Identification of Bluetooth-Enabled IoT Devices Through Syntactic Similarity Techniques

Argyro Mavrogiorgou, Athanasios Kiourtis, Marios Touloupou, Dimosthenis Kyriazis
Department of Digital Systems
University of Piraeus
Piraeus, Greece
{margy, kiourtis, mtouloup, dimos}@unipi.gr

***Abstract*—Moving into 2019, the number of new devices is growing exponentially, expecting more than 50 billion Bluetooth-enabled devices to be worldwide connected by 2020. As a result, emerging back-end support technologies not only have to anticipate this dramatic increase of connected devices, but also their heterogeneity. Considering this existing gap in the Internet of Things (IoT) area, this paper proposes a mechanism for discovering, connecting, and identifying heterogeneous Bluetooth-enabled IoT devices of unknown device type, in order to be integrated into different platforms. This mechanism implements three (3) stages to successfully identify an unknown device. Initially, the discovery and the connection of both known and unknown devices take place so as to retrieve explicitly required specifications of these devices. Sequentially, the measurement of the syntactic similarity of all the devices' specifications and the comparison among the derived results occurs. Based on these results, the estimation of the overall syntactic similarity takes place, identifying finally the device type of the connected unknown devices. The prototype associated with this paper provides an example of this mechanism, demonstrating in detail each stage.**

Keywords*—*IoT devices, Bluetooth-enabled devices, heterogeneous devices, device type, syntactic similarity

I. Introduction

It is an undeniable fact that the Internet of Things (IoT) and its ability to offer new technologies for services and decision-making, have made it one of the fastest growing technologies today [1]. Even though the term IoT made its first appearance in a presentation made in 1999 about Radio Frequency Identification [2], 19 years go by, and the number of existing IoT devices never stopped growing, having even surpassed the number of existing people on the planet in 2008. In last years, the number of new devices has grown exponentially and if today there exist 15 billion devices, in 2020 it is anticipated that there will be 50 billion devices worldwide [1]. Even though each one of these devices has its specific way of functioning, all of them have the common feature of being able to communicate, using mainly the Bluetooth technology.

However, all these IoT devices are characterized by a high degree of heterogeneity, thus needing applications to deal with each specific device, understand its nature, and finally use its properties and data. As a result, IoT developers should build applications that are independent of hardware and software capabilities, being able to dynamically integrate devices of unknown nature [3]. However, existing technologies lack sufficient flexibility to adapt to these changes, as their integration techniques are both static and sensitive to new or changing devices' implementations and requirements [4]. Consequently, devices' identification and integration is a very challenging research topic in the IoT era.

To address this challenge, in this paper a mechanism is proposed for dynamically connecting and recognizing Bluetooth-enabled IoT devices of both known and unknown nature (i.e. device type), by exploiting the syntactic similarity technique. Hence, through this mechanism it becomes feasible to identify the type of an incoming device, which was unknown in advance, thus specifying whether this device is related with each specific platform that wants to be connected, and as a result whether its data is relevant to the platform and needs to be collected. In order to achieve that, initially the mechanism discovers and the connects all the available both known and unknown devices, so as to retrieve some useful specifications that they contain. In sequel, it captures the syntactic similarity of these specifications, and compares it with the corresponding specifications' syntactic similarity of some pre-existing known devices. Finally, it estimates the overall syntactic similarity among these devices' specifications, resulting into the identification of the nature of the connected unknown devices based upon the similarity that they had with the pre-existing known devices. All these steps are tested and evaluated by applying this mechanism in a specific use case.

This paper is organized as follows. Section II describes the study of the state of the art regarding the approaches that exist for recognizing, and thus integrating heterogeneous IoT devices of unknown device type into platforms, focusing on the approach of the syntactic similarity that is going to be exploited in the current research. Section III describes the developed mechanism for connecting and recognizing heterogeneous IoT devices of unknown device type, Section IV analyses a use case based upon the proposed mechanism, while Section V is addressing our conclusions and plans.

