UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

An Analysis of Machine Learning-Based Semantic Matchmaking

Karabulut, E.; Sofia, R.C.

DOI
10.1109/ACCESS.2023.3259360

Publication date
2023

Document Version
Final published version

Published in
IEEE Access

License
CC BY

Link to publication

Citation for published version (APA):
Karabulut, E., & Sofia, R. C. (2023). An Analysis of Machine Learning-Based Semantic Matchmaking. *IEEE Access*, *11*, 27829-27842. https://doi.org/10.1109/ACCESS.2023.3259360

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:11 Oct 2026

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 February 2023, accepted 11 March 2023, date of publication 20 March 2023, date of current version 23 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3259360

RESEARCH ARTICLE

An Analysis of Machine Learning-Based Semantic Matchmaking

ERKAN KARABULUT[1] AND RUTE C. SOFIA[2], (Senior Member, IEEE)
[1]Informatics Institute, University of Amsterdam, 1098 XH Amsterdam, The Netherlands
[2]fortiss GmbH, 80805 Munich, Germany

Corresponding author: Erkan Karabulut (e.karabulut@uva.nl)

This work was supported in part by the Project Horizon 2020 (H2020) European Connected Factory Platform for Agile Manufacturing (EFPF) under Grant 825075, and in part by the Project H2020 European Union (EU)-Internet of Things (IoT) under Grant 956671.

ABSTRACT Interoperability remains to be one of the main challenges in the Internet of Things. The increasing number of IoT data sources from various vendors augments the complexity of integrating different sensors and actuators on the existing platforms, requiring human involvement and becoming error prone. To improve this situation, devices are usually coupled with a semantic description of their attributes. Such semantic descriptions, *Things Descriptions, TD*, are therefore an abstraction of devices, that is helpful to achieve a smoother integration of devices into IoT platforms. However, TD are usually vendor-based, so for large-scale IoT infrastructures, the integration complexity increases, as there will be different descriptions of similar sensors, provided by different vendors to be interconnected into IoT platforms. In this context, the paper assesses different ML-based semantic matchmaking approaches, against a sentence-based statistical similarity approach. For the ML approaches, the paper focuses on clustering and Natural Language Processing. The three approaches have been implemented on a realistic testbed, and experiments carried out show that the best performance achieved in terms of accuracy, time to completion of a matchmaking request, and memory usage is the NLP-based approach.

INDEX TERMS IoT, machine learning, semantics, matchmaking, interoperability.

I. INTRODUCTION

Internet of Things (IoT) infrastructures today integrate a wide variety of IoT devices such assensors, actuators, that are provided by different vendors. In such environments, each vendor usually provides IoT services via a vendor-specific IoT platform. Hence, the process of integration of IoT devices in large-scale IoT infrastructures becomes complex, requiring a significant level of manual intervention and as consequence, being error prone.

The required level of intervention can be reduced by considering standardised semantic models to describe IoT data sources. Therefore, standards such as the *Web of Things (WoT)*[1] architecture or the *One Data Model (OneDM)*[2] *Semantic Definition Format (SDF)* provide the support to describe IoT data sources semantically and are widely adopted to increase interoperability.

Still, there are differences in terms of semantic descriptions applied by vendors which arise for different reasons. For instance, such differences may relate with specific regulation differences across vertical domains (e.g., Health, Manufacturing). They may also result from specific requirements derived from the vendor platform. For instance, temperature sensors provided by different sensors may rely by default on different units, or may identify the attribute ''temperature'' with different wording, e.g., ''Temp'', ''Temperature''.

Adding to this complexity, IoT services can also be described (to improve interoperability) via semantic technologies often derived from standards, e.g., ETSI SAREF.[3]

The associate editor coordinating the review of this manuscript and approving it for publication was Sangsoon Lim.

[1]https://www.w3.org/WoT/
[2]https://onedm.org/
[3]https://saref.etsi.org/

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Therefore, a way to improve the integration of IoT devices into IoT platforms is to consider a semantic matchmaking approach that can, in a semi-automated way, provide a finer-grained matchmaking between IoT semantic descriptions, the so-called *Thing Descriptions*, and semantic IoT service descriptions.

The focus of this work is therefore to address how to build such a semantic matchmaking approach, proposing a semi-automated mechanism to match IoT Things Descriptions into IoT services, based on *Machine Learning (ML)*, and following an ontological approach. The current work is based on the approach that has been briefly described in [1]. The key contributions of this paper are: i) debate on the reasons to apply ML and which approaches may best support semantic matchmaking in the context of IoT services; ii) an analysis of different ML approaches to be applied in the context of semantic matchmaking; iii) a validation based on a realistic testbed of selected ML proposals for semantic matchmaking in IoT environments.

The remainder paper is organized as follows. Related work is presented in section II, where it is also explained how our work contributes beyond state of the art. Section III explains terminology and provides a use-case to assist in the explanation of our work. Section IV debates on the different available semantic matchmaking categories. Section V presents the proposed software architecture and its implementation. Section VI provides an evaluation of selected ML algorithms in terms of matchmaking accuracy and time to completion. Section IX summarizes the paper and explains next steps.

II. RELATED WORK

Several related work addresses interoperability aspects by considering the use of semantic technologies to provide some level of abstraction to IoT devices or services, be it at a communication protocolar level, session, or even application level. In this context, the work developed by the W3C Web of Things working groups towards a semantic description of IoT devices, *Web of Things Thing Description (WoT TD)*[4] is one of the main semantic abstractions for IoT devices. The WoT TD is used to describe an IoT device, its attributes, interfaces, meta-data, security-related properties, and can also be enriched by context-related information. SensorThings API Sensing[5] that is published by *Open Geospatial Consortium (OGC)*, describes a geospatial-enabled and unified way to interconnect IoT devices. The SDF[6] language from One Data Model (OneDM) is another approach to semantic interoperability, a language that provides a way to describe IoT devices independently of the underlying communication technology.

Still, when handling with a large number of sensors and IoT services, it is also important not just to semantically describe cyber-physical systems, but also to have a way to compose enriched semantic descriptions for categories of devices, to speed up the data aggregation process, and to provide finer-grained results. In this context, a composition approach quite used in Web services is the *atomic mashup*. For the specific context of IoT, Kast et al. have proposed the use of semantic descriptions to identify a set of IoT devices that can be used together [2]. The authors have introduced the concept of *System Description (SD)* which represents a *mashup* of IoT devices containing an enriched WoT TD. A mashup in this context is a set of IoT Things that can be used together to offer a specific service. The authors also introduce an atomic aggregation approach, the *atomic Mashup*, which refers to a series of indivisible interactions to perform a certain task, from the perspective of a service.
The atomic mashup approach is interesting as it assist in fine-grained data aggregation, via an atomic model of categorizing similar TDs. However, this approach requires pre-defined templates and a set of user-configurable rules for the design space reduction. Therefore, atomic mashup generation still requires a considerable effort from a human operator. To assist this problem, Korkan et al. have developed an atomic mashup generator that aims to reduce the design space for a mashup generation process [3].

Ontology-based semantic matchmaking is the most common approach supporting IoT semantic matchmaking. For instance, Shu et al. proposed an ontological approach based on OWL which computes the semantic distance of concepts in an ontology [4]. Ontologies support scalability within a specific domain, e.g., manufacturing, health. However, cross domain semantic matchmaking based on ontological approaches creates issues due to the lack of specific cross-domain approaches. Cassar et al. address this problem via a hybrid semantic matchmaking approach, that relies on probabilistic matchmaking combine with latent semantic analysis with a weighted-link analysis based on logical signature matching [5]. While their solution is interesting in terms of cross-domain interoperability and scalability, it falls short in terms of the capability of detecting closeness between different TDs, e.g., when different ontologies are applied. Things to Service Matchmaking (TSMatch) is a fortiss semantic matchmaking concept and open-source middleware[7] which has first relied on statistical proximity (Sørensen–dice coefficient and term frequency–inverse document frequency) to support semi-automated semantic matchmaking [6]. Bnouhanna et al. showed that by applying statistical similarity, it was feasible to improve the overall service response and time to completion of specific requests derived from IoT services. Our work builds on the TSMatch learnings, and relies on ML to improve the overall IoT service answer, in terms of latency (time to completion) and accuracy.

