Open Access

Open Journal of Internet of Things (OJIOT)
Volume 7, Issue 1, 2021

http://www.ronpub.com/ojiot
ISSN 2364-7108

A Graph-Based Web Services Discovery Framework for IoT EcoSystem

Ivan Madjarov[A], Fatma Slaimi[B]

[A]Aix-Marseille Univ, Université de Toulon, CNRS, LIS, DIAMS, 52 Av. Escadrille Normandie Niemen, 13397 Marseille Cedex 20, France, ivan.madjarov@lis-lab.fr
[B]IUT Aix-Marseille, CNRS, LSIS, 163 Avenue de Luminy, 13288 Marseille cedex 9, France, fatma.slaimi@univ-amu.fr

Abstract

Nowadays, the Internet of Things (IoT) represents an important topic and research domain with multiple objectives. However, most IoTs communicate poorly across the multitude of network interfaces. It should be preferably used a single universal application layer protocol for the devices and services interconnection, regardless of how they are physically connected. The IoT paradigm boosts the device connectivity and the user accessibility benefits of services introduced within the network of connected objects associated with a context-awareness. Within this frame of reference, Web service is the appropriate technological approach to exhibit a set of related IoT functionalities loosely coupled with other services discovered or composed through the Web. In this work, we consider the heterogeneity of connecting technologies for IoT and the applications and devices integration in a single interoperable framework as a research objective. With this in mind, we introduce a five layers multigraph model for Web Services discovery and recommendation, and we address Web services-based applications for IoT data integration. The launched service discovery process permits the interaction between the user/application and the IoT environment. In this context, the choice of suitable services represents a challenge that covers the functionality and the required quality to combine composite services, namely mashups for IoT data management and interconnection. For proof of concept, we test a RESTful Web Services framework as an experimental platform to animate a graph-based approach for dynamic IoT services discovery. We develop a recommender system that performs graph analytics to produce a set of services according to the user's request. The quality of the recommendation process is evaluated by analyzing the correlation of user satisfaction.

Type of Paper and Keywords

Short Communication: *Internet of Things, Web of Things, Graph-based Semantic, Web Services Discovery*

1 Introduction

There is a large number of devices that make a connection between the digital and the physical world. To sense the activity in the surrounding area services are provided in different domains, e.g., smart homes, smart cities, industrial assembly machinery, intelligent transport, and even health care services. Devices providing sensing, actuation, control, and monitoring activities are defined in [39] as the Internet of Things (IoT) ecosystem.

Wireless technologies offer significant opportunities to facilitate data exchange between connected devices and related applications in an independent manner. It is a simplification for sensors, controllers, and actuators installation. In such domains, services-based applications are developed or discovered to collect and process data issued from IoT devices.

In this work, we consider Web service as a type of M2M (Machine-to-Machine) software application providing a platform for IoT interconnections via RFID[1], Wi-Fi[2], Li-Fi[3], IP[4], etc. Web services manage the information gathered by IoT devices via network connections.

As argued in [10], Web service discovery is a process that facilitates the implementation of complex and reconfigurable applications in a service-oriented architecture. However, most of the applied approaches for IoT device management are designed in a centralized way, whose efficiency recently meets challenges because of the large scale of heterogeneous device modules and highly dynamic essence of the networks, as noted in [20]. It is especially true for cloud-based services [2]. However, due to an increase in the number of proposed cloud services, users feel difficulties in finding needed services in real-time with the desired relevance and identifiable provenance.

Today, most IoT communicate poorly across the multitude of network interfaces, and it is preferable to rely on a single application layer protocol for the devices and services interconnection, regardless of how they are physically connected. Instead of creating a newer one from scratch, it is preferable to reuse a protocol that is already widely applied for building scalable and interactive applications, such as the Web family application protocols. In this case, the Web service is in charge of the communication complexity of any heterogeneous connected object and webcast their data.

As many Web services could be available for an IoT environment, the Web service discovery process takes a leading role in ensuring a suitable choice. A context-aware and adaptive Web services discovery approach can be applied to an IoT scenario with an access control mechanism. From a technical point of view, Web services for IoT applications can be based on the HTTP[5] compatible REST[6] architectural style. The IoT intrinsic features are heterogeneous, and this impacts the semantic composition of the Web service discovery and composition process.

In this paper, we consider every physical and logical entity as a resource in a defined state that can be manipulated via an appropriated service. To achieve this, we present a Web services discovery and recommendation approach for an interoperable and distributed IoT ecosystem. Our objective is to perform online service discovery and recommendations when the user expresses requests. For this, a graph-based system is queried to determine services that can meet the needs of a connected object. The recommender system performs graph analytics to produce a set of recommended services or mashups[7], according to the user's request.

The contributions in this paper focus the heterogeneity of proprietary technologies and the integration of applications and devices in a single interoperable framework. With this in mind, we address Web Services for IoT applications, and we introduce a five layers multigraph semantic model for Web Services discovery and recommendation. The service discovery process permits the interaction between user requirements and the IoT environment. In this context, the choice of suitable services represents a challenge that covers the functionality and the required quality to combine services as a composite one, namely mashups for IoT data management and interconnection. For proof of concept, we test a RESTful Web Services framework as an experimental platform to animate a graph-based approach for dynamic IoT services discovery in users' spaces. We develop a recommender system that performs graph analytics to produce a set of services, according to the user's request. The quality of the recommendation process is evaluated by analyzing the correlation of user satisfaction.

The rest of this paper is organized as follows. We present our motivation and some related works in Section 2. The research approach and some technical details are presented in Section 3. The settings for the graph-based services discovery and recommendation are discussed in Section 4, the experimental environment, and evaluation in Section 5. Finally, we discuss some conclusion notes in Section 6.

2 Motivation and Related Works

2.1 IoT System Architecture

The IoT ecosystem represents a wide range of devices from simple tagged products such as QR codes[8], NFC[9], and RFID tags to sensors, actuators, computation, and communication interfaces. It's a smart network of devices (Things) that can be interconnected through Internet protocols and can be processed via applications and services with identified provenance.

[1] Radio Frequency IDentification
[2] Wireless Fidelity
[3] Light Fidelity
[4] Internet Protocol
[5] HyperText Transfer Protocol
[6] Representational State Transfer
[7] Application created by combining data or functionality from different sources
[8] Quick Response Code
[9] Near Field Communication

As argued in [13], a single universal application layer protocol is recommended to simplify Things interconnections. At first sight, this seems probably unrealistic to achieve. However, devices and applications can talk to each other regardless of the nature of their physical interconnection. As an illustration, we can note the echo-dot device (Alexa). It's a cloud-based service supported by Amazon that can play some music, tell about news, weather forecast, and much more. This gadget can also interfere with other Internet-based technologies at home and beyond. For instance, if at home a set of smart light bulbs and smart speakers are installed, a dedicated service-application can control them to turn on/off the lights or speakers in a room and synchronize them for all rooms at home. In this case, web service-based applications ensure the interconnection between the echo-dot device and the setting-up of the smart-home device (IoT).

Indoor geolocation is another example that reinforces our motivation with a solution based on a graph-path exploration. In this case, the graph vertices are presented by IoT devices (e.g., Li-Fi lights, Wi-Fi spots, NFC and RFID tags, Bluetooth devices, BLE[10], etc.), all of them TCP-IP/HTTP/REST compatible. Then, the edge (path) can be recommended by a service.