II. Related Work

There exist various methods and technologies for categorizing and integrating heterogeneous IoT devices into platforms. For instance, the authors in [5] developed a framework for integrating specific data sources, by using the Ontology Web Language (OWL) to model and describe them, whilst in the same concept, the authors in [6] proposed an ontology-based cognitive computing eHealth system for integrating heterogeneous IoT fitness devices and wellness appliances. Moreover, the authors in [7] proposed a smart-home platform that offers an open and unified Application Programming Interface (API) for accessing apriori known data sources. Another example is the one that is proposed in [8], where the authors developed a web-based platform for integrating heterogeneous devices with applications. Xively [9] is an additional example, offering a cloud-based IoT platform for managing data derived from various known devices, by providing a RESTful API for connecting to the underlying device, and thus getting their data. Finally, S³OIA [10] is another proposed approach for integrating various devices, which had developed a service-oriented architecture for integrating various devices in the IoT context, by using a tuple space approach to semantically express information about the devices that were integrated into the platform.

Hence, it becomes clear that there exist various approaches for recognizing and integrating different IoT devices into different platforms. However, in the current research, the method of syntactic similarity is used for addressing the aforementioned challenge, by comparing the specifications of the different devices, and thus identify their nature. In more detail, syntactic similarity techniques are considered to have an important role in text related research and applications in tasks, as they are widely used in the majority of the Natural Language Processing (NLP) tasks, including information retrieval, text classification, document clustering, etc., whilst they are currently used in areas including fraud detection, fingerprint analysis, plagiarism detection, and ontology merging. Finding similarity between strings is a fundamental part of syntactic similarity, which is then used as a primary stage for sentence, paragraph, and document similarities. In order to calculate the syntactic similarity between two (2) different words, different string metrics have been developed over the years in order to address different needs and applications.

Among the most widely known string metrics that are currently most used are the Levenshtein distance, the City Block distance, the Jaro-Winkler distance, the Jaccard similarity, and the Grammar-based or TF/IDF distance [11]. In more detail, the authors in [12] presented a hybrid plagiarism detection method by investigating the use of a diagonal line, which was derived from the Levenshtein distance and simplified Smith-Waterman algorithm, with a view to the application in the plagiarism detection. Moreover, the authors in [13] used the City Block distance to compute the membership functions of fuzzy sets and find initial partition of a dataset, thus being able to handle minute differences between two (2) miRNA expression profiles. In the same context, the authors in [14] used the Cosine and Jaccard Similarity for information retrieval so as to deal with the problem of document similarity out of a large amount of data. Furthermore, in [15] the authors used the Cosine distance of TF/IDF weighted document vectors for a quick and reliable way to align documents. Finally, the authors in [16] proposed the Jaro-Winkler Distance and Naive Bayes Classifier in order to identify pests and diseases of paddy, taking as an input the text that contained symptoms of the disease, and utilizing Jaro-Winkler Distance to identify symptoms from user input.

As it can be identified, different string metrics can be used in multiple domains and areas, based on the corresponding needs and purposes of each different domain. With that in mind, in this paper we are going to extend the list of the areas where syntactic similarity has been used, by implementing it in the context of IoT devices' identification and discovery, using the Levenshtein distance based on the results of the research of [11]. Shortly, in this research four (4) of the mostly used syntactic similarity metrics were implemented in healthcare ontologies (Levenshtein distance, Cosine similarity, Jaro-Winkler distance and Jaccard similarity), resulting that the Levenshtein distance provided more reliable results.

III. Proposed Approach

In our approach, a mechanism is proposed for dynamically connecting and recognizing IoT devices of both known and unknown nature (i.e. device type). This mechanism consists of three (3) discrete stages: (i) Devices Discovery & Connection, (ii) Specifications Syntactic Similarity, and (iii) Overall Syntactic Similarity, as depicted in Fig. 1. In more detail, in the first stage the discovery and the connection of both known and unknown devices takes place in order to retrieve some useful specifications that they contain (i.e. name, MAC address). Sequentially, in the second stage, the measurement of the syntactic similarity of the devices' specifications occurs, capturing the syntactic similarity among the different devices both names and MAC addresses. Thus, in the final stage, the estimation of the overall syntactic similarity takes place, taking into consideration the results of the second stage. To this end, it should be mentioned that this is an extension of our previous work in [17], where we were capturing and comparing specific specifications (i.e. measurements, length, width, height, weight, and type) of different devices of unknown type so as to recognize their type. In this paper, we extended this work and enhanced its results by capturing two (2) additional specifications (i.e. name, MAC address) of these devices, in order to calculate their syntactic similarity with a number of pre-existing known devices' specifications, and combine these results with the results of our previous work.

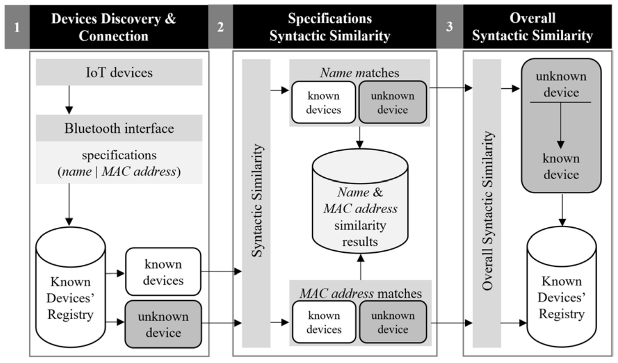

Fig. 1. Overall architecture.

A. Devices Discovery & Connection

In the first stage of the mechanism, the discovery as well as the connection of the available heterogeneous both known and unknown IoT devices (in terms of known and unknown device type) takes place, followed by the collection of their specifications, as depicted in Fig. 2.

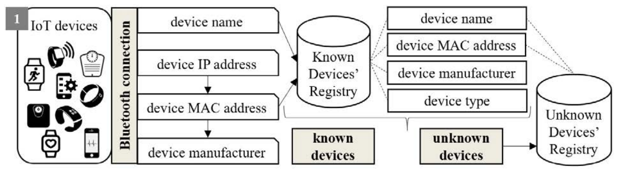

Fig. 2. Devices discovery & connection stage.

In deeper detail, the various IoT devices are discovered and connected to the mechanism through the provided Bluetooth interface, which is responsible for collecting various information about the devices' specifications. This interface is based upon the Bluetooth Low-Energy (BLE) [18], which is a core protocol for IoT applications, having become the most commonly used protocol in the era of IoT

[19]. Therefore, through this interface, the mechanism is able to communicate with devices that can be connected only via Bluetooth, gathering information about the Internet Protocol (IP) address of each one of these devices that want to be connected to the mechanism. As soon as the IP address of each device is collected, through the latter it becomes feasible to identify each device's Media Access Control (MAC) address that is a unique identifier for it. It should be mentioned that part of this address is reserved for the Organizationally Unique Identifier (OUI) that allows the association of MAC addresses to a specific manufacturer. In our case, this information is very useful, as the MAC address in combination with the name of the manufacturer in which each device belongs to is considered to be a characteristic specification for the proposed mechanism. However, this information considers to be poor as it only reveals the name of the manufacturer, whereas sometimes no organization is associated with the MAC address received. In many cases, one manufacturer may produce many different devices that will have the same OUI, thereby preventing the use of this specification for the devices' discovery. For this reason, apart from the IP address of each device, the mechanism gathers information about the name of each device as well, so as to be used and combined with the MAC address of the corresponding device in the next steps of the mechanism.

Henceforth, as soon as all the devices are connected and their useful specifications (i.e. name and MAC address) are captured, the latter are queried and automatically compared with the corresponding specifications of a number of devices that already exist in a created registry that includes information of known IoT devices (i.e. known devices' registry). Through this way, we are able to find out whether the connected devices already exist in this registry or not, and thus they will be characterized as either known or unknown correspondingly. More specifically, if the specifications of the connected devices match exactly with the specifications of the known devices' registry, then the connected devices are automatically considered to be known (i.e. devices of known type). On the contrary, if the comparison concludes that there is neither a resemblance nor even a partial resemblance among the specifications of the connected devices and the known devices' registry, then the connected devices are considered to be unknown (i.e. devices of unknown type). Thus, all the connected devices are characterized to be either of known or of unknown device type.

After capturing the specifications of the different devices and categorizing them, their information storage occurs, where the specifications of the unknown devices are stored into a new registry (i.e. unknown devices' registry). Consequently, the latter consists of the fields of the name and the MAC address of each unknown device. With regards to the specifications of the known devices, these do not have to be stored, as they already exist in the known devices' registry that consists of the fields of the name, the MAC address, the manufacturer, and the device type of each known device.