III. USE-CASE

Fig. 1 provides a representation of an IoT semantic matchmaking use-case between IoT Things, and IoT services,

[4]https://www.w3.org/TR/wot-thing-description/
[5]https://docs.ogc.org/is/18-088/18-088.html
[6]https://onedm.org/sdflanguage/
[7]https://git.fortiss.org/iiot_external/tsmatch

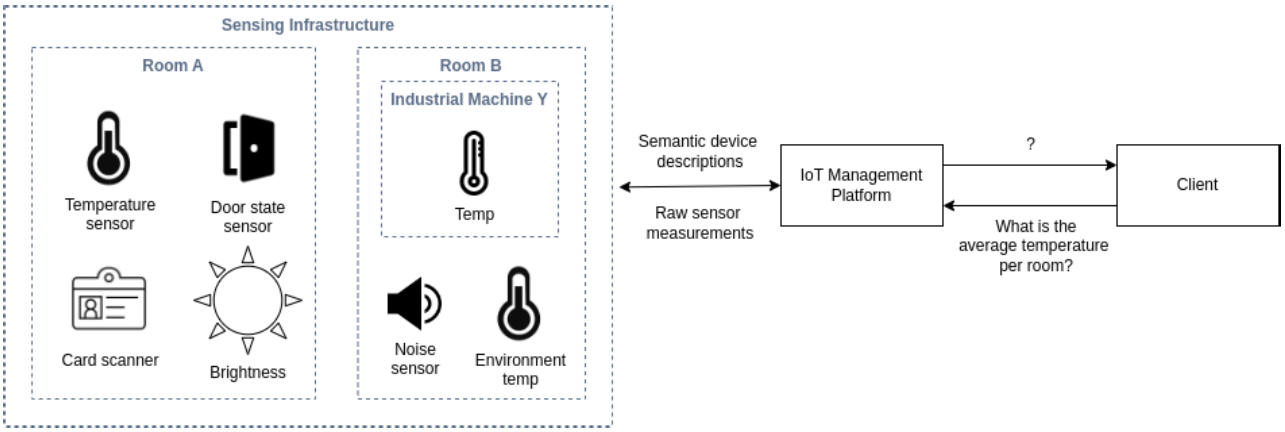

FIGURE 1. Using semantic matchmaking to improve data aggregation results in a Smart Facility use-case.

within the context of a Smart Factory shop-floor. Today, factories hold multiple sensors integrated into the different factory environments, e.g., shop-floor, warehouse. There are machines from different vendors, with coupled sensors attached on them. Moreover, sensors are also used to monitor the environment, e.g., CO2, temperature, humidity. 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. Different services, e.g., data analytics tooling, environmental monitoring services, 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 shop-floor) and others run on the Cloud.

In this scenario, a semantic matchmaking can help identifying sensors that can be used together to provide a requested service. The matchmaking can occur on different aspects such as measurement type or measurement unit. As an example, various machines obtained from different vendors has similar sensors integrated on them, however with differences in their semantic descriptions. The proposed semantic matchmaking approach facilitates interoperability in such scenarios. From the computational perspective, 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 to also reduce energy consumption, as most of the data processing (including aggregation) is performed closer to the end-user.

For this scenario, the following assumptions are considered in the context of the work developed:

- semantic matchmaking occurs on the Edge, to reduce latency.
- Any IoT devices are represented by a standardised TDs.
- Any IoT service is represented by a standardised semantic description.
- A set of ontologies can be used, to improve the matchmaking.
- The IoT device discovery is handled by an existing process.

IV. SELECTED SEMANTIC MATCHMAKING APPROACHES

To understand which matchmaking approaches exist, we have analysed and categorized existing related work on semantic matchmaking, as listed in Table 1. This paper focuses on the matchmaking for semantic descriptions that are written in English language only. Taking multiple languages into account during the matchmaking remains to be an open issue for a future work.

The related work analysed shows that 4 different approaches have been frequently used in the context of IoT semantic matchmaking: i) ontology-based; ii) clustering-based; iii) statistics-based; iv) supervised learning-based.

The first category of work (ontology-based) takes ontology data (or a set of classes) as input and matches or labels a given data point to one of the elements in the ontology using rule-based functions or queries that are developed specifically for the elements of that ontology [7], [8]. In case there are data points which are described based on another ontology, or the characteristics of the data isn't covered within currently used ontology, then this approach will fail to do a matchmaking. Therefore, this approach is strictly dependant on an ontology and also requires manual intervention each time a new term or a notion is introduced in the system. The proposed solution should work with any given ontology and therefore this approach may not be the most suitable.

The second category of work analysed,clustering-based, creates clusters that consist of semantically similar elements [9], [10], [11]. When a new data point has been

detected, it is matched to the semantically closest cluster. This approach requires a training process per ontology or per set of output classes. However, it can be accomplished without any source code changes or manual intervention, and also without labelling data required for training. Assuming that an ontology may not change too frequently, then this approach can be a good candidate for semantic matchmaking in the context of IoT.

The third approach (statistical) matches words that are semantically close to each other. Notion of similarity can be obtained using various statistical approaches such as Jaccard similarity [12] and term frequency-inverse document frequency, or also using a lexical database [13]. This approach has the advantage of not requiring learning, and of also not requiring an ontology.

The last category (supervised learning) has been used in smaller scale sensing use-cases, given that the derived datasets may be of different sizes, and integrate very different features. It requires a training process with all possible output classes which can't be known beforehand in this type of use-cases. As an example, assume that a model is trained using a supervised approach. When a new IoT sensing device that is described using a different information model is introduced to the system, the model will not understand the new information model, requiring adjustment.

Out of the studied work, we have selected three different approaches that may be applicable cross-domain (do not dependend of an ontology) and that do not require labelled data: i) a statistical approach, based on sentence similarity; ii) a *Natural Language Processing (NLP)* neural network-based approach; iii) a clustering-based approach. The first approach is based on [13] and doesn't include any learning process. It uses the WordNet [16] lexical database to create a semantic vector representation of a sentence and also considers the order of the words in a sentence. Since there is no learning process, it can be implemented without being dependant on any information model or ontology. It focuses directly on calculating the similarity between sentences and it can also be used short texts or phrases. Values in a Thing Description (TD) are usually a couple of words long, or in the case of a ''description'' field it can be as long as a couple of sentences. Therefore, this approach is a good candidate for semantic matchmaking of IoT Things to services.

The second approach is based on NLP and uses a neural network model to learn word associations from a large corpus of text [17], [18]. In this context, Mikolov et al. introduced *Continuous Bag-of-Words Model (CBOW)* and *Continuous Skip-gram Model (Skip-gram)*. These approaches are used to create vector representations of words. CBOW predicts a word based on the context used, while Skip-gram predicts a set of words before and after a given word. The first model fits better to the described scenario, since the goal is to predict output classes based on the context, e.g. match a given TD to an output class with the name ''Temperature''.

The third approach relies on clustering, being based on the *k-Means* algorithm. k-Means is a well known clustering method that is also used in semantic matchmaking [11]. Clustering-based algorithms have the drawback that the total number of clusters to be considered need to be defined beforehand, so for a semantic matchmaking process it is necessary to assess how that number can be obtained. For the proposed approach, the number of clusters has been set to the number of *ontological categories* for an ontology element. A category in an ontology refers to a list of possible values for an ontology element. Taking the example of the cross-domain FIESTA-IoT ontology [19], ''Temperature'', ''Humidity'' and ''Air Quality'' are some of possible values for the FIESTA-IoT ''Quantity Kind'' ontology element.

TABLE 1. Semantic matchmaking approaches and categories.

Type	Title	Matching
Ontology-based	A Hybrid Semantic Matchmaker for IoT Services [5]	IoT services to requests
Ontology-based	Semantic Matchmaking for Job Recruitment: An Ontology-Based Hybrid Approach [14]	Job seekers to job postings
Supervised learning-based	A Machine-Learning Approach for Semantic Matching of Building Codes and Building Information Models (BIMs) for Supporting Automated Code Checking [15]	Building Codes to Building Information Models (BIMs)
Cluster analysis for db schemas, supervised learning for instance-level matching	Semantic Matching Across Heterogeneous Data Sources [9]	Data source matching, e.g. matching of 2 relational db
Supervised learning-based	Ontology Matching: A Machine Learning Approach [12]	Different ontologies in a single domain
Clustering	Short Text Clustering Enhanced by Semantic Matching Model [10]	Query to documents on (Short texts is social media) dataset
Clustering	The k-Means Clustering Algorithm With Semantic Similarity To Estimate The Cost of Hospitalization [11]	Clustering of patients using a semantic similarity as distance based on an ontology
Statistics-based	Sentence Similarity Based on Semantic Nets and Corpus Statistics [13]	Semantic meaning of different sentences
Ontology-based	Machine learning in the Internet of Things: A semantic-enhanced approach [7]	Sensor data to ontology elements

V. ARCHITECTURAL DESIGN

The developed concept illustrated in Fig. 2, where the blue boxes represent the software components that have been added in the context of this work, has been integrated into the fortiss open-source software TSMatch v2.0.[8]

TSMatch has been applied in experimental pilots (TRL6) across the H2020 European Connected Factory Platform for Agile Manufacturing (EFPF)[9] project, and a demonstrator is

[8]https://git.fortiss.org/iiot_external/tsmatch
[9]https://www.efpf.org/

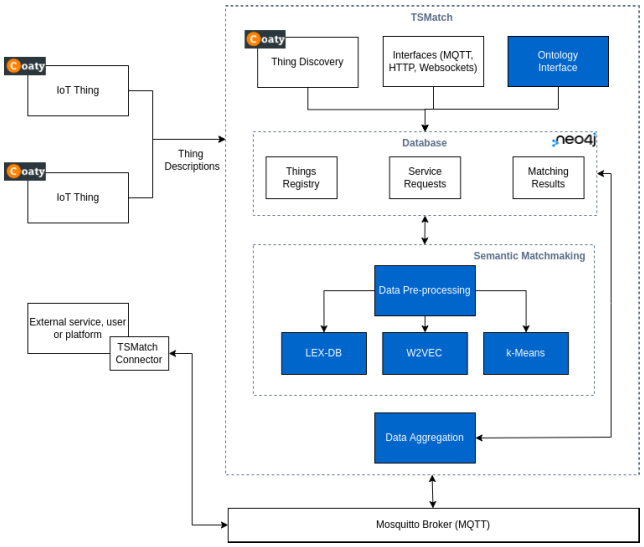

FIGURE 2. The implemented modules on the TSMatch semantic matchmaking moddleware.

available and interconnected to the EFPF data spine via the fortiss IIoT Lab.[10]

The main goal of TSMatch is to automate the data exchange between IoT data sources and services, while satisfying the service needs. For that, release v1 of TSMatch followed a semantic similarity matchmaking approach. Release v2 of TSMatch integrates the results of our work. Following a client-server approach, TSMatch comprises a server-side, the TSMatch engine, and a TSMatch client.

TSMatch relies on the coaty.io[11] framework to discover IoT devices on an infrastructure. The respective TDs are stored on a database (graphDB) implemented via Neo4j, and which stores IoT TDs, service requested, and matchmaking results. IoT Things are interconnected via MQTT (Mosquitto) with coaty.io.

Moreover, multiple connectors have been developed to allow interoperability of TSMatch with different platforms. For instance, the EFPF connector corresponds to a MQ Telemetry Transport (MQTT)-Advanced Message Queuing Protocol (AMQP) connector. A REST connector is used to interconnection with external service platforms.

The design of the proposed semantic matchmaking process considers the following functional blocks (blue):

- Ontology interface.
- Data pre-processing.
- Semantic matchmaking, where the three different approaches selected have been implemented: i) statistical approach based on cosine similarity (LEX-DB); ii) NLP neural-network model approach (W2VEC), iii) clustering-based approach (k-Means).
- Data Aggregation.

The proposed mechanism runs in 2 different phases: a setup phase (section V-A) and a run-time phase (section V-C).

[10]https://www.fortiss.org/en/research/fortiss-labs/detail/iiot-lab
[11]https://coaty.io/

During setup, an ontology is imported into the TSMatch via the ontology interface module. During run-time, both the TD dataset(s) and the ontology dataset(s) are pre-processed, and then passed to the semantic matchmaking functional block, to be handled by one of the three proposed algorithms. The algorithm then matches TDs to the ontology centroids (aggregation points). Service requests are captured by the data aggregation module and matched to the aggregated TD, so an aggregated (averaged) value is provided to the service.

A. SETUP PHASE-ONTOLOGY IMPORT

Fig. 3 illustrates the communication sequence for the ontology importing process. First, an external data provider sends the ontology data to the ontology interface module via a REST interface. The ontology interface then replaces the current ontology data with the new ontology in the graph database. Then it notifies the TD to Ontology Matching module about the ontology change. The matching module gets the existing TDs from the database, matches them to the new ontology elements, and stores the new matching.

TSMatch is expected to perform the semantic matchmaking based on any given ontology. In order to import ontology data into the TSMatch, a module named ontology interface has been developed. The ontology interface provides a REST API that allows importing an ontology into the graph database. Currently, it accepts both JSON and OWL files. When a new ontology import is triggered, the following steps are executed in order:

- Delete matching of sensors to ontology elements in the graph database.
- Delete currently used ontology nodes.
- Convert the given ontology to JSON if necessary.
- Create new nodes and edges that corresponds to the classes and relations in the given ontology.
- Find and mark aggregation points on the ontology if not already given via the service request.
- Trigger TD to ontology matching service to match available sensors to the new ontology elements.

In the proposed algorithm, an aggregation point is detected automatically based on its *centrality*, i.e., it corresponds to a node with a high node degree. Child nodes of an aggregation node, e.g. sub-classes in an ontology, represent possible values for that aggregation point. To give an example, in the cross-domain FIESTA-IoT ontology, there are 3 nodes that have a higher centrality. These are the ''QuantityKind'', ''Unit'' and ''SensingDevice'' ontology elements. A given TD will be matched to one of the child nodes for each of the aggregation points. An example matching would be; ''QuantityKind'': ''Temperature'', ''Unit'': ''DegreeCelsius'' and ''SensingDevice'': ''Thermometer''.

In case these aggregation points aren't given in the request to the ontology interface, then they can be found by running an anomaly detection method on the number of neighbors that each node has. The proposed implementation includes a simple anomaly detection method that is based on the statistical empirical rule: according to the statistical empirical

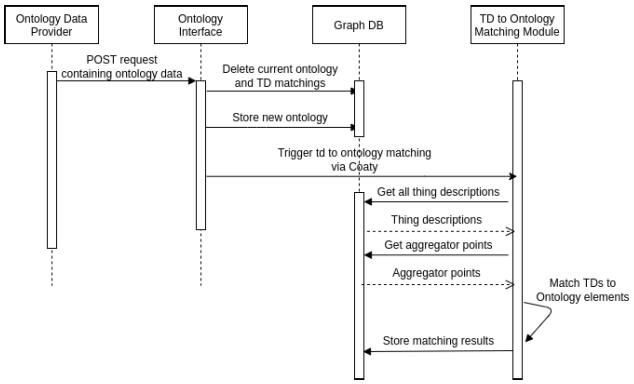

FIGURE 3. A sequence diagram showing the exchanged messages during an ontology import process.

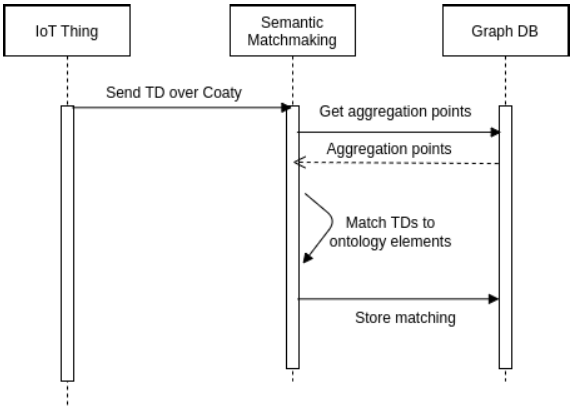

FIGURE 4. Communication sequence for a Things Discovery process.

rule, 99.7 percent of the values are within a range that corresponds to 3 times the standard deviation.

B. SETUP PHASE-THINGS DISCOVERY

Figure 4 illustrates the communication sequence for the Thing Discovery process.

When an IoT device boots up or becomes active after a period of inactivity, it publishes its TD via coaty.io. These TD are stored on the local database of TSMatch. 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 as described in the next section, and stores the new data nodes on the local database (graphDB).

C. RUN-TIME-DATA CLEANUP

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 TD to the ontology elements. Then a relation is created in the graph database between the TD and the ontology nodes that it is matched to.

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

FIGURE 5. Part of a TD from [20].

D. RUN-TIME: SEMANTIC MATCHMAKING

This section defines how TD to ontology element matchmaking is performed using the proposed three semantic matchmaking algorithms.

Some attributes in a TD appear in different information models. Common terms in WoT or OGC and SDF are ''name'', ''description'' or ''title''. These fields include valuable information regarding the type of the sensor, its measurement unit for instance. As an example, the sample TD shared in Fig. 5 has name and description fields where it mentions the ontology ''quantity kind'' (Light intensity) and measurement unit (Lux) of a sensor. Therefore, the algorithm initially checks if a matchmaking can be done using these fields.

Each collected TD has one or more sensor description. Secondly, the algorithm checks if a matching can be found using the different sensor descriptions. If this is also unsuccessful, then it checks the remaining parts of the TD to find a matching to the aggregation points that have been earlier extracted from the ontology.

1) STATISTICS BASED SENTENCE SIMILARITY: LEX-DB

LEX-DB is based the work by Li et al [13]. It produces a similarity score between 0 and 1 using semantic and syntactic information contained in the given pair of texts. Initially, the algorithm creates raw semantic vectors and word order vectors for the sentences with the assistance of the lexical database WordNet. Contribution from each word to the meaning of a sentence is marked by assigning it a weight value using a text corpus and then the weights are combined with the raw semantic vector. Finally, an overall semantic similarity score is calculated using a weighted word order similarity and semantic similarity. Li et al. argue that syntax has more effect on the semantic processing of a text and hence the authors use higher weight for syntactic similarity than the word order similarity

2) NLP-BASED SEMANTIC SIMILARITY APPROACH: W2VEC

Word embeddings refers to vector representations of words in terms of real numbers. Our W2VEC approach is based on the work by Mikolov et al. where the authors introduce CBOW (rf. to section II) [17], [18]. As has been explained in Section II, CBOW trains word vectors from a TD dataset.

W2VEC relies on cosine similarity while calculating similarity of an ontological element to a value field in a TD. In order to obtain a single vector for a text field that consists of multiple words, an average vector is calculated. However, while comparing two texts with different sizes, e.g. 2 words and 10 words, average vector representation might not reflect the actual similarity even the 2 words inside the first text appear in the second text. As an example, consider the following 2 short texts: ''temperature sensor'' and ''a highly sensitive temperature sensor with model number X from company Y''. Even though it can be inferred that that these two texts describe a ''temperature sensor'', the extra words in the second sentence (''high'', ''sensitive'', ''model'', ''number'', ''X'', ''company'', ''Y'') changes the average representation of the sentence and hence when put on a vector space, it gets further away from the vector that represents ''temperature sensor'' only. To overcome this, the n-gram method with a dynamic n value has been applied in W2VEC. In the n-gram method, the words that appear together in a long text are put in the same set and such multiple sets of words created for a single text.

The following steps have been used to compute semantic similarity between an aggregation point name or category node name and a key or value in a TD:

- Assign the word count in a category node name or aggregation point name. (output class name in short) to N
- If N is smaller than 2, then assign 2 to N.
- Calculate average vector representation for the output class name.
- Separate a given key or value in a TD into multiple sets based on the value of N.
- Calculate average vector representations for each set.
- Calculate the cosine similarity between the vector that represents the output class and vectors that represent each set.
- Find the set with the highest cosine similarity.
- If the cosine similarity is higher than a threshold, then accept it as a match.

3) CLUSTERING-BASED SEMANTIC SIMILARITY APPROACH: K-MEANS

The last semantic similarity approach creates cluster representations for each ontology elements based on an adaptation of k-Means. For a given TD dataset, the algorithm checks if it matches to an ontology element and extracts the matching phrase in case of a match. Each matching phrase is then put into a cluster that is dedicated to a single ontology element. As an example, there are 11 category elements for domain of interest category in the FIESTA-IoT ontology and therefore 11 clusters are created. A TD is assigned to only one of the 11 clusters.

The k-Means algorithm requires a notion of distance between clusters and data points. In the original version of the algorithm, different number of clusters are evaluated to find the right amount of clusters that can best represent the variety among the data points. For our adapted version, we consider as centroids for each cluster the nodes that have the highest betweenness centrality value. Then the algorithm assigns data points to the nearest cluster. After this initialization phase, centroids for each cluster are calculated again, this time by taking the average values of data points in that cluster. Then data points are reassigned to the closest clusters. This final step is repeated for a predefined amount of time or until no data point is assigned to a new cluster. Hence, in the k-Means version used, the number of clusters is equal to the total number of aggregation points obtained from the used ontology/ies. For instance, again considering the FIESTA-IoT ontology, there are 178 different classes defined for ''QuantityKind'', which correspond to the ontology aggregators (cluster centroids). Hence, in total and for FIESTA-IoT the algorithm would consider 178 clusters.

```
{
    "QuantityKind": "Illuminance",
    "Unit": "Lux",
    "SensingDevice": "LightSensor",
    "DomainOfInterest": "Environment",
    ...
}
```

FIGURE 6. A service request enriched with elements from the FIESTA-IoT ontology.

E. RUN-TIME: DATA AGGREGATION

Data aggregation is performed based on the assumption that every IoT service can be described semantically according to an ontology. Upon receiving a service request, data aggregation module looks for matching sensing devices according to the ontological elements inside the request. Then the engine subscribes to data from those sensors and aggregate using a simple average function. Based on the service example provided in Fig.6, the data aggregation module searches for sensor descriptions in the graphdb that have a relation to ''Illuminance'', ''Lux'', ''LightSensor'', and ''Environment'' nodes. It sends a response back to the requester that contains a list of sensor and TDs. Lastly, the module subscribes to data from those IoT devices, aggregates data using an average function (to be improved) and continuously sends the aggregated data to the requester until the service request is deleted.

VI. EXPERIMENTAL ENVIRONMENT AND DATASETS

The three discussed ML-based approaches have been implemented and analysed on the TSMatch testbed of the fortiss IIoT Lab. The performance evaluation parameters that have been considered are:

- **Accuracy**, defined as the semantic matchmaking accuracy of Things to service matching. Percentage of correct matchings to all matchings.
- **Time to completion** of a matchmaking process, corresponding to the instant of time measured on the device where TSMatch resides, for matching TD to ontology elements.

- **Peak memory usage** during TD to ontology matching for all of the proposed approaches.

In terms of methodology, we have first created training and testing datasets for each of the proposed approaches. The data used as baseline for comparison was manually extracted from the testing dataset. Each of the semantic matchmaking algorithms results were then compared to the baseline in terms of accuracy. The experiments have been repeated several times.

A. DATASETS

1) TD DATASET

The matchmaking requires datasets comprising heterogeneous TDs. For the purpose of analysis, a large TD dataset has been created[12] having as basis two existing datasets. The first is the WoT Dataset composed of heterogeneous TDs,[13] corresponding to sensor TD. This dataset comprises 327 files, each of which holds one or more than one sensor or actuator. There are 2191 sensor TDs in total [20]. The second is the OneDM SDF TD dataset,[14] composed of 200 sensor TDs [21].

Therefore, in total a sensor TD dataset (DAT1) comprising 2391 heterogeneous sensor TDs has been created.

2) ONTOLOGY DATASET

After experimenting with different ontologies, we have selected the FIESTA-IoT ontology [19], [22]. FIESTA-IoT has the particularity of integrating cross-domain elements and therefore it has a good coverage relevant for scenarios across different domains.

FIESTA-IoT comprises 483 different class of entities including 178 category elements for ''QuantityKind'', 92 category elements for ''Unit'', 109 category elements for ''SensingDevice'' and 11 category elements for ''DomainOfInterest'' class.

3) TRAINING AND TESTING DATASETS

Out of DAT1, 2 testing and 1 training datasets have been built.

A first testing dataset, TESTING1,[15] consists of 200 sensor descriptions, roughly 10%, out of DAT1, that have been randomly selected based on a uniform distribution using the package from NumPy.[16] The testing dataset has been manually labeled based on the FIESTA-IoT ontology. Therefore, due to time constraints, instead of trying different testing-training split percentages, only 1 split (10%) option is considered. The remaining %90 percent of DAT1 dataset is used as the training dataset, TRAINING,[17] for Word2Vec and K-Means algorithms.

A second testing dataset (C-TESTING[18]) has been considered, still relying on 200 sensor descriptions, but removing the sections of the TD that are not applicable to the matching towards an ontology. For instance, the WoT TD attribute ''created'' or ''modified'' defines when the TD is created and modified. Therefore, it doesn't contribute to the semantic meaning while understanding which ontology element does this description relates to. In C-TESTING, only the sections of the WoT TD that are applicable to the semantic matchmaking process are considered. The aim of this specific dataset is to evaluate potential improvements in terms of node usage, e.g., CPU, memory, and eventually, running time.

4) BASELINE TESTING DATASET

The performance evaluation that has been carried out considers the testing datasets (TESTING1, C-TESTING) as baseline[19] which has been manually labelled. This dataset has been built based on human assessment of the categorisation of each sensor on the testing datasets against the aggregation points of the FIESTA ontology. Baseline dataset contains sparsity for ''domain of interest'' category due to very few TDs having domain related information. Therefore the ''domain of interest'' category is excluded from all of the accuracy measurements that are described in the next sections.

To exemplify how the labelling was done, consider a TD that has ''humidity'' as its title, a description section which mentions that this sensor is measuring relative humidity in a room and another field named ''unit'' and its value is ''%''. By looking at the first 2 fields, and to the QuantityKind category in FIESTA-IoT Ontology, one can infer that the TD can be linked to the FIESTA aggregation point ''RelativeHumidity''. Among the sensing device category elements, this description refers to a ''HumiditySensor''. By looking at the unit field, one can also infer that this sensor provides data in percentages and hence it corresponds to ''Percent'' category element for the unit category.

5) WORD VECTOR TRAINING DATASET

This dataset is used to derive worst-case and best-case accuracy thresholds, that can be used to apply on the semantic matchmaking process. A worst-case and best-case assessment assists in defining limits that are relevant in particular when considering that the matchmaking process occurs on the Edge, eventually being applicable to far Edge devices (constrained devices).

The word vectors published by Google includes 3 million vectors with 300 dimensions each that takes around 6 GBs of space in a text file. Therefore a subset of the word vectors will be used. In order to analyse different word vector subsets, an experiment that shows the effect of subset size on the accuracy is carried out. Table 2 shows how different subset

[12] https://git.fortiss.org/iiot_external/tsmatch/-/tree/master/dataset
[13] https://github.com/w3c/wot-testing/tree/main/events/2022.03.Online/TD
[14] https://github.com/one-data-model/playground/tree/master/sdfObject
[15] https://git.fortiss.org/iiot_external/tsmatch/-/tree/master/dataset/testing
[16] https://numpy.org/
[17] https://git.fortiss.org/iiot_external/tsmatch/-/tree/master/dataset/training
[18] https://git.fortiss.org/iiot_external/tsmatch/-/tree/master/dataset/testing_cleaned
[19] https://git.fortiss.org/iiot_external/tsmatch/-/blob/master/dataset/testing/ground_truth.txt

TABLE 2. Size and accuracy comparison for word vector subsets of different sizes.

Subset size	Size in MBs	Accuracy
1,000	3.48	49.16
2,000	7.96	50.33
5,000	17.4	68.83
10,000	34.8	62.66
20,000	69.6	64.83
50,000	174	70.16
100,000	348	73
200,000	796	73.83
300,000	1194	74.66

of word vectors perform. The word vectors are ordered based on the frequency of the words in descending order. Therefore each subset with size x refers to the most popular x words. Subset size in MBs refers to how much space does a subset requires when stored in a text file. Accuracy is calculated using the best performing key and value thresholds, based on the results obtained after evaluating different key and value thresholds (see section VII). While the subset size is linearly correlated with the size of each file, the increase in accuracy gets lower as the subset size increases.

For the accuracy comparison, the Word2Vec-based and k-Means-based semantic matchmaking algorithms are trained using both the best performing and the worst performing, smallest and the biggest set of vectors.

B. EXPERIMENTAL ENVIRONMENT

The experiments have been carried out via the fortiss IIoT Lab TSMatch demonstrator,[20] having as basis the reference scenario provided in Section III. The testbed components that have utilized during this study are: i) Raspberry Pi 3B+ and Raspberry Pi 4B with 5 sensors attached to each, measuring temperature, humidity, CO2 concentration, particle size in the air and noise, ii) An Intel NUC NUC10i7FNH[21] device that hosts the dockerized TSMatch components, namely, the TSMatch Engine, the MQTT broker and the graph database (Neo4J). A full explanation on how to run TSMatch can be found in the Git open documentation, on the TSMatch technical report.[22] The TSMatch components have been containerized with Docker, and several scripts have been developed to run the TSMatch stack.

VII. PERFORMANCE EVALUATION

This section includes a performance comparison of the selected similarity approaches. Table 3 shows the settings for each of the approaches. For the ones that requires a training process, all besides LEX-DB, the training is performed on a Lenovo ThinkPad T460p laptop (see table 5). However the training process is not subjected to an evaluation since the main goal of the proposed approach is to perform a semantic matchmaking on the edge.

[20]https://www.fortiss.org/en/research/fortiss-labs/detail/iiot-lab

[21]https://www.intel.com/content/www/us/en/products/sku/188811/intel-nuc-10-performance-kit-nuc10i7fnh/specifications.html

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

TABLE 3. Summary of settings for the three selected approaches.

Algorithm	Dataset	Abbreviation
LEX-DB	WordNet [16] lexical db	LEX-DB
W2VEC	TRAINING	W2VEC-TD
W2VEC	300,000 vectors for most popular words published by Google	W2VEC-300k
W2VEC	1,000 vectors for most popular words published by Google	W2VEC-1k
K-MEANS	TRAINING	K-MEANS-TD
K-MEANS	300,000 vectors for most popular words published by Google	K-MEANS-300k
K-MEANS	1,000 vectors for most popular words published by Google	K-MEANS-1k

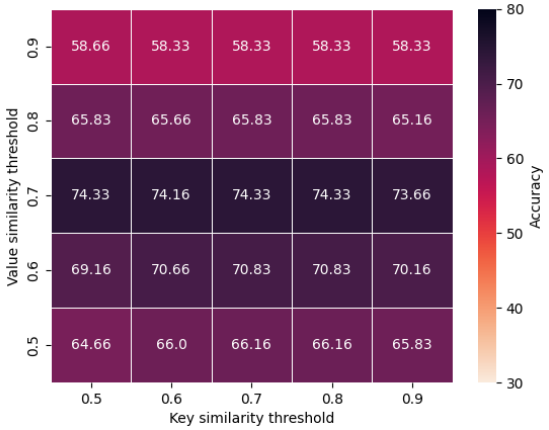

FIGURE 7. Impact of key and value thresholds on the accuracy of W2VEC-300k.

A. SIMILARITY THRESHOLD IMPACT ON W2VEC

There are two threshold values that need to be set manually before running the semantic matchmaking algorithm. These thresholds are the *key* threshold which refers to the value where a key in a TD is accepted as similar to an aggregation point, and the *value* threshold, where a value in a TD is accepted as similar to child nodes of an aggregation point. In order to find which key and value thresholds provide better accuracy, different combinations of the two thresholds have been evaluated, for all of the proposed approaches. The sentence similarity approach performed worse than the other 2 approaches in terms of accuracy with using different key-value thresholds. The results for 2 best performing similarity approaches, word2vec word embeddings and k-Means, are shown in this and the following section.

1) THRESHOLD ANALYSIS FOR A 300,000 VECTORS DATASET

The experiment has been carried out by considering the worst-case and best-case word vectors from table 2.

Figure 7 provides the results when considering 300,000 vectors.

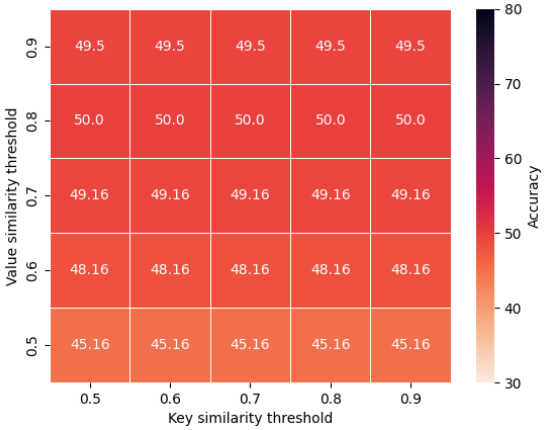

FIGURE 8. Impact of key and value thresholds on the accuracy of W2VEC-1k.

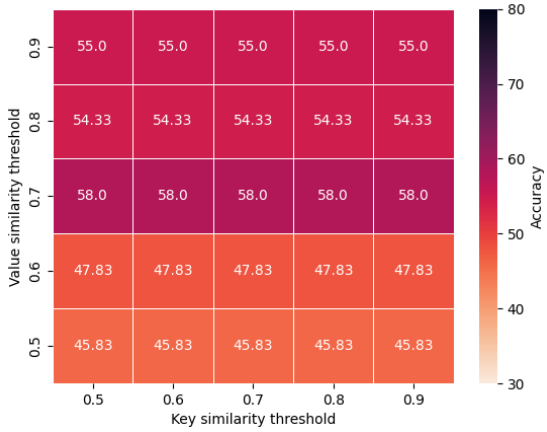

FIGURE 9. Impact of key and value thresholds on the accuracy of K-MEANS-300k.

Both key and value similarity is ranging between 0.5 and 0.9. Accuracy is at the highest level when the key similarity threshold is 0.5, 0.7 and 0.8, and the value similarity threshold is 0.7. When looking at the row where value similarity is 0.7, changing the key similarity impacts the accuracy in less than 1 percent. A roughly similar situation holds for all of the value similarity thresholds. On the other hand, if the key similarity is kept fixed and the value similarity is changed, then the accuracy varies more than 10 percent. As an example, when the key similarity threshold is 0.5 and value similarity changes between 0.5 and 0.9 then the lowest accuracy value becomes 58.66 while the highest accuracy is 74.33 percent. This leads to the conclusion that value similarity threshold has more relevancy than the key threshold on accuracy. This is expected, given that the value threshold relates with the matching to a child node on an ontology, so provides a finer-grained matchmaking.

Since the accuracy is at the highest when the value similarity threshold is 0.7 and the key similarity has a lot less relevancy than the value similarity threshold, another experiment is carried out by ranging the value similarity threshold between 0.65 and 0.75 and keeping the key similarity threshold fixed at 0.5, 0.7 and 0.8. The purpose of this experiment is to see how much the accuracy changes when a more finer-grained increase on the value threshold is applied. This experiment showed that the accuracy is higher when the value threshold is set to 0.67, 0.68 and 0.69. Setting the key similarity threshold to 0.5, 0.7 or 0.8 produces the exact same results. Increasing the value similarity threshold above 0.69 reduces the accuracy of the algorithm. A list of best-performing key value pairs for all of the approaches described in this and the following sections are given in the table 4. Also a more detailed threshold analysis can be found in [23].

2) THRESHOLD IMPACT ANALYSIS FOR A 1,000 VECTORS DATASET

A larger vector dataset intuitively provides the best performance. However, it also brings the trade-off of having to handle large storage and large processing times. Therefore, a second experiment to calibrate the key and value thresholds is carried out, by considering a small dataset, using only the 1,000 vectors for most popular words from Google's dataset.

Figure 8 provides the results achieved when the key and similarity thresholds varies between 0.5 and 0.9. The highest accuracy obtained from this experiment is 50. This experiment shows that changing the key similarity threshold doesn't impact the accuracy result when 1,000 vectors are used. Accuracy values change only when the value similarity change. We hypothesize that for sparse datasets, matchmaking to aggregation points may not be enough to perform well.

Next, another experiment is performed to see how finer-grained changes in value similarity threshold effects the accuracy. Since the key similarity threshold didn't have significant impact in the accuracy in the previous experiment, a key threshold is trivially picked for this experiment. The results show that the highest accuracy is again 50 when the value threshold is 0.79, 0.8, 0.81, 0.82, 0.83 or 0.84. Increasing the threshold further reduces the accuracy.

B. SIMILARITY THRESHOLD IMPACT ON K-MEANS

1) THRESHOLD IMPACT ANALYSIS FOR THE 300,000 VECTORS DATASET

The same experiments are repeated for the K-Means approach when different key and value thresholds ranging between 0.5 and 0.9 are applied. Results are presented in Figure 9. In this experiment, changing the key threshold doesn't impact the accuracy. The best performing value threshold is 0.7 with an accuracy of 58.

A second experiment is conducted, in which the value similarity ranges between 0.65 and 0.75 since the best performing value threshold in the previous experiment was 0.7. In this case, 0.69 value similarity threshold lead to the highest accuracy of 60.66.

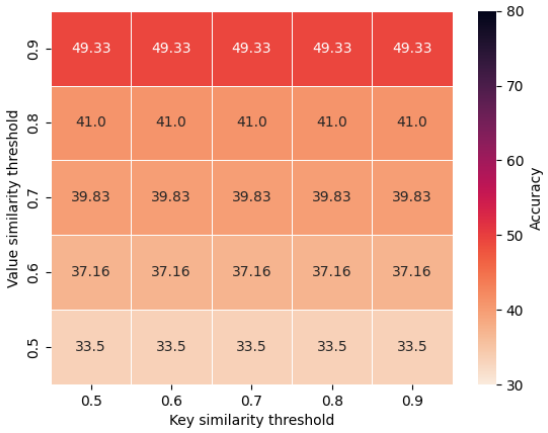

FIGURE 10. Impact of key and value thresholds on the accuracy of K-MEANS-1k.

TABLE 4. Summary, best performing key and value thresholds.

Approach	Key	Value
W2VEC-300k	0.5, 0.7 and 0.8	0.67, 0.68 or 0.69
W2VEC-1k	No impact	0.79, 0.8, 0.81, 0.82, 0.83 or 0.84
K-MEANS-300K	No impact	0.69
K-MEANS-1K	No impact	0.91, 0.92, 0.93 or 0.94

2) THRESHOLD IMPACT ANALYSIS FOR THE 1,000 VECTORS DATASET

We have run a similar experiment for 1,000 vectors using different key and value thresholds and results are shown in Figure 10. Again, changing the key similarity threshold for this experiment didn't impact the algorithm accuracy. The best performing value similarity threshold is 0.9 with the highest accuracy of 49.33 percent.

Another experiment with the finer-grained value thresholds is conducted. In this time, value threshold ranges between 0.85 and 0.95 since the previous experiment showed 0.9 as the best performing threshold. The accuracy value is at the highest 49.6 level when the value threshold is 0.91, 0.92, 0.93 or 0.94.

C. THRESHOLD IMPACT ANALYSIS SUMMARY

A summary for the threshold analysis is provided in Table 4. In regards to W2VEC-300K, the threshold analysis shows that the best performing key threshold values are 0.5, 0.7 and 0.8 for key and one of 0.67, 0.68 or 0.69 for value. When only 1,000 word vectors (W2VEC-1K) are used, the key similarity threshold does not impact significantly the accuracy and the best performing value thresholds are: 0.79, 0.8, 0.81, 0.82, 0.83 or 0.84. In regards to K-MEANS-300K, the key similarity threshold changes did not impact significantly the algorithm accuracy, and the best performing value threshold is 0.69. When 1,000 word vectors are considered (K-MEANS-1K) the key similarity threshold again didn't have a significant impact on the accuracy and the best performing value threshold was one of 0.91, 0.92, 0.93 or 0.94.

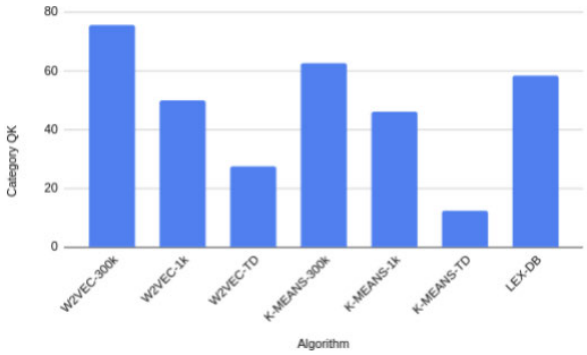

FIGURE 11. Accuracy for category QK.

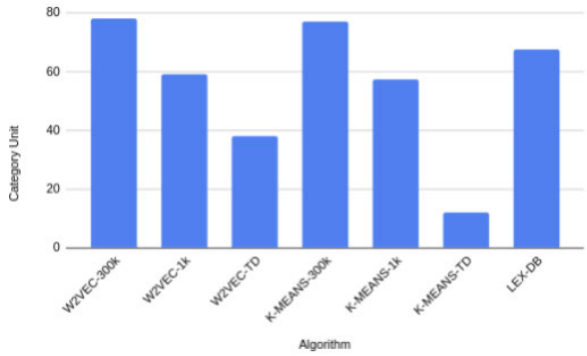

FIGURE 12. Accuracy for category Unit.

D. ALGORITHM ACCURACY ANALYSIS

A second set of experiments has been carried out to assess the accuracy of the proposed approaches, for the best and worst-case threshold scenarios summarised in Table 4.

The results are compared both in terms of the total matchmaking accuracy and also accuracy per category used in the FIESTA-IoT ontology. There are 4 categorizations published for quantity kind (QK), unit, sensing device (SD) and domain of interest ontology elements. As mentioned in section VI-A4, the baseline testing dataset has a high sparsity when it comes to the domain of interest category. Hence it has been excluded in the accuracy analysis.

A match occurs when the selected semantic matchmaking algorithm matches a given TD to the same ontology element as the one that is in the baseline dataset. A mismatch occurs when the algorithms matches the TD to a different ontology element. Lastly, if the algorithm can't find a match for a given TD even though there is a match described in the baseline dataset, then we call it an undetected occurrence. A detailed results table showing the number of matched, miss-matched and undetected instances can be found in [23].

E. SUMMARY OF ACCURACY RESULTS

Fig. 11 presents the accuracy values achieved by each algorithm when performing a matchmaking of IoT TDs to

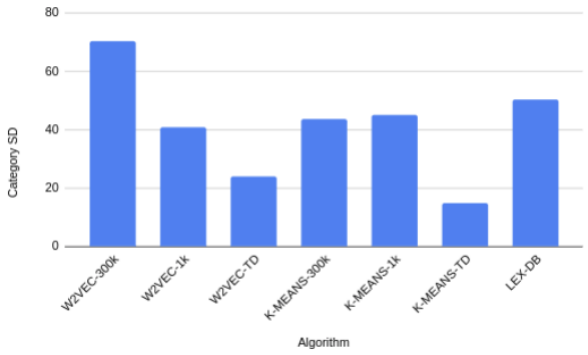

FIGURE 13. Accuracy for category SD.

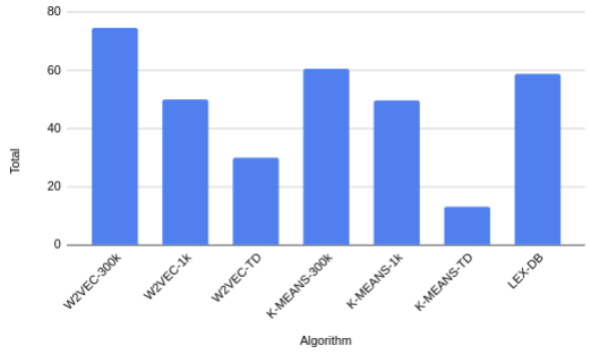

FIGURE 14. Total accuracy.

the FIESTA-IoT category QK. Results show that W2VEC achieves the best accuracy overall. However, the accuracy significantly decreases (as expected) when sparser datasets are used. K-Means is the second best performing algorithm overall, while LEX-DB also shows good performance (however, LEX-DB has been trained with Word2NET). Figure 14 provides accuracy results for all algorithms, when TDs are matched to the FIESTA-IoT category Unit. In this case, W2VEC and K-MEANS achieve a similar performance, with one exception, when the smaller and sparser dataset is considered. Overall, both algorithms achieve a good accuracy percentage both for 300k vectors and for 1k vectors. LEX-DB is presented here, for reference. As explained, LEX-DB has been trained against WordNet and therefore, the accuracy should not change.

Figure 13 provides accuracy results for all algorithms, when TDs are matched to the FIESTA-IoT category SD. W2VEC is the algorithm that provides the best accuracy for this case. K-MEANS exhibits a different result pattern, where the accuracy is slightly higher for the sparser testing dataset (45 instead of 43). The LEX-DB value is now lower for the SD category, which implies that the type of wording may not adequately cover the SD category.

Figure 14 shows the total averaged accuracy per algorithm and case run. W2VEC has a particular good performance when 300k vectors are used, but it significantly reduces if 1k vectors are used. On the other hand, K-MEANS achieves less accuracy than W2VEC when 300k vectors are considered, but seems more stable when 1k vectors are applied.

Overall, W2VEC seems to provide better results when large, dense datasets are considered. For the 1K case, W2VEC accuracy lowers and is similar to the one of K-Means. However, for the majority of cases tested, W2VEC achieves the overall best performance.

LEX-DB accuracy, which has been trained against WordNet, can only be compared with the cases run for 300k vectors. In such case, LEX-DB achieves lower performance than either W2VEC or K-MEANS across all use-cases.

For the case of the smaller and sparser testing dataset (TD cases), both W2VEC and K-MEANS is very low.

Overall, the accuracy achieved for all algorithms and ran cases is between 60% and 80%. Ideally, such accuracy should reach a 90% level, in order for an adequate automated process to run better. We believe this can be improved by relying on multiple ontologies, e.g., a cross-domain approach.

VIII. NODE USAGE ANALYSIS

A last batch of experiments has been run to assess node usage of each algorithm, having in mind future deployments in Edge devices. For far Edge devices, we have selected three different types of equipment representing different types of Edge devices, and Table 5 shows CPU, memory, disk and operating systems for each selected device:

- A Lenovo ThingPad T460p, standing for a regular end-user equipment device.
- A Raspberry Pi 4B, standing for an embedded Edge controller device.
- An Intel NUC 10, standing for an example of a IoT gateway device.

The node usage analysis considers average time required to perform a match (Matching Duration in milliseconds, MD), and peak memory in MBytes (PM, MB) use for each selected algorithm. To obtain the running time, the semantic matchmaking algorithm is executed with a given similarity algorithm for 200 times and average time required to match a TD to an ontology element is calculated. Memory usage of the process that runs the matchmaking algorithm is also tracked during the whole execution and highest memory usage is observed. Table 6 presents results for TESTING1.

In terms of MD, the best algorithm is W2VEC across all devices. LEX-DB is by far the algorithm performing worse across all devices. In regards to memory, the algorithm that shows better performance is LEX-DB overall, while W2VEC and K-MEANS show a similar performance. We have repeated the experiment for the C-TESTING set, to understand if a deeper cleaning process of the dataset may in the future significantly impact node usage results. Results are provided in Table 7. Results show that a thorough cleaning process can improve results in terms of MD, but we highlight that there is not a significant impact. For instance, W2VEC-300k has a performance improvement from 774ms to 708ms (0.08%) for the Lenovo equipment; a reduction from 2783 to 2184 ms (0.2%) for the Raspberry Pi 4B.

TABLE 5. Hardware details and operating system of each testing device.

Device	CPU	Memory	Disk (Total / Free	OS
Lenovo ThingPad T460p	Intel® Core™ i7-6700HQ CPU @ 2.60GHz × 8	Samsung M471A1K43-BB0-CPB 16 GiB	Samsung SSD 850 (465 GiB / 245 GiB)	Ubuntu 20.04.4 LTS
Raspberry Pi 4B	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	4GB RAM	32 GB / 16.8 GB SD Card	Raspbian GNU/Linux 10 (buster)
Intel® NUC 10 Performance kit NUC10i7FNH	Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz	Kingston SODIMM DDR4 Synchronous 2667 MHz 16 GiB	Samsung SSD 970 EVO Plus (500 GBs / 404 GBs)	Ubuntu 20.04.3 LTS

TABLE 6. Node usage analysis results for TESTING1.

Device Type	Device	MD (ms)	PM (MB)	Algorithm
End-user device	Lenovo ThinkPad T460p	156189	258	LEX-DB
		774	3894	W2VEC-300k
		2025	3910	K-MEANS-300k
		179	249	W2VEC-1k
		609	249	K-MEANS-1k
Edge controller	Raspberry Pi 4B	above 10 minutes	142	LEX-DB
		2783	2401	W2VEC-300k
		3899	2402	K-MEANS-300k
		715	121	W2VEC-1k
		2531	122	K-MEANS-1k
IoT Gateway	Intel® NUC 10 Performance i7FNH	60979	256	LEX-DB
		585	3889	W2VEC-300k
		1440	3890	K-MEANS-300k
		145	240	W2VEC-1k
		556	240	K-MEANS-1k

TABLE 7. Node usage analysis for C-TESTING.

Device Type	Device	MD (ms)	PM (MB)	Algorithm
End-user device	Lenovo ThinkPad T460p	148911	257	LEX-DB
		708	3892	W2VEC-300k
		1266	3908	K-MEANS-300k
		160	249	W2VEC-1k
		404	249	K-MEANS-1k
Edge controller	Raspberry Pi 4B	above 10 minutes	141	LEX-DB
		2184	2400	W2VEC-300k
		3671	2401	K-MEANS-300k
		626	121	W2VEC-1k
		1665	121	K-MEANS-1k
IoT Gateway	Intel® NUC 10 Performance i7FNH	56031	255	LEX-DB
		515	3888	W2VEC-300k
		910	3891	K-MEANS-300k
		124	240	W2VEC-1k
		376	240	K-MEANS-1k

The improvement range is similar when considering the 1k cases (0.1%). In terms of memory usage, there is also not a significant improvement.

Adding a deeper cleaning process will also impact the overall memory and matching time. The improvements observed hint that such integration may not pay up in terms of node usage improvement.

IX. SUMMARY

This work addressed the development of semi-automated matchmaking processes that can assist IoT interoperability. The proposal considers an existing middleware, TSMatch, and addresses the reasons to integrate ML in order to develop a finer-grained matchmaking. Three approaches have been selected (LEX-DB, W2VEC, K-MEANS) and evaluated in a realistic testbed, fortiss IIoT Lab, in terms of accuracy and node usage impact. Results achieved show that the best performing solution is W2VEC (NLP based on a neural network model), being LEX-DB the worse performing solution. The results achieved allowed also to detect gaps that can be addressed in future work.

ACKNOWLEDGMENT

The authors would like to thank Prof. Jörg Ott for his advisorship in the M.Sc. work that lead to this publication. The work of Erkan Karabulut was done while at fortiss GmbH.

REFERENCES

[1] E. Karabulut, N. Bnouhanna, and R. C. Sofia, "ML-based data classification and data aggregation on the edge," in *Proc. CoNEXT Student Workshop*, Dec. 2021, pp. 21–22.
[2] A. Kast, E. Korkan, S. Kabisch, and S. Steinhorst, "Web of things system description for representation of mashups," in *Proc. Int. Conf. Omni-Layer Intell. Syst. (COINS)*, Aug. 2020, pp. 1–8.
[3] E. Korkan, F. Salama, S. Kaebisch, and S. Steinhorst, "A-MaGe: Atomic mashup generator for the web of things," in *Proc. Int. Conf. Web Eng.* Cham, Switzerland: Springer, 2021, pp. 320–327.
[4] G. Shu, O. F. Rana, N. J. Avis, and C. Dingfang, "Ontology-based semantic matchmaking approach," *Adv. Eng. Softw.*, vol. 38, no. 1, pp. 59–67, Jan. 2007.
[5] G. Cassar, P. Barnaghi, W. Wang, and K. Moessner, "A hybrid semantic matchmaker for IoT services," in *Proc. IEEE Int. Conf. Green Comput. Commun.*, Nov. 2012, pp. 210–216.
[6] 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," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Affiliated Events (PerCom Workshops)*, Mar. 2022, pp. 433–438.
[7] M. Ruta, F. Scioscia, G. Loseto, A. Pinto, and E. Di Sciascio, "Machine learning in the Internet of Things: A semantic-enhanced approach," *Semantic Web*, vol. 10, no. 1, pp. 183–204, Dec. 2018.
[8] C. Malewski, A. Broring, P. Maue, and K. Janowicz, "Semantic matchmaking & mediation for sensors on the sensor web," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 3, pp. 929–934, Mar. 2014.
[9] H. Zhao, "Semantic matching across heterogeneous data sources," *Commun. ACM*, vol. 50, no. 1, pp. 45–50, Jan. 2007.

[10] Z. Peng, G. Xin, Y. Wei, W. Wang, B. Wang, and L. Wang, "Short text clustering enhanced by semantic matching model," in *Proc. 2nd Int. Conf. Inf. Syst. Comput. Aided Educ. (ICISCAE)*, Sep. 2019, pp. 480–484.
[11] I. B. G. Sarasvananda, R. Wardoyo, and A. K. Sari, "The K-means clustering algorithm with semantic similarity to estimate the cost of hospitalization," *Indonesian J. Comput. Cybern. Syst.*, vol. 13, no. 4, pp. 313–322, 2019.
[12] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Ontology matching: A machine learning approach," in *Handbook on Ontologies*. Cham, Switzerland: Springer, 2004, pp. 385–403.
[13] Y. Li, D. McLean, Z. Bandar, J. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1138–1150, Aug. 2006.
[14] M. Fazel-Zarandi and M. S. Fox, "Semantic matchmaking for job recruitment: An ontology-based hybrid approach," in *Proc. 8th Int. Semantic Web Conf.*, vol. 525, 2009, p. 2009.
[15] R. Zhang and N. El-Gohary, "A machine-learning approach for semantic matching of building codes and building information models (BIMs) for supporting automated code checking," in *Proc. Int. Congr. Exhib. Sustain. Civil Infrastruct.* Cham, Switzerland: Springer, 2019, pp. 64–73.
[16] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 1–9.
[19] R. Agarwal, D. Gomez, T. Elsaleh, L. Sanchez, J. Lanza, and A. Gyrard, "Federated interoperable semantic IoT/cloud testbeds and applications," Zenodo, Reston, VA, USA, Tech. Rep., Apr. 2016, doi: 10.5281/zenodo.1193299.
[20] W3C. *Web of Things (WoT) Testing*. Accessed: Apr. 15, 2022. [Online]. Available: https://github.com/w3c/wot-testing/
[21] O. D. Model. *Onedm SDF Playground*. Accessed: Apr. 15, 2022. [Online]. Available: https://github.com/one-data-model/playground/
[22] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, and V. Issarny, "Unified IoT ontology to enable interoperability and federation of testbeds," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 70–75.
[23] E. Karabulut, "Ml-based data classification and data aggregation on the edge," M.S. thesis, Dept. Inform., Tech. Univ. Munich, Munich, Germany, May 2022.

ERKAN KARABULUT received the B.Sc. degree in computer engineering from Yildiz Technical University, Istanbul, in 2019, and the M.Sc. degree in computer science from TU Munich, in 2022. He is currently pursuing the Ph.D. degree with the INtelligent Data Engineering Laboratory (INDElab), University of Amsterdam. He was a Research Assistant with the fortiss—Research Institute of the Free State of Bavaria for software-intensive services and systems, for two years, until 2022. His current research interests include semantics in the IoT, sensor networks, edge computing, and digital twins.

RUTE C. SOFIA (Senior Member, IEEE) received the Ph.D. degree, in 2004. She co-founded the Portuguese startup Senception Lda, a startup focused on personal communication platforms, from 2013 to 2019. She was the COPELABS Scientific Director, from 2013 to 2017, where she was a Senior Researcher, from 2010 to 2019. She is currently the Industrial IoT Head with the fortiss—Research Institute of the Free State of Bavaria for software-intensive services and systems. She is also an Invited Associate Professor with Universidade Lusófona de Humanidades e Tecnologias and an Associate Researcher with ISTAR, Instituto Universitário de Lisboa. She is the Co-Founder of the COPELABS Research Unit. Her research background has been developed in industrial and academic contexts. She holds more than 70 peer-reviewed publications in her fields of expertise, one book, 14 book chapters, and nine patents. Her current research interests include network architectures and protocols, the IoT, edge computing, in-network computing, and network mining. She was an ACM Senior Member. She is an ACM Europe Councilor, from 2021 to 2025. She was an IEEE ComSoc N2Women Awards Co-Chair, from 2020 to 2021.