Today, many incompatible protocols co-exist in the IoT domain (e.g. XMPP[11], CoAP[12], MQTT[13], AMQP[14]) [11]. It forms a truly set of competitive application-layer communication protocols developed to satisfy the IoT ecosystem's intrinsic features. As detailed in [21], each communication protocol is designed for a specific set of application requirements, such as low power operation, service discovery, bootstrapping, etc. Consequently, the process of data and services integration for various devices becomes highly complex and costly. A part of the proposed solutions that target IoT interoperability requires all communications to be routed through a centralized server that speaks all IoT communication protocols.

Nowadays, it is obvious that building a single global ecosystem of IoT that can communicate with each other flawlessly is a difficult or even impossible task. We notice that today most IoT can't communicate across the many available networking interfaces through the fault of a unique and universal application protocol. To address this issue, it is preferable to rely on a single universal application layer protocol for the devices and applications interconnection, regardless of how they are physically connected. Instead of creating a newer protocol from scratch, it is technically possible to reuse a protocol that is already widely applied for building scalable and interactive applications, such as the Web family application protocols. The transition from an Internet of Things (IoT) to Web of Things (WoT) is informal if any device can be accessed using standard Web protocol and can offer a standard Web API and a dedicated/appropriated Web service. The Web service can take charge of the communication complexity of any heterogeneous object (IoT/WoT) and webcast their data.

Several papers [4] [26] [20] [13] propose the REST approach for IoT or WoT data and parameter processing. Our analysis points to the poor software support of the application development that should benefit from REST technology advantages. In [4], the authors propose a developer's support for WoT modeled as Web Services. The main idea is to exposing connecting objects through RESTful APIs that may facilitate the implementation of the application on top of them.

To dynamically interconnect IoT and WoT objects, authors in [28] describe a process with embedded RESTful Web services. Their suggestion focuses on interoperable communication between objects based on Web Services technology. As depicted in [35], the Web services feature promotes the ability to deal with heterogeneous sources of information, as well as the service discovery and composition. We argue for embedding Web Services into IoT and WoT objects to facilitate the dynamic integration of distributed processes. Based on the RESTful style software architecture cited in [28], the embedded Web services are considered as physical information servers for distributed systems in constrained nodes.

2.2 IoT Semantics

In [4] and [35], the authors discuss the multitude of ontologies developed to describe concepts and relationships for IoT applications. They conclude: the proposed semantic techniques increase the complexity and processing time; they are unsuitable for dynamic and responsive IoT environments. In this context, a lightweight ontology is developed by W3C [5] to describe IoT resources, entities, and services. In [32], authors develop IoT-Lite ontology to represent key IoT concepts. This IoT-Lite ontology is, in fact, an instantiation of the Semantic Sensor Network Ontology (SSN) [35]. The main issue in their approach is allowing interoperability and data discovery in heterogeneous IoT platforms. In [4], the authors proposed ten rules for scalable semantic model design and implementation for the IoT-Lite ontology. A demonstration of scalability is provided with an experimental analysis and assessment.

As presented in [7], the SSN ontology is a significant

[10] Bluetooth Low Energy
[11] Extensible Messaging and Presence Protocol
[12] Constrained Application Protocol
[13] Message Queuing Telemetry Transport
[14] Advanced Message Queuing Protocol

model to describe sensors and IoT-related concepts. This ontology provides a concept describing sensors, outputs, observation value, and data with detailed properties. Authors argue that this enables flexible processing over a wide range of applications. However, for many use cases, this ontology is complex for querying and data processing because it includes a lot of non-essential components.

In [35], two projects are compared as an extension of SSN ontology. The IoT-A and IoT.est are two semantic models to represent services and objects in sensor devices. Authors present IoT-A as a base for IoT project development. We notice that the IoT-A model is a bit complex for user adaptation in dynamic environments. The authors also present the IoT.est model as an extension of the IoT-A model with additional service and test concepts.

The Sensor Web Enablement (SWE) group, part of the Open Geospatial Consortium (OGC), has developed a set of standards and languages that described sensors and associated data, as presented in [7]. For instance, SensorML is an XML-based language that provides syntactic descriptions for sensors. However, it lacks expressibility compared to more powerful ontology languages such as OWL (Web Ontology Language). The Semantic Sensor Observation Service (SemSOS) is introduced in [18] as a Web service. It's a standardized effort in the direction of the sensor's parameters and data discovering and processing on the Web. The Web service maps the XML tags into OWL concepts as described in [18] to represent observations over IoT with limited related notions.

A literature overview in applying Semantic Web technologies to IoT is presented in [17]. The Resource Description Framework (RDF) is cited as a predominant technique for representing IoT semantics. RDF represents knowledge as triples (subject, predicate, object). A set of triples forms a graph where subjects and objects are vertices, and predicates are the edges. To express the domain knowledge, OWL is often used with RDF to define ontologies on the Web.

In [24], authors proceeded with a combination of two technologies, namely the REST architecture style and JSON[15] Schema/JSON Meta Schema. This approach permits the construction of knowledge graphs for a RESTful Web services composition. JSON-LD[16] is presented in [20] as a lightweight serialization syntax to express RDF in JSON format. Authors argue that this is an ideal data format for Web-based programming environments, RESTful Web services, and unstructured (NoSQL[17]) databases such as MongoDB[18]. A lightweight vocabulary for describing RESTful Web services using the OpenAPI Specification is presented in [9]. In our work, we adopt and develop this technical approach.

As argued in [17], Web APIs provide an efficient way of interacting between Web applications ensuring so operation and dealing with the IoT integration. This approach aims to IoT applications in smart cities, smart homes, enterprise solutions, all based on Web services. The technical contribution of this paper is based on this concept.

2.3 Web Service Discovery Process

In general, a Web-service discovery process refers to the ability to promote relevant and related resources, and in some particular cases, according to metadata collected from a historical activity. In this context, a service discovery process, based on search criteria, can allow retrieving accessible resources. Then, these resources should be described in an unambiguous, machine-interpretable way to overcome their potential heterogeneity. To act in this direction, the authors in [12] propose a solution named ForwarDS-IoT. They discuss a federated "*discovery service based on multiple attributes, range queries, and synchronous/asynchronous operations*". This approach also includes an ontology-based model for semantically describing resources dedicated to IoT services.

In [19], the authors argued that the service discovery process uses information from retrieval-based techniques. The Web services discovery process attached to the IoT management has certain particularities linked to the nature of deployed service, such as non-semantic or semantic Web services, as argued in [9] .

In Figure 1, a non-semantic Web service is depicted with details about service functionalities, data types, and access protocols generally in WSDL (Web Service Description Language) [6]. On the other hand, ontology-based languages are used to describe Semantic Web Services, where the discovery process is based on matchmaking approaches [9], further divided into semantic-based, syntax-based, and context-aware. For both, an XML-based document is used to construct the basic blocks of communication protocols, such as Simple Object Access Protocol (SOAP) [34] and Representational State Transfer (REST) [7].

The RESTful Web Services-based technology has certain advantages in IoT / WoT context over SOAP-based Web Services technology, such as reduced parsing

[15] JavaScript Object Notation

[16] JavaScript Object Notation for Linked Data

[17] Not only SQL

[18] https://www.mongodb.com/

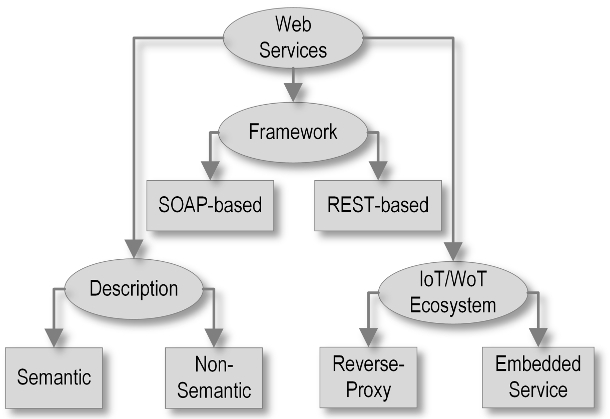

Figure 1: Web service description and implementation in IoT ecosystem

complexity and more efficient integration with HTTP protocol. Besides, as considered in [22], REST-based applications perform better on the wireless sensors as well as they are easier to implement.

In the state of implementation, there are two cases in point:

1. Devices with embedded native Wi-Fi/Bluetooth support and embedded HTTP server;
2. Devices with Web service integration through a software bridge (reverse-proxies) when native TCP/IP and HTTP support is not available as suggested in [14].

It is not realistic to consider that all devices will be powerful enough to host a Web service stack, so we argue that a bridge-based approach will be a useful complement solution. To better define the scope of our research work, we introduce a simplistic functional WoT architecture with four layers, as shown in Figure 2.

As mentioned in [3], the Device Driver Layer (DDL), presented in Figure 2, may contain two types of devices:

1. Smart sensors and appliances with embedded TCP/IP connection that can exchange data directly through RESTful Web service with user's application;
2. Devices that are too resource-constrained to be directly connected to RESTful Web Service Layer by itself.

Therefore, an additional Smart Gateway Layer (SGL) is required as an intermediate to translate and to convert the device-based protocol to RESTful Web Service protocol and vice versa.

The SGL presented in [14] is built on four levels of functionality:

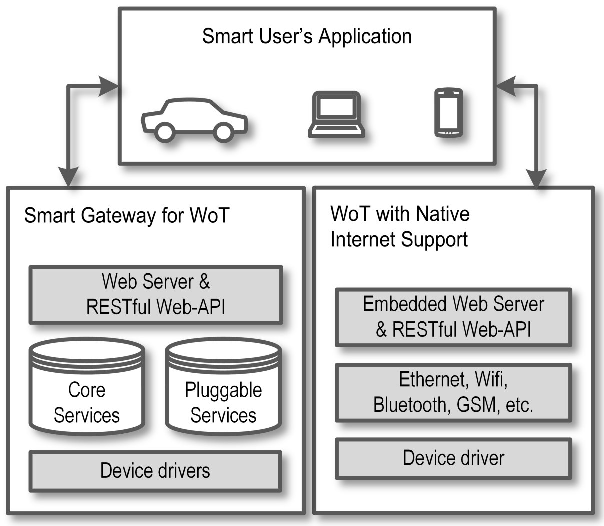
Smart User's Application
Smart Gateway for WoT
Web Server & RESTful Web-API
Core Services
Pluggable Services
Device drivers
WoT with Native Internet Support
Embedded Web Server & RESTful Web-API
Ethernet, Wifi, Bluetooth, GSM, etc.
Device driver

Figure 2: Smart WoT architecture

1. Device Drivers to communicate with low-level things,
2. Core Services for Web-APIs creation,
3. Pluggable Services to offer additional functionality and a Web server.
4. Application layer for user's interactions

2.4 Web Service Recommender System

Recommender systems are adopted to ease Web services discovery by information filtering before suggesting service items according to users' interests, past experiences, and recommendations. Web Service Recommender Systems (WSRS) can help actors (clients and applications) to discover the service that meets specified needs in the set of accessible Web services. So, the main task of WSRS is to recommend the optimal Web service, which best meets the requirements, and when two or more Web services have the same functionalities but different QoS performance. Web service technologies create an environment where actors can search and compose services automatically and seamlessly.

A short survey about the WSRS is presented in [27]. The authors classified WSRS into three categories:

1. Content-based,
2. Collaborative filtering,
3. Hybrid

The *Content-based* recommendation is performed by analyzing the similarity between artifacts (items) and the user's profile (set of interest). The more similar items to the user's profile are suggested first. The similarity identification process depends on the specified parameters in the user request.

The *Collaborative Filtering* (CF) is presented as the base method for the identification of similar user needs and recommends related Web services. A CF algorithm starts a process of filtering based on the association of a group of similar users with a group of similar services. A user-service matrix is introduced to evaluate the QoS retained by users.

The *Hybrid* Web Service Recommender Systems (HWSRS) proposes a combination of two methods (i.e., content-based and collaborative filtering recommendations) to address the limitations of each recommendation type. As noted by authors, it arises many approaches to combine WSRS techniques, but a complex study does not exist that states which combination approach is more dominant than another. They conclude that the existing studies show that hybrid WSRS provides statistically better results.

On the other hand, authors in [29] note the insufficiency of the semantics and syntax approaches to discover a service that best suits the user's needs. The reviewed recommender approaches are developed essentially for service discovery in centralized registries.

In this work, we argue for an HWSRS approach, and we aim at the discovery in a distributed registry environment based on user ratings and a set of similarity factors, presented in a multigraph structure discussed below.

3 Contributions

In this work, we analyze how IoT's intrinsic features impact the proposed Web service discovery and composition process in the context of existing Web service technologies. We note that there is a trend of wrapping anything (like IoT) into services and then, to adopt existing service technologies for discovery, composition, etc. Our approach is different in three elements.

1. A Web-based unified communication protocol;
2. A process of discovering and composing services based on a multigraph semantics;
3. A "*light*" technical solution developed that takes advantage of Web technologies.

In this paper, we consider the similarity between connected objects and linked Web services, and we argue for the use of Web services discovery methods in the IoT context, where the interoperability raises problems with the description of connected objects, as well as that of services.

Even if all the suppliers offer an API for each connected object, it's obvious that for comparable objects (e.g., two smarts LED bulbs of distinct brands), APIs can have disparities. To overcome this, one possible solution is to abstract the interface of the service by adding a semantic aspect to their descriptions. This can allow access to the functions, not more by the name, but by the description carrying meaning on the name. So, services can be interpreted automatically without having to consider the model or the brand of the object. Based on this context, we exploit the semantic descriptions of the object to discover an appropriate service. Our semantic approach consists of a graph of connected objects. Relationships are a function of several parameters: similarity between semantic descriptions of objects, geographical location, composition (common uses of objects), etc.

In this paper, we also propose a Web services-based communication approach for an interoperable and distributed IoT ecosystem. Our approach enables seamless interaction between homogeneous things that speak similar languages as well as heterogeneous things that speak different communication languages. Such an approach takes account of the distributed and dynamic nature of IoT and associated resources. Therefore, this is merely to reuse the existing network infrastructure with the associated popular protocols instead of creating yet another one. We argue the use of Web technologies for their interoperability, scalability, interactivity, and distributed applications.

Web technologies make simpler the system and application integration when a service-based approach is associated with connecting heterogeneous devices. As an experimental platform, we propose a RESTful Web Services framework animated by a graph-based approach for dynamic IoT/WoT services discovery in users' spaces.

As a complement of contribution, from reviewed related works, two aspects can be highlighted :

1. The semantic annotation of sensor data,
2. The smart objects' functionalities annotation.

In [23], a concise paper, we presented further defining the issues involved and setting out some proposed functional and architectural principles around which the Web services-based recommender system might coalesce.

In the evolution of the presented concept, we analyze IoT's semantic. We denote that semantic annotation concerns either data or functionalities. So we argue that it is more useful to have both annotations together to perform complex remote operations. For instance, to switch on/off a light bulb and to activate an air conditioning system. To address this concern, we opt for a *light service-based platform* (LSP) that can bring annotations together to perform more complex operations. To annotate IoT's data and functionalities semantically in a unified syntax, we apply a JSON-LD serialization format. This data format presentation is exploited natively by RESTful Web services from any NoSQL database.

To better define the scope of our research work, we introduce a simplistic functional WoT architecture with four layers, presented in the previous section, by adopting the SGL framework to manage RESTful Web Services, as depicted in Figure 2. The user's application addresses a smart device (e.g., smarts LED Lamps, air-conditioning system, smarts speakers) in the function of embedded Web-services characteristics correlated to users' needs or profiles. With this functional architecture, we argue that any connected device (Thing) can be accessed through Web technology and so be managed (data and parameters) via a RESTful type service, either by a smart gateway or by a native WS support.

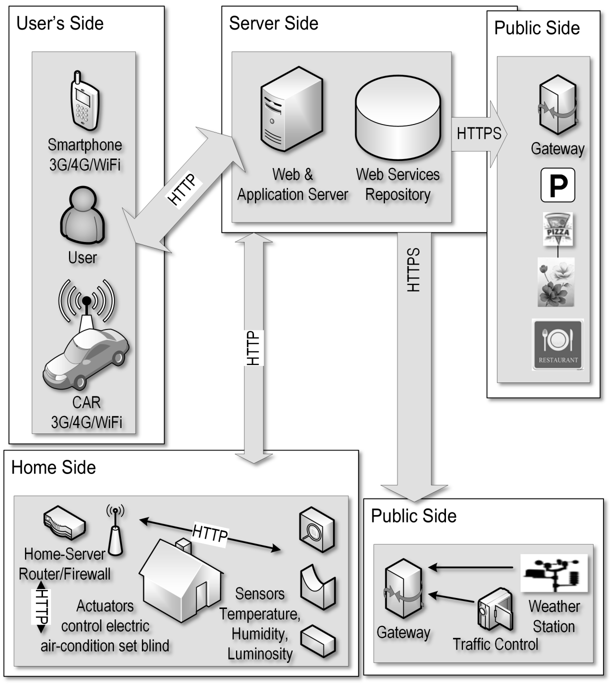

Figure 3: WoT use case scenario managed by RESTful Web Services

4 IoT Ecosystem Data Management

In [40], the authors suggest a scenario for IoT/WoT device-management that requires the discovery of appropriate Web services. They consider a smart environment in an interconnected IoT world. So, when a user departs from work, his car can provide information about his geolocation. The integrated GPS service can verify, using traffic data, that the destination is home and the arrival time can be estimated. Using this data, heating and air conditioning systems can be activated to obtain a comfortable temperature in the house. Meanwhile, along the way, the user can quest for the nearest parking to have the facility to buy flowers, or to book a table in a restaurant nearby, or to order pizzas to be delivered at home. On its way, the traffic conditions can change at any time. This context suggests changes: (1) in estimated arrival on the destination, and (2) changes in real-time for some parameters for the concerned IoT units. In this case, the process of Web services discovery and proposal should be adjusted to the new required IoT parameters and functionalities. This extended use case scenario, shown in Figure 3, can be easily fitted into a RESTful Web Service-based Information System (ReWSIS).

The set of generated data from IoT opens interaction with Databases and Information Systems (IS). Nevertheless, the emergence of IoT within IS leads to multiple challenges. Among these challenges, we can find the management of the resulting data and their integration with existing IS. To integrate our extended scenario in an IS, we propose a system based on a graph of services and parameters of connected objects (IoT profiles). A query is sent by the system to determine the services that can meet the needs of a connected object. We argue that an approach of personalization can build profiles of connected objects which can evolve in time as the technologies evolve. The profiling improves the process to recommend services by adapting itself to the environmental and technological changes.

4.1 The Graph Components

The use case scenario presented in Figure 3, although it is not entirely new, fits into a secured framework based on Web services with the scalability and data privacy core in the IoT context. A graph-based approach is applied to confirm the extended scenario, presented schematically in Figure 4, where a device connected to a sensor produces data managed by a service. This network is modeled as a heterogeneous multigraph based on users/services, sensors, devices, and mashups relationships. The multigraph, presented in Figure 5, is organized into five levels according to the different

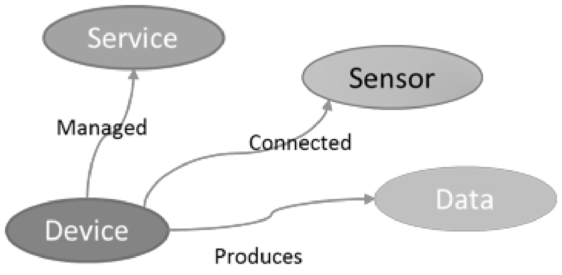

Figure 4: Theoretical approach

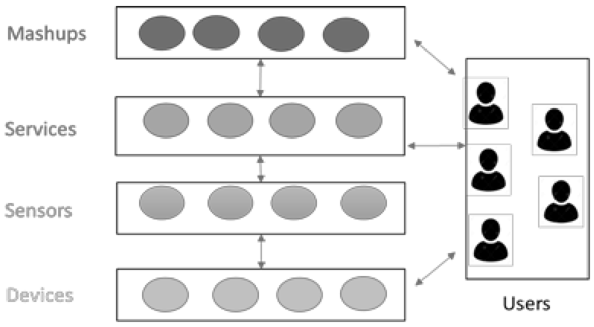

Figure 5: Model for heterogeneous multigraph

nodes:

1. Devices level,
2. Sensors level,
3. Services level,
4. Mashups level,
5. Users levels

We introduce below our definition for a multigraph and the relationships with services and mashups.

Definition 1. An IoT multigraph MG=<U,S,M,D,E> is an directed heterogeneous multigraph where:
U = $u_1, u_2, \ldots, u_N$ is a set of users (vertices),
S = $s_1, s_2, \ldots, s_M$ is a set of services (vertices),
M = $m_1, m_2, \ldots, m_L$ is a set of mashups,
D = $d_1, d_2, \ldots, d_L$ is a set of devices,
and:
E = (u_i, u_j) | user u_i is similar to a user u_j
$\cup$ (c_i, s_k) | service s_k belongs to a category c_i
$\cup$ (u_i, s_k) | user u_i uses a service s_k
$\cup$ (s_k, s_l) | service s_k is similar to service s_l
$\cup$ (s_k, m_l) | service s_k belongs to a mashup m_l
$\cup$ (u_i, m_k) | user u_i tracks a mashup m_k
$\cup$ (c_i, m_k) | mashup m_k belongs to a category c_i
$\cup$ (m_k, m_l) | mashup m_k is similar to a mashup m_l

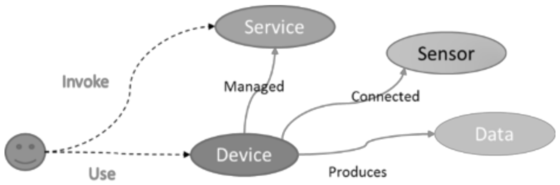

Figure 6: The user/device/service interaction

are a set of edges that materialize oriented relations between vertices

Services Relationships. In the real world, several services could be equivalent. A relationship can be defined between each pair of functionally similar services [29]. By comparing service items (e.g., *URI, name, description, and tags*) the similarity can be established.

Mashups Relationships. We also consider the mashup component. A mashup [32] in IoT/WoT is an application that retrieves information from multiple devices and uses several Web services related to these *things* (devices). In our model, we connect each object to the mashups that use it for several reasons. In the first step, it allows us to have information about the objects that can be used together. In a second step, we will have a way to recommend to the user things or mashups. So, two mashups m_1 and m_2 are connected if they share one service. The similarity is measured according to the number of common services.

Sensor-Sensor Relationships: For example, sensors related to identical things (IoT/WoT) or used together are connected. If two things t_1 and t_2 are managing two sensors s_1 and s_2 and t_1 and t_2 was used together, a relation is created between s_1 and s_2. The localization of things (sensor devices) is taken into account.

Device-Device Relationships: Things used together or that manage the same sensor and/or service are connected.

User-User Relationships: We also consider similarity relationships between users. The similarity relationship between users is determined if two users used the same services or device or invoked the same service to manage a sensor or a smart device. A relationship is then referred to as a similar relationship between these two users.

Figure 6 showed the user's interaction with smart devices and services. The number of services and connected objects that users have in common determines the users' similarity relationship. Users with similar requests could be considered as similar in that addressing several common services. The similarity between two users u_i and u_j is expressed using the following function [31].

$$Sim(u_i, u_j) = \frac{\mid H_{ui} \cap H_{uj} \mid}{\mid H_{ui} \mid} \quad (1)$$

where H_{ui} and H_{uj} are the recent histories of users u_i and u_j respectively.

4.2 Graph-based Approach for Service and Mashup Recommender

If all WoTs are abstracted as resources, then they become addressable, searchable, and accessible via the Web family protocols. The developer can use a uniform Web standard (API) to integrate all the abstracted resources needed as well as existing virtual Web service (e.g., Google Map) to create a *mashup* [39], [33].

Our objective is to perform online service discovery and recommendations when the user expresses a request. To achieve this, the graph-based system is queried to determine services that can touch a connected object. We leverage the multigraph by exploring *users/similarity* relationships and to come up with a set of Web services (or *mashups*) that would satisfy the requester needs (based on his preferences and/or the object profile stored in the services repository).

For a query, the search process consists of retrieving the most relevant services or mashups to manage the smart device that interacts with sensor data. The process returns the top *k-similar* relevant Web services. The query is keyword-based and may include one (or more) service name, tags, and protocol. This method is scalable but challenged by accuracy and timeliness.

The graph-based discovery process is introduced in Figure 7 to exploit interconnections between users, services, and smart devices. The discovery process selects services on the user's query from the service layer (Figure 5). A filtering step is applied to keep only services associated with the request. The queried device functionality is compared to the services' description. The most similar services to this query are selected and returned to the user. If no atomic services[19] could satisfy the required needs, a mashup is proposed to the user. As mentioned previously, mashups are applications/services created by composing various original services from disparate or even competing providers. For involved smart devices, a set of mashups and services are proposed to the user depend on similarity relations in the WoT graph. A list of mashups is proposed based on the required functionalities in the user's query. When looking just for a service, the list of mashups containing this service is recommended.

So, the discovery engine analyzes users request based on things that the current user uses and then, select the service corresponding the most to the query. A set of similar and complementary services is recommended in the function of services relationships in the graph. If a user needs change, the system can dynamically adjust and propose services or mashups based on the new submitted query.

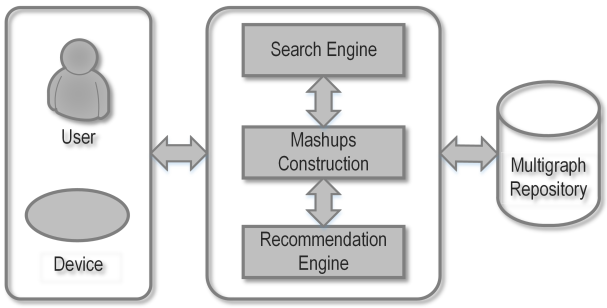

Figure 7: Schema for graph-based service composition

Algorithm 1 For *k-similar* services

```
Input: $S^M$ // Requested service
Output: $S^s$ or $M^s$ // Suggested service or mashup
1: while ($i \leq S^M.N$) do
2:     Insert $S^M.s_i$ to SetMG //MultiGraph
3:     while ($k \leq S^s.N$) do
4:         if ($S^s.s_k = S^M.s_i$) then
5:             return $S^s.s_k$
6:         else
7:             $k \leftarrow k + 1$
8:         end if
9:     end while
10:    while ($l \leq M^s.L$) do
11:        if ($M^s.s_l = M^L.s_i$) then
12:            return $M^s.s_l$
13:        else
14:            $l \leftarrow l + 1$
15:        end if
16:    end while
17:    $i \leftarrow i + 1$
18: end while
```

The matching[20] process is formalized in *Algorithm 1* as illustration of the *Definition 1*. Let S^M be the Web service that is requested, and S^s be the Web service that is recommended. If no matching service occurred, a M^s *mashup* as a similar service is proposed.

The dynamic mashups adaptation to satisfy user's needs according to context changes may involve many services. The service composition process involves

[19] The Atomic Service provides very specific functionality for raw data

[20] The problem of finding a similarity in a graph

selection and coordination. In an IoT context, let us return to the scenario of someone who launched a process of a set of services (GPS, turn on the air conditioning, etc.). In his way, the person decides to go and take dinner outside. So the first launched process becomes obsolete. In this case, an adaptation (service modification) is needed. To do so, we proceed to generate a new mashup based on extracted information from sensors/user's request. The newest discovery process is performed to find out services that can satisfy the user's request. According to discovered services, the mashups are modified. The modification consists of the first level, adds new services to the initial mashup, and then reinvokes already existing service with new user input parameters. This process is performed to discover the best services according to the user's previous uses, as depicted in Figure 7, while Listing 1 shows a graph as an output in adapted format.

5 IoT System Architecture for REST Style Web Services

In this section, the recommender system architecture is discussed, the architecture analytics, the data store model, and the technical contributions of this paper are also presented.

5.1 The WoT System Architecture

Figure 8 illustrates the different layers for accessing, discovering, sharing, and composing resources as depicted in [8]. Besides, we compare WoT functional layers and IoT models with the ISO model to create a better understanding of the functional model of Web services interaction.

- The *Thing layer* manages hardware and data processing of connected objects to upper layers.
- The *Accessing layer* transforms WoT/IoT into programmable devices (smart devices) as it ensures to have a Web API that exposes services through a RESTful API either directly or through a gateway.
- The *Discovering layer* provides a way to locate WoT/IoT based on his semantic description and/or the associated services.
- The *Sharing layer* ensures that data generated by WoT/IoT can be shared efficiently and securely across services.
- The *Composing layer* integrates services and data from devices (things) into more complex Web tools such as analytics software, mashup applications, and virtual Web services.

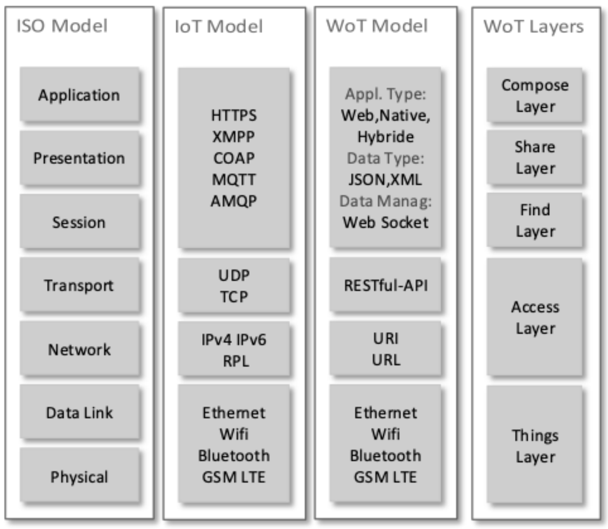

Figure 8: WoT model and layers vs IoT model and ISO model

According to the presented model, data and WoT parameters are accessible via a REST approach. REST is a light architectural style that uses simple HTTP commands (GET, POST, PUT, and DELETE), then makes it a particularly attractive candidate to interact with the Web of Objects [20]. In [25], authors cite REST as a preferred candidate to build APIs of connected objects. In this way, the connected objects obtain URIs and can then be exchanged and referenced according to their features. So, RESTful based services can manipulate resources in the *Thing Layer*, which can include physical objects (e.g., temperature sensors, light sensors, smart lamps), abstract concepts (e.g., collection of objects), dynamic and transient concepts (e.g., state of objects).

5.2 Database Architecture for Recommender System

Structured data are already in a required form for a relational database and are stored as is it. Relational databases can also manage semi-structured and unstructured data, but data-normalization and compromises are required to achieve efficient storage.

NoSQL databases represent an alternative to relational databases where rigid schemes and many other limitations concerning raw (no structured) data are avoided. The NoSQL databases support horizontal scalability, and managing them is simpler where the relational databases perform the vertical scalability with the complexity that this approach entails.

The data management model for our recommender system is based on a multigraph, and usually, this could suggest the involvement of a graph database. The key concept in a graph database is the data representation in the store as a collection of nodes and edges in order to customize the relationships between nodes. Other NoSQL storage mechanisms use a key-value store (i.e., dictionary or hash table) or document-oriented database (e.g., BSON[21], JSON, XML) to represent the graph data model.

In [16] NoSQL databases are categorized into four major classifications, which are:

1. Key-value stores (e.g., ArangoDB[22], also supports graph database model),
2. Graph databases (e.g., Neo4j[23]),
3. Wide column stores (e.g., Cassandra[24] certain similarity with MongoDB),
4. Document stores (e.g., MongoDB),

A survey over the published benchmark[25] denotes MongoDB as better performances than Neo4J in aggregation over a single collection, memory consumption over tests, neighbors with profile data, and fewer performances in single read and write data.

While authors in [32] used Neo4j to store a multigraph, in this work, we opt for the MongoDB database platform where the multi-level graph is represented in JSON format to manipulate and modeling the Web services ecosystem.

An analytics comparison between these two approaches is presented below.

Neo4j is a native graph database with several utilities to provide data visualization (e.g., Web browser tool) and other features such as the relationships between the data models and graphs. Neo4j offers the query language Cypher. It is much similar to structured query language for performing database operations. SQL database operations such as INSERT, CREATE, SELECT, UPDATE, SET, DELETE, ORDER BY, SKIP, LIMIT, MATCH, and WHERE clause can be applied. Besides, Neo4j supports importing data from external sources. It requires the Java development kit (JDK) to run Neo4j.

MongoDB is a data store collection, especially well-suited for JSON-style documents. A collection of MongoDB documents can have varying sets of fields, with different types for each field, including objects. MongoDB allows graph manipulation faster comparing to Neo4j. Besides, it offers navigation between the graph nodes to ensure a global view of the graph. MongoDB supports dynamic queries without requiring the database format to be fixed in a schema before running the query. MongoDB indexes the documents to speed up the query extraction time. This accelerates process execution time on the database.

Neo4j is a graph-based NoSQL database, while MongoDB, although it is also NoSQL, is a schemaless database with a holistic view of the data. In opposition to Neo4j, MongoDB does not create relationships between the data collections. Each data set stored in the database is disaggregated and independent as a native data collection. While Neo4J enables navigation through the graphs as a tree, MongoDB cannot provide visualization of the collection stores as graphs. Optionally, the use of Postman[26] API as a REST client allows the fastest interrogation of the graph based on a “GET” request, which is an advantage of using JSON format for graph representation compared to Neo4j.

As we are using a multigraph-based schema for our recommender system, it would make sense to integrate associated data in a graph database. Nevertheless, such a choice risks moving away from light and portable technical solution that takes advantage of Web technologies. So, we opt for an LSP that can bring together more complex operations.

To conclude this section and to support the use case presented in this work, we argue that IoT/WoT is a typical application scenario for using the MongoDB database. In opposition to relational databases (RDB), data in MongoDB are organized in documents (equivalent to rows in an RDB) with fields (as a column in an RDB) that are grouped into collections (equivalent to tables in an RDB). As a document-oriented database, MongoDB is well-adapted to Web-based applications (e.g., Node.js, JavaScript-based scripting, RESTful Web Services). The JSON format is using for document storage with the possibility to extend the data implementation with additional types (e.g., arrays) and a JSON-LD schema.

6 Technical Contribution and Test

In this section, we follow the REST-based system architecture. The system presents a uniform interface and interactions with WoT built around universally supported methods. As shown in Figure 9, the system queries the graph to determine the services that can meet the parameters of a connected object.

[21] Binary JavaScript Object Notation
[22] https://www.arangodb.com/
[23] https://neo4j.com/blog/iot-graphs-business- requirements
[24] https://cassandra.apache.org/
[25] https://www.arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/
[26] https://www.getpostman.com/

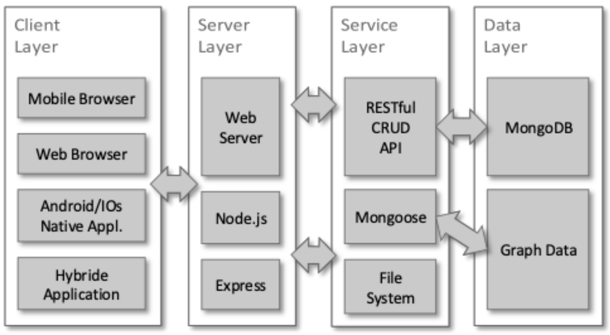

Figure 9: System layers for the technical solution

6.1 Test

For proof of our graph-based service discovery concept, according to the scenario depicted in the previous chapter, we conduct a test. The test uses a real dataset built on RESTful CRUD (i.e., CREATE, RETRIEVE, UPDATE, DELETE) API with the Node.js[27] platform, Express.js[28] framework, and MongoDB server both assisted by Mongoose[29] framework for better data integration. The level of system and data integration and the functionalities of concerned components are as follows:

- *Node.js* is a server-side JavaScript framework for building cross-platform applications. This server platform is composed of an event-driven architecture that is used for asynchronous (non-blocking) I/O operations.
  We build the tested RESTful API in front of Node.js and Express.js.

- *Express.js* is a Web Application Framework with an HTTP module built on top of the Node.js server.
  To organize our lightweight Web application into an MVC[30] architecture, we set up the body-parser rules to manage the incoming requests dependency destined to the RESTful API.

- *MongoDB* is a NoSQL database that can store graph data. A graph is a data structure that consists of a set of nodes and a set of edges that relate nodes to one another. A MongoDB record is a document where the data structure is similar to JSON objects. To represent a graph in this database, links and nodes are stored as objects. To do this, we bring Mongoose as an object-relational mapping library.

```
{ "@WoT": [
    { "@id": "devicemanagement",
      "@type": "service",
      "tags" :"lampe , light",
      "URL": https://www.sevices.net },
    { "@id": "devicespeaker",
       "@type": "device",
       "description": "smart speaker" },
    { "@id": "sensorlampe",
       "@type": "sensor",
       "description": "lamp managing" },
    { "edges": [
            { "source": "lampemanagement",
               "target": "sensorlampe",
               "relation": "manage" },
            {  "source": "sensorspeaker,
                "target": "devicespeaker",
                "relation": "manage"  },
            {  "source": "LED Lamp,
                "target": "lampemanagement",
                "relation": "use" }
            ]
     }
     { "user":[ { "@type": "user",
                   "name": "Michel Blanc",
                   "jobTitle": "devlopper"} ]
        }
}
```

Listing 1: Fragment of WS collection in JSON-LD format

- *Mongoose* is an Object Document Mapping (ODM) tool for Node.js and MongoDB and helps to convert the objects in the code to documents in the database. They are often used together because of their shared use of JSON format. A fragment of our tested collection is présented in Listing 1 in adapted JSON-LD form.

The discussed in this work infrastructure is designed for building scalable RESTful network applications. With Node.js, it is also relatively simple to set up an API, based on a server running with a few lines of code, that natively returns data in JSON format.

7 EXPERIMENTATION

In this section, we present a preliminary evaluation of our multigraph-based recommender approach. We discuss the experimentally obtained results following the implementation of the recommender system. For proof of our graph-based service discovery concept, a test collection has been used to assess a set of recommendations, i.e., services and mashups.

It should be noted that as a result of the significant disruption that is being caused by the COVID-19 pandemic, we were not able to conduct a full-scale evaluation. However, all components of the

[27] https://nodejs.org/
[28] https://expressjs.com/
[29] http://mongoosejs.com/
[30] Model_View_Controller

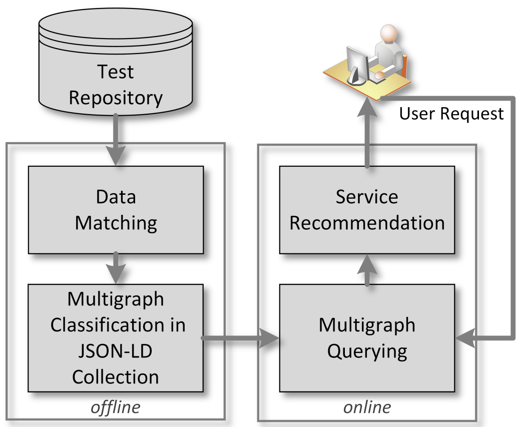

Figure 10: Multigraph workflow

recommender system have been tested and validated with external public resources. For evaluation, we crawled a set of services, devices, and mashups from a test collection derived from SWTC[31]. "*This project contains shared resources used for testing semantic web services technologies. These test resources are a collection of pre-existing and derived test collections produced by a number of researchers*".

7.1 Implementation

The used test collection results from 18 queries from different domains, i.e., communication, food, economy, medical, travel and education. The requests involved 586 services and 28 ontologies used to Web services semantically annotation. To evaluate the recommendation precision a relevant dataset is provided for each query. We built on MongoDB a graph by creating a JSON-LD collection (Listing 1) by exporting the set of recommended results. The workflow of the multigraph modeling for the test is presented in Figure 10.

To reduce the test complexity, we first selected offline data randomly from the external test repository. In this manner, we ensure that the results are independent of the input dataset. The extraction process collects data on users and services that are identified on watchlists (user history). The mapping module determines the similarities between users and services. The graph building module creates a node for each: category, service, and user. Then it proceeds to the creation of the relationships between the nodes, i.e., between users and services; and similarity between services. The classification process groups the similar services. The recommendation process works online and comports two modules: (1) a graph query module and (2) a recommendation module. The query module identifies candidate services in the graph collection. A list of resulting services is passed to the recommender module. During the recommendation process, the module sorts and enriches the results of this list.

[31] https://github.com/kmi/sws-test-collections

7.2 Evaluation

The goal of the experimentation is to evaluate user satisfaction with recommended services and mashups based on MongoDB database exploration. As noted previously, the recommender system performs graph analytics to produce a set of recommended services and mashups according to a user request, which consists of a set of keywords. The similarity between the user query and services components is evaluated to discover the most similar services or mashups. The recommender system returns a set of ranked services or mashups that are supposed to satisfy the user's needs. The quality of the recommendations is evaluated by the built-in MongoDB JSON-LD graph that is browsed for a set of services, smart-devices, and mashups. A fragment of the JSON-LD WoT/user collection is shown in Listing 1. The involved algorithm is carried out ten times with random inputs. Experiments have been conducted on a Rasberry PI 4 model B, with 8G RAM, under Raspberry Pi OS (Raspbian).

7.3 Quality of Recommendation

In essence, recommender systems filtering selective data extracted from large amounts of dynamically generated information related to user's preferences, interests, items, or events. We used an implementation [30] of the Apriori algorithm [1] for NoSQL databases to determine the most common itemsets within the used test dataset. Two parameters are to be defined: (1) *support* that refers to items' frequency of occurrence, and (2) *confidence* to express the conditional probability. Following tests conducted on 100 users with 20 services in their watchlist, we set the two parameters respectively to 0.5 and 0.9.

This data mining technique is part of a collaborative filtering technique that is used to calculate the similarity between users by comparing their relationships (ratings) with services and to compute a recommendation item. Our approach is based on a hybrid technique, i.e., (1) memory-based collaborative filtering to compute the similarity between users and services and (2) a model-based algorithm for data mining.

To evaluate the quality/performance of the recommender system (QoRS), three measures were inspected: (1) Precision; (2) Recall; (3) RMSE measures.

1. *Precision* refers to the ratio of correctly predicted services to the number of all recommended services.

2. *Recall* refers to the ratio of correctly predicted services to the number of all the services in the testing set.

3. *RMSE* refers to the Root Mean Squared Error that is used in evaluating the accuracy of predicted rating (r_s).
   In [38], authors testify that the RMSE measure is widely used in service recommendation approaches. It corresponds to the mean absolute error between the actual opinion and the user's prediction about a service.
   For each service $(s \in L)$, we define the prediction rating (r_s) that represents the prediction satisfaction.
   Thus, we consider : $r_s = \{(s \in P \ ?\ 1; 0)\}$

Let us introduce PR as a set of relevant recommended services, R as a set of recommended services, and P as a set of relevant services.

$$Precision = \frac{\mid PR \mid}{\mid R \mid}, Recall = \frac{\mid PR \mid}{\mid P \mid} \qquad (2)$$

$$RMSE = \sqrt{\frac{\sum_{u,s}(r - r_s)}{N}} \qquad (3)$$

Where N is the number of recommended services and r is the real evaluation.

The first resulting recommended services (mashups) are used to compute precision and recall. Figure 11 shows the five and the ten first resulting recommender services and mashups that are used to compute the Recall, Precision, and RMSE. It is obvious that the accuracy of the recommendations increases proportionally with the number of returned services and mashups. In this case of use, the probability of providing a relevant service increases as the number of recommended services increases too.

Unlike existing recommender approaches (e.g., TrustSVD [15]), we also tested user-service and user-mashup relationships, and better performances are observed even when users have an incomplete watchlist. The support for user-profiles and backgrounds are assets in our recommender system compared to existing service recommendation systems.

Despite the lack of a benchmarking system, the experimental results are promising, even the lack of relevant touchstone.

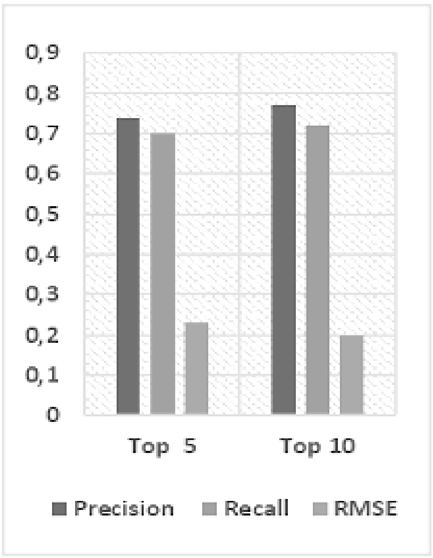

Figure 11: Evaluation results

8 Conclusions

In this paper, we consider the Internet of Things as an emerging research domain. The main contribution of the presented research work is a single interoperable framework to overcome the heterogeneity of proprietary technologies to enable the data and application (device) integration for IoT devices. To achieve this objective, we consider the similarity between connected objects and linked Web services. We use Web services discovery methods in the IoT context, where the interoperability raises problems with the description of connected objects, as well as that of services.

In this paper, we also argue that Web technologies offer a suitable environment to access IoT (WoT) data and parameters via Web service-based applications. In this way, users profit from well-known Web mechanisms to interact and share IoT by managing them with popular Web languages, platforms, and services. We propose a system based on graphs of services and parameters of connected objects associated with a REST system architecture for Web of Things (WoT) management. The graph-based approach for Web service discovery, recommendation, and composition is tested in real life. The validation process is based on analyzing the correlations of IoT usage frequency and the satisfaction of users' needs. This factor, among others, reflects the scalability of the presented recommender system concerning work contributions. The discussed system-model fits easily into the new computing paradigm, i.e., Dew Computing (DC). The scalability of the DC [36] model integrates a large number of heterogeneous devices and different types of equipment [37], and this corresponds to the presented in this paper use case.

The analysis of the discussed in Section 3 case of use pushes us towards the idea in future work to perform an adapted machine learning approach.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, p. 487–499.

[2] A. M. Alkalbani, W. Hussain, and J. Y. Kim, "A centralised cloud services repository (ccsr) framework for optimal cloud service advertisement discovery from heterogenous web portals," *IEEE Access*, vol. 7, pp. 128 213–128 223, 2019.

[3] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[4] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "Iot-lite: A lightweight semantic model for the internet of things," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, 2016, pp. 90–97.

[5] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "Iot-lite ontology," http://www.w3.org/TR/soap12/, November 2015.

[6] R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana, "Web service description language," http://www.w3.org/TR/wsdl20, January 2007.

[7] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. Le Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor, "The ssn ontology of the w3c semantic sensor network incubator group," *Journal of Web Semantics*, vol. 17, pp. 25–32, 2012.

[8] V. T. Dominique Guinard, *Building the Web of Things: With examples in Node.js and Raspberry Pi*. Manning, 2016.

[9] Y. Elshater, K. Elgazzar, and P. Martin, "Web service discovery for large scale iot deployments," *Services Transactions on Services Computing*, vol. 4, pp. 55–68, 01 2016.

[10] M. Fang, D. Wang, Z. Mi, and M. Obaidat, "Web service discovery utilizing logical reasoning and semantic similarity," *International Journal of Communication Systems*, vol. 31, 11 2017.

[11] A. Gerber and J. Romeo, "Connecting all the things in the internet of things : A guide to selecting network technologies to solve your iot networking challenges," https://developer.ibm.com/technologies/iot/articles/iot-lp101-connectivity-network-protocols/, 2017.

[12] P. Gomes, E. Cavalcante, T. Rodrigues, T. Batista, F. C. Delicato, and P. F. Pires, "A federated discovery service for the internet of things," in *Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT*, ser. M4IoT 2015. New York, NY, USA: Association for Computing Machinery, 2015, p. 25–30.

[13] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *2010 Internet of Things (IOT)*, 2010, pp. 1–8.

[14] D. Guinard, "A web of things application architecture – integrating the real-world into the web," Ph.D. dissertation, ETH, Zurich, 7 12011.

[15] G. Guo, J. Zhang, and N. Yorke-Smith, "Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 123–129, 01 2015.

[16] A. Gupta, S. Tyagi, N. Panwar, S. Sachdeva, and U. Saxena, "Nosql databases: Critical analysis and comparison," in *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, 2017, pp. 293–299.

[17] S. N. Han and N. Crespi, "Semantic service provisioning for smart objects: Integrating iot applications into the web," *Future Generation Computer Systems*, vol. 76, pp. 180–197, 2017.

[18] C. A. Henson, J. K. Pschorr, A. P. Sheth, and K. Thirunarayan, "Semsos: Semantic sensor observation service," in *2009 International Symposium on Collaborative Technologies and Systems*, 2009, pp. 44–53.

[19] D. Hussein, S. Han, G. M. Lee, N. Crespi, and E. Bertin, "Towards a dynamic discovery of smart services in the social internet of things," *Computers and Electrical Engineering, Elsevier*, vol. 58, 01 2017.

[20] L. R. D. Jyoti L. Khachane, "Survey paper on web services in iot," *IJSR International Journal of Science and Research*, vol. 4, pp. 635–637, December 2013.

[21] A. E. Khaled and S. Helal, "Interoperable communication framework for bridging restful and topic-based communication in iot," *Future Generation Computer Systems*, vol. 92, pp. 628–643, 2019.

[22] M. Laine, "Restful web services for the internet of things," https://www.semanticscholar.org/paper/RESTful-Web-Services-for-the-Internet-of-Things-Laine/, 2011.

[23] I. Madjarov and F. Slaimi, "A Multigraph for RESTful Services Discovery in IoT Ecosystem," in *2019 IEEE World Congress on Services (SERVICES)*, Milan, Italy, Jul. 2019. [Online]. Available: https://hal-amu.archives-ouvertes.fr/hal-02192564

[24] F. Musyaffa, L. Halilaj, R. Siebes, F. Orlandi, and S. Auer, "Minimally invasive semantification of light weight service descriptions," in *IEEE International Conference on Web Services*, 06 2016, pp. 672–677.

[25] S. N. A. U. Nambi, C. Sarkar, R. V. Prasad, and A. Rahim, "A unified semantic knowledge base for iot," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 575–580.

[26] F. Paganelli, S. Turchi, and D. Giuli, "A web of things framework for restful applications and its experimentation in a smart city," *IEEE Systems Journal*, vol. 10, no. 4, pp. 1412–1423, 2016.

[27] S. M. C. Pallavi P. Gupta, "Web service recommender system with location and qos prediction framework," *International Journal for Research in Engineering Application and Management*, vol. 2, 03 2017.

[28] I. Pencheva Evelina, Atanasov, "Engineering of web services for internet of things applications," *Information Systems Frontiers*, vol. 18, pp. 277–292, 2016.

[29] M. Sellami, S. Tata, Z. Maamar, and B. Defude, "A recommender system for web services discovery in a distributed registry environment," in *2009 Fourth International Conference on Internet and Web Applications and Services*, 2009, pp. 418–423.

[30] S. Senhadji, I. Benzeguimi, and Z. Yagoub, "Association rules mining and nosql oriented document in big data," *International Journal of Computer and Information Engineering*, vol. 14, no. 12, pp. 494 – 499, 2020. [Online]. Available: https://publications.waset.org/vol/168

[31] F. Slaimi, S. Sellami, and O. Boucelma, "From a web services catalog to a linked ecosystem of services," in *Semantic Keyword-Based Search on Structured Data Sources*, J. Szymański and Y. Velegrakis, Eds. Cham: Springer International Publishing, 2018, pp. 86–98.

[32] F. Slaimi, S. Sellami, O. Boucelma, and A. Hassine, "A multigraph approach for web services recommendation," in *On the Move to Meaningful Internet Systems: OTM 2016 Conferences*, 10 2016, pp. 282–299.

[33] J. Sorg and R. Kunkel, "Conception and implementation of an ogc-compliant sensor observation service for a standardized access to raster data," *ISPRS International Journal of Geo-Information*, vol. 4, pp. 1076–1096, 07 2015.

[34] W3C, "Messaging framework (second edition)," http://www.w3.org/TR/soap12/, January 2007.

[35] W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner, "A comprehensive ontology for knowledge representation in the internet of things," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012, pp. 1793–1798.

[36] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[37] A. Whitmore, A. Agarwal, and L. Xu, "The internet of things—a survey of topics and trends," *Information Systems Frontiers*, