from the results of the hybrid semantic matchmaking process.

- **Exact match**: When all three selection criteria are met, which include the IoT service required IoT data input, QoS, and the user QoE.
- **Functional match**: When only the functional requirements are matched. For example, a set of IoT Things that can provide the environmental temperature at location A is found; however, they do not meet both the user QoE and the IoT service QoS.
- **Intersection match**: Refers to having a functional match plus partially matched QoS and QoE requirements. This is a challenging category since IoT Things falling in this category would need to be ranked. For example, if one set of IoT Things fulfills all functional requirements plus two service QoS and one user QoE while another set meets all functional requirements plus one (different) IoT QoS and one (different) user QoE, how can both sets be compared?

- **No match**: When the service functional requirements cannot be matched.

8.5.3 Tradeoff

For the intersection match category, a further analysis is required to rank the set of IoT Things that fall into this group. The analysis should enable the comparison between different QoS and QoE requirements to permit ranking. One option is to use tradeoff analysis to serve a specific optimization goal. The optimization goal can be either defined by the user or derived from the operation domain and adjusted using the feedback loop. A domain-based tradeoff analysis, for instance, would optimize to fulfill the domain-specific requirements; therefore, the requirements relevant to the domain in question have higher value and hence higher ranking.

Using the categorization of IoT Things combined with the tradeoff analysis matching, the goal is to identify the optimal set of IoT Things among available ones by ranking them and providing useful information to the service and the user regarding the selection criteria such as the list of matched concepts and the optimization goal used. We proposed the following ranking classes:

- **Perfect match**: Provide a list of IoT Things that match perfectly functional, QoE, and QoS requirements.
- **Intersection match**: If no-perfect match is not available, select a set of IoT Things that match the functional requirements as well as partial QoS and QoE requirements. This ranking is affected by the tradeoff process since it creates various IoT Things selections, taking into consideration the optimization goal of the tradeoff analysis.
- **Functional match**: A set of IoT Things that only fulfill the functional requirements.

8.5.4 Feedback Loop

Based on the IoT service selected/used set of IoT Things compared to the identified optimal set of IoT Things, a feedback loop can be established to enable learning. Various information can be adjusted such as automatically associating a tradeoff goal to a specific domain, identifying relevant QoS and QoE for a specific IoT domain, adjusting the clustering algorithm's parameters, re-evaluating the similarity measures used, or reducing the processing time by focusing on specific aspects most relevant for the service category.

8.6 Conclusion

Semantic matchmaking provides a way to tackle some interoperability challenges in IoT environments, via the use of semantic definitions of sensors, actuators, machines, and IoT services. Based on standardized approaches, and on the use of supervising-based approaches, it is feasible to reduce the operational cost of setting up and maintaining large-scale IoT infrastructures. While there are solutions, such as TSMatch, which perform such matchmaking already with some degree of freedom, the increasing variety of vendor-based approaches across different vertical domains require an approach that considers unsupervised learning.

Hence, in this chapter and to further provide an answer to the challenges of semantic matchmaking between IoT Things and services on the edge, a five-step approach is proposed. The approach includes using a hybrid semantic matchmaking to match between the IoT Things extracted information and the service requests, categorize the matching results based on the completeness of the match, use tradeoff analysis to decide on the ranking of partial matches, rank the IoT Things subset from no-match to perfect-match, select the optimal IoT Things subset that meets the service request and finally use a feedback loop to improve the process. Overall, the approach aims to optimize on the edge identification of an optimal set of IoT Things to fulfill the requirements of the service including its quality of service while considering the user requirements and the specifications of IoT scenarios and use cases as constraints.

Acknowledgements

The work described in this chapter has been partially supported by the H2020 projects EU-IoT project funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement no. 956671 and H2020 DT-ICT-07-2018-2019 project "European Connected Factory Platform for Agile Manufacturing" (EFPF) under Grant Agreement no. 825075.

References

[1] Sebastian Raff. The MQTT Community. Available online: https://github.com/mqtt/mqtt.github.io/wik.

[2] OPC Foundation. OPC Unified Architecture: Interoperability for Industrie 4.0 and the Internet of Things. pp. 1-44. Available online: https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf

[3] Leitner, S.-H.; Mahnke, W. OPC-UA/Service-Oriented Architecture for Industrial Applications; Technical Report; ABB Corporate Research Center: 2006.

[4] Gergely Marcell Honti, Janos Abonyi, "A Review of Semantic Sensor Technologies in Internet of Things Architectures", *Complexity*, vol. 2019, Article ID 6473160, 21 pages, 2019. https://doi.org/10.1155/2019/6473160

[5] EU-IoT OntoCommons Ontological Interoperability report.

[6] Giunchiglia, F., & Shvaiko, P. (2003). Semantic matching. The Knowledge Engineering Review, 18(3), 265-280.

[7] Ferrara, A., Montanelli, S., Noessner, J., & Stuckenschmidt, H. (2011, May). Benchmarking matching applications on the semantic web. In Extended Semantic Web Conference (pp. 108-122). Springer, Berlin, Heidelberg.

[8] Gmati, F. E., Ayadi, N. Y., Bahri, A., Chakhar, S., & Ishizaka, A. (2016, April). Customizable Web services matching and ranking tool: Implementation and evaluation. In International Conference on Web Information Systems and Technologies (pp. 15-36). Springer, Cham.

[9] Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., & Cheng, X. (2016, March). A deep architecture for semantic matching with multiple positional sentence representations. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 30, No. 1).

[10] Wang, J., Pan, M., He, T., Huang, X., Wang, X., & Tu, X. (2020). A pseudo-relevance feedback framework combining relevance matching and semantic matching for information retrieval. Information Processing & Management, 57(6), 102342.

[11] Ru, L., Zhang, B., Duan, J., Ru, G., Sharma, A., Dhiman, G., ... & Masud, M. (2021). A detailed research on human health monitoring system based on internet of things. Wireless Communications and Mobile Computing, 2021.

[12] Hasan, S., & Curry, E. (2014). Approximate semantic matching of events for the internet of things. ACM Transactions on Internet Technology (TOIT), 14(1), 1-23.

[13] Akritidis, L., Fevgas, A., Bozanis, P., & Makris, C. (2020). A self-verifying clustering approach to unsupervised matching of product titles. Artificial Intelligence Review, 53(7), 4777-4820.

[14] Cassar, G., Barnaghi, P., Wang, W., & Moessner, K. (2012, November). A hybrid semantic matchmaker for IoT services. In 2012 IEEE International Conference on Green Computing and Communications (pp. 210-216). IEEE.

[15] Klusch, M., Fries, B., & Sycara, K. (2006, May). Automated semantic web service discovery with OWLS-MX. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (pp. 915-922).

[16] Otero-Cerdeira, L., Rodríguez-Martínez, F. J., & Gómez-Rodríguez, A. (2015). Ontology matching: A literature review. Expert Systems with Applications, 42(2), 949-971.

[17] Shvaiko, P., & Euzenat, J. (2011). Ontology matching: state of the art and future challenges. IEEE Transactions on knowledge and data engineering, 25(1), 158-176.

[18] F. Skarmeta, J. Santa, J. A. Martínez, J. X. Parreira, P. Barnaghi, S. Enshaeifar, M. J. Beliatis, M. A. Presser, T. Iggena, M. Fischer et al., "Iotcrawler: Browsing the internet of things," in 2018 Global Internet of Things Summit (GIoTS). IEEE, 2018, pp. 1-6.

[19] Fenza, Giuseppe, Vincenzo Loia, and Sabrina Senatore. "A hybrid approach to semantic web services matchmaking." International Journal of Approximate Reasoning 48.3 (2008): 808-828.

[20] N. Bnouhanna, E. Karabulut, R. C. Sofia, E. E. Seder, G. Scivoletto and G. Insolvibile, "An Evaluation of a Semantic Thing To Service Matching Approach in Industrial IoT Environments," 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 2022, pp. 433-438, doi: 10.1109/PerComWorkshops53856.2022.9767519. Pisa, Italy.

[21] E. Karabulut, N. Bnouhanna, R. C. Sofia, ML-based data classification and data aggregation on the edge. inProc. ACM CoNext2021, December 2021, Munich, Germany. Student poster. https://doi.org/10.1145/3488658.3493786.

[22] N. Bnouhanna, R. C. Sofia and A. Pretschner, "IoT Thing to Service Matching" 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events

(PerCom Workshops), 2021, pp. 418-419, vol 1, pp 418-419, DOI; 10.1109/PerComWorkshops51409.2021.9431128

[23] E. Karabulut, R. C. Sofia, J. Ott, https://www.overleaf.com/project/61dff801dd153d87a37f1460.Masterdissertation,2022.Shortversionundersubmission(2023).