B. Specifications Syntactic Similarity

In the second stage of the mechanism, the syntactic similarity that exists between the existing known IoT devices and each connected device's specifications takes place, so as to categorize the unknown devices to the device type of the known ones, depending on the similarities that they will have among their specifications (i.e. name and MAC address). It is worth mentioning, that this stage takes place only for the connected devices that are of unknown device type, as the known ones already exist in the known devices' registry and their device type is apriori known. Henceforth, the names and the MAC addresses of the connected unknown devices that have derived from the previous step are taken as an input, in order to be compared with the corresponding names and MAC addresses of the devices that can be found in the known devices' registry.

In order to identify the syntactic similarity of the aforementioned, the following steps are followed with regards to devices' name and MAC address similarities. For both cases, as stated in Section II, the Levenshtein string metric is used, which operates between two (2) input strings, returning a number equivalent to the number of substitutions and deletions needed in order to transform one input string into another. In more detail, in the case of Levenshtein string metric, transformations are the one-step operations of insertion, deletion, and substitution. In the simplest versions, substitutions cost two (2) units, except when the source and the target are identical, where the cost is zero, whereas insertions and deletions cost half than that of substitutions. The algorithm starts with the upper left-hand corner of a two-dimensional array indexed in rows by the letters of the source word, and in columns by the letters of the target. It fills out the rest of the array while finding all the distances between each initial prefix of the source, and each initial prefix of the target. Each [i,j] cell represents the (minimal) distance between the first i letters of the source word and the first j letters of the target word. Mathematically, the Levenshtein distance between two (2) strings a, b (of length i and j respectively) is given by $\text{lev}_{a,b(i,j)}$ as stated in equation (1). More specifically, $1_{(ai \neq bi)}$ is the indicator function equal to 0 when ai≠bi and equal to 1 otherwise, and $\text{lev}_{a,b(i,j)}$ is the distance between the first i characters of a and the first j characters of b. The first element in the minimum corresponds to deletion (from a to b), the second to insertion, and the third to match or mismatch, depending on whether the respective symbols are the same or not.

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j)=0, \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j)+1 \\ \text{lev}_{a,b}(i,j-1)+1 \\ \text{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \quad (1)$$

Therefore, regarding the devices' name similarity, the developed mechanism gets as an input the name of each connected unknown device and the names of the devices that can be found in the known devices' registry (Fig. 3). Henceforth, for each different unknown device's name, following equation (1), the Levenshtein distance is calculated between each unknown device's name and each different known device's name. By the time that all the different combinations of comparisons have occurred, different registries of the calculated syntactic similarities are created. Therefore, this sub-mechanism probabilistically identifies the name of the device of the known devices' registry for which each connected unknown device has greater degree of resemblance, and as a result its name matches with this corresponding known device's name. However, since this sub-mechanism provides syntactic similarity results concerning only the name of each connected unknown device, an additional sub-mechanism is constructed for measuring the devices' MAC addresses similarity.

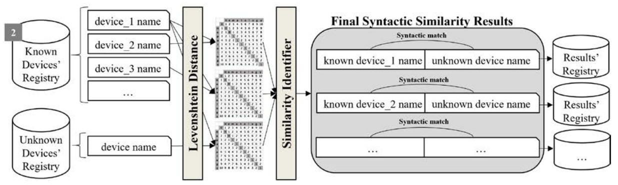

Fig. 3. Syntactic similarity stage based on name.

Regarding the devices' MAC address similarity, the developed mechanism gets as an input the MAC address of each connected unknown device and the different MAC addresses of the devices that can be found in the known devices' registry (Fig. 4). As mentioned in the previous stage, through the IP address we are able to find out the MAC address of each device from which it derives an OUI, which is a 24-bit number that uniquely identifies the manufacturer of the device. In every case, the OUI is basically the first three (3) octets of a MAC address. With that in mind, the developed sub-mechanism splits the first three (3) octets of each MAC address of the known devices and compares them with the corresponding first three (3) octets of each connected unknown device's MAC address. Hence, for each unknown device's MAC address, following equation (1), the Levenshtein distance is calculated between each unknown device's MAC address and each different known device's MAC address that exists in the known devices' registry. As in the previous case, by the time that all the different combinations of comparisons have occurred, different registries of the calculated syntactic similarities are created. Thus, this sub-mechanism identifies the MAC address of the device of the known devices' registry for which each connected unknown device has greater degree of resemblance, and as a result a part of its MAC address matches with this corresponding part of the known device's MAC address. As soon as this comparison is completed, through the MAC address of the known device that is recognized to be the most similar with the MAC address of the unknown device, we are able to find the name of the corresponding manufacturer that is stored in the known devices' registry for this specific known device's MAC address, and as a result, it will be the same manufacturer for the unknown device. As in the previous sub-mechanism, this part provides syntactic similarity results concerning only the MAC address of each connected unknown device. Hence, an additional mechanism needs to be implemented for measuring the overall syntactic similarity, based on both the name and the MAC address similarity of the connected unknown devices, in order to finally identify their device type.

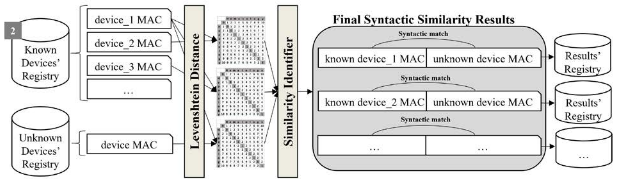

Fig. 4. Syntactic similarity stage based on MAC address.

C. Overall Syntactic Similarity

In the third stage of the mechanism, having calculated the aforementioned syntactic similarities, the aggregation of the derived results occurs (Fig. 5). Consequently, the mechanism queries through the values that have been calculated for each different case and provides a mean between these results. This mean (i.e. Overall mean) is calculated by the equation (2), representing the total of the name and the MAC address syntactic similarities of each combination of compared devices, divided by two:

$$\text{Overall mean} = \frac{\text{Similarity (Name)} + \text{Similarity (MAC)}}{2} \qquad (2)$$

After the calculation of the Overall mean, the device of the known devices' registry with higher probability of similarity with each connected unknown device is automatically considered that it is identical or almost-identical with the corresponding connected unknown device. As a result, the device type of each connected unknown device becomes known, as it has been categorized into one of the groups of the devices with the known device type. To this end, it should be noted that in order to consider the results as trustful and reliable, these must exceed the set threshold of 85% Overall mean. Henceforth, each unknown device that transcends this threshold, is considered as known, providing us detailed information about its name, MAC address, manufacturer, and device type. Thus, each one of these connected recognized devices in combination with their corresponding specifications are finally stored in the registry of the known devices' registry, so as to be considered as known devices for future usage.

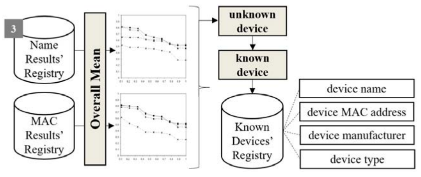

Fig. 5. Overall syntactic similarity stage.

IV. Use Case

A. Use Case Description

In order to perform a complete testing and evaluation of the proposed mechanism, 28 IoT devices were chosen that could communicate via Bluetooth with our mechanism. In more detail, 26 of them were of known device type, whereas 2 of them were of unknown device type. For the 26 devices we had prior knowledge about four (4) of their specifications (i.e. name, MAC address, manufacturer, and type), outlining the type of the device that each one of them belonged to (i.e. smart blood pressure monitor (BPM), glucometer (GL), body weight scale (BWS), activity tracker (AT), and toothbrush (TB)). However, regarding the 2 devices, we had only prior knowledge about two (2) of their specifications (i.e. name, and MAC address), whereas their manufacturer and type were unknown (UN), as depicted in Table I. To this end, it should be mentioned that for finding the corresponding manufacturer for each OUI of the different MAC addresses, the MAC Vendors API was used [20], through which it became feasible to find the corresponding manufacturer of each OUI of the different MAC addresses.

TABLE I. Devices' Specifications

#	Name	Mac Address	Manufacturer	Type
1	Fitbit Flex	1800DB6C572B	Fitbit	AT
2	Withings BPM	0024E46C58B4	Withings	BPM
3	iHealth BP3L	00070D6E97A2	iHealth	BPM
4	Oral-B 7000	0012216C4B49	Braun	TB
5	TaiDoc TD-3128	0026DF6C4BCE	Taidoc	BPM
6	ARA toothbrush	00078E6B71A3	Kolibree	TB

7	Garmin Vivomove	00054F6C6192	Garmin	AT
8	iHealth HS4S	00070D6E58A3	iHealth	BWS
9	iHealth BG5	00070D6E56B4	iHealth	GL
10	Garmin Vivosport	00054F6BD535	Garmin	AT
11	Fitbit Charge 2	1800DB6BE0BD	Fitbit	AT
12	iHealth AM4	00070D6E24C2	iHealth	AT
13	Oral-B Genius	0012216C42DA	Braun	TB
14	Apple Watch	68EF436A14A2	Apple	AT
15	Fitbit Zip™	1800DB6C6713	Fitbit	AT
16	Fitbit Aria®	1800DB6C572C	Fitbit	BWS
17	iHealth BP5	00070D6E27B5	iHealth	BPM
18	Withings Pulse O2	0024E46C4D4A	Withings	AT
19	iHealth 550BT	00070D6E26A1	iHealth	BPM
20	iHealth BP7	00070D6E97B1	iHealth	BPM
21	Garmin Vivofit	00054F6BD3AB	Garmin	AT
22	TaiDoc TD-2555	0026DF6BE307	Taidoc	BWS
23	Fitbit Alta HR	1800DB6C2CAA	Fitbit	AT
24	TaiDoc TD-4277	0026DF6C2A2B	Taidoc	GL
25	iHealth HS6	00070D6E25C2	iHealth	BWS
26	Pebble Watch	34049E6C57E1	Pebble	AT
27	Garmin Vivosmart	00054F6BD52B	UN	UN
28	Xiaomi Mi Band 2	D832E36BE105	UN	UN

B. Experimental Results

The proposed mechanism was developed in Java SE using the NetBeans IDE v8.0.2 [21], and used a processing environment with 16GB RAM, Intel i7-4790 @ 3.60 GHz x 8 CPU Cores, 2TB Storage, and Windows 10 operating system. The results of the mechanism are depicted below, following the three (3) stages explained in Section III.

Devices Discovery & Connection. Initially, in the first stage, 2 devices wanted to be connected to the mechanism (i.e. the last 2 entries highlighted with grey in Table I), whereas 26 devices already existed in the known devices' registry (i.e. the first 26 entries of Table I). Therefore, as soon as the 2 devices were recognized and connected to the mechanism, their specifications (i.e. name and MAC address) were captured. Sequentially, these were compared with the specifications of the 26 devices that already existed in the known devices' registry, concluding that the 2 devices were not included in the registry. Thus, they considered to be of unknown device type, and their specifications were stored into a new registry (i.e. unknown devices' registry).

Specifications Syntactic Similarity. Afterwards, in the second stage, the names and the MAC addresses of the 2 unknown devices in combination with the different names and MAC addresses of the 26 devices that existed in the known devices' registry were given as an input. As described in Section III, for each different unknown device's name the syntactic similarity was calculated, implementing the Levenshtein distance between each unknown device's name and each different known device's name that existed in the known devices' registry. As soon as all the different combinations of comparisons had occurred, the mechanism concluded into the results of the columns Levenshtein Name of Tables II and III correspondingly, whilst these results were stored into different registries. The same process was followed for each different unknown device's MAC address, where the syntactic similarity was calculated again through the use of the Levenshtein distance between each unknown device's MAC address and each different known device's MAC address that existed in the known devices' registry. As soon as all the different combinations of comparisons had occurred, the mechanism concluded into the results of the columns Levenshtein MAC of Tables II and III correspondingly, whilst these results were stored into different registries.

TABLE II. GARMIN VIVOSMART SPECIFICATIONS' SYNTACTIC RESULTS

Unknown device Garmin Vivosmart -------------------- **00054F6BD52B**	**Known devices' Name** ------------ **MAC**	**Levenshtein Name** ------------ **MAC**	**Overall**
	Garmin Vivomove 00054F6C6192	0.71 1.00	0.86
	Garmin Vivosport 00054F6BD535	0.88 1.00	0.94
	Garmin Vivofit 00054F6BD3AB	0.79 1.00	0.89
	Withings Pulse O2 0024E46C4D4A	0.36 0.21	0.28

TABLE III. XIAOMI MI BAND SPECIFICATIONS' SYNTACTIC RESULTS

Unknown device Xiaomi Mi Band 2 -------------------- **D832E36BE105**	**Known devices' Name** ------------ **MAC**	**Levenshtein Name** ------------ **MAC**	**Overall**
	Garmin Vivofit 00054F6BD3AB	0.12 0.03	0.08
	Garmin Vivomove 00054F6C6192	0.24 0.03	0.14
	Withings Pulse O2 0024E46C4D4A	0.11 0.13	0.12
	Apple Watch 68EF436A14A2	0.03 0.01	0.02

Overall Syntactic Similarity. Finally, in the third stage, having calculated the aforementioned syntactic similarities, their aggregation took place, concluding into the columns Overall of Tables II and III correspondingly. To this end, it should be mentioned that in Tables II and III are outlined only the top four (4) results of the comparisons of the connected unknown devices, according to the results of the calculation of the Overall mean.

C. Discussion of Results

As it has been discussed in Section III, after the calculation of the Overall mean, the known device with higher probability of overall syntactic similarity with each connected unknown device is automatically considered to be identical or almost-identical with the latter. Thus, through Table II we identified that Garmin Vivosmart had greater degree of similarity with Garmin Vivosport, concerning both its name and MAC address, having a percentage of 94% overall similarity that far exceeded the set threshold of 85%. Thus, Garmin Vivosmart belonged to the Garmin manufacturer, while its type was AT, as all the Garmin devices of Table I were AT. Moreover, through Table III we identified that Xiaomi Mi Band 2 had greater degree of similarity with Garmin Vivomove, having a percentage of 14% overall similarity, a number that was extremely low compared with the set threshold of 85%. Therefore, this result was not efficient enough to conclude neither to the manufacturer nor to the type of the device. Thus, since the results of neither the name similarity nor the MAC address similarity were sufficient enough, it was not possible to map Xiaomi Mi Band 2 to any of the devices of the known devices' registry. Consequently, this device could not be recognized, and its device type remained unknown. Fig. 6 and Fig. 7 visualize all the captured results, comparing the derived overall syntactic similarity results of each different case with the case of the exact match (i.e. 100%).

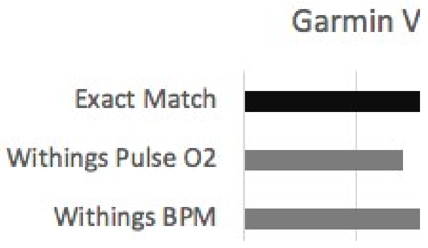

Fig. 6. Garmin Vivosmart overall mean results.

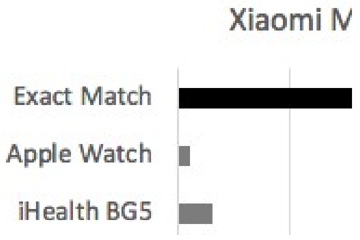

Fig. 7. Xiaomi Mi Band 2 overall mean results.

Based on the derived results, we can conclude that through this mechanism the device type of the connected unknown device is feasible to become known, as it can be categorized into one of the groups of the devices with the known device type (Table II). Besides the aforementioned, through the derived results it is clear that it is not possible to create rules or patterns, as an unknown device's either name or MAC address may have syntactic similarity with a specific device, but there might be some cases where the unknown device may not have any similarity with any other known device, and thus further information of this unknown device is needed to recognize its device type (Table III).

V. CONCLUSIONS

It is an undeniable fact that devices' identification and integration is a very challenging research topic in the IoT area. For that reason, in this paper we have studied the challenging topic of recognizing heterogeneous Bluetooth-enabled IoT devices of both known and unknown nature. Therefore, we have considered data coming from devices of both known and unknown device type, and proposed a mechanism for facilitating the identification of different IoT devices and the prediction of their devices' type, based on some of their specifications. In this mechanism, three (3) discrete stages were followed to identify the devices' type by estimating the syntactic similarity among some pre-existing known devices' specifications and the incoming unknown devices' specifications, thus matching the specifications between the known and the unknown devices, and as a result finding out the device type of the unknown devices.

Currently, we are working on the evaluation of the proposed mechanism by testing it with more heterogeneous Bluetooth-enabled IoT devices, considering additional combinations of specifications. Our future work includes the update of this mechanism by not only offering a Bluetooth interface, but also a general interface in which all the devices will be able to be connected regardless of their communication protocol. Apart from this, we are willing to extend our mechanism by capturing additional similarity metrics. On top of this, we are willing to compare the proposed approach with other existing methods that are not based on syntactic similarity, so as to evaluate its overall applicability and effectiveness. Finally, we aim to implement a security module for ensuring the originality of the devices' specifications, avoiding potential spoof of them (e.g. MAC address spoofing) that could result into misleading decisions.

ACKNOWLEDGMENT

The research leading to this result has received funding from the European Union's Horizon 2020 research and innovation programmes under grant agreement No 826106 (InteropEHRate project) and grant agreement No 727560 (CrowdHEALTH project).

REFERENCES

[1] J. Q. Anderson, et. al., "The Internet of Things Will Thrive by 2025", 2014.

[2] K. Ashton, "That 'internet of things' thing", RFiD Journal, 22(7), pp. 97 – 114, 2009.

[3] A. M. Nagib, et. al., "SIGHTED: A Framework for Semantic Integration of Heterogeneous Sensor Data on the Internet of Things", Procedia CS, pp. 529 – 536, 2016.

[4] P. Gong, "Dynamic integration of biological data sources using the data concierge", HISS, 1(1), pp. 7, 2013.

[5] B. Pötter, et. al., "Adapting heterogeneous devices into an IoT context-aware infrastructure", Software Engineering for Adaptive and Self-Managing Systems, pp. 64 – 74, 2016.

[6] A. Carbonaro, et. al., "Integrating Heterogeneous Data of Healthcare Devices to enable Domain Data Management", Journal of e-Learning and Knowledge Society, 14(1), 2018.

[7] M. Moazzami, et. al., "SPOT: A smartphone-based platform to tackle heterogeneity in smart-home IoT systems", World Forum on Internet of Things, pp. 514 – 519, 2016.

[8] P. F. Pires, et. al., "A platform for integrating physical devices in the Internet of Things", EUC, pp. 234 – 241, 2004.

[9] N. Sinha, et. al., "Xively based sensing and monitoring system for IoT", Computer Communication and Informatics, pp. 1 – 6, 2015.

[10] M. Vega-Barbas, et. al., "Smart Spaces and Smart Objects Interoperability Architecture (S^3OIA)", Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 725 – 730, 2012.

[11] A. Kiourtis, et. al., "A String Similarity Evaluation for Healthcare Ontologies Alignment to HL7 FHIR Resources", Computing Conference 2019, in press, 2019.

[12] Z. Su, et. al., "Plagiarism detection using the Levenshtein distance and Smith-Waterman algorithm", ICICIC, pp. 569-569, 2008.

[13] S. Paul, et. al., "City block distance and rough-fuzzy clustering for identification of co-expressed micrornas", Molecular BioSystems, 10(6), pp. 1509 – 1523, 2014.

[14] A. Jain, et. al., "Information retrieval using cosine and jaccard similarity measures in vector space model", International Journal of Computer Applications, 164(6), 2017.

[15] C. Buck, et. al., "Quick and reliable document alignment via tf/idf-weighted cosine distance", In Proceedings of the First Conference on Machine Translation, Vol. 2, pp. 672 – 678, 2016.

[16] A. P. Sari, et. al., "The Implementation of Jaro-Winkler Distance and Naive Bayes Classifier for Identification System of Pests and Diseases on Paddy", ITSMART, 7(1), pp. 1 – 7, 2018.

[17] A. Mavrogiorgou, et. al., "A comparative study of classification techniques for managing iot devices of common specifications", GECON, pp. 67 – 77, 2017.

[18] Gomez, C., et. al., "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology", Sensors, 12(9), pp. 11734 – 11753, 2012.

[19] Bluetooth 5 and its Role in the Internet of Things, https://www.iotforall.com/bluetooth-5-iot/

[20] Find MAC Address Vendors Now, https://macvendors.com/

[21] NetBeans IDE, https://netbeans.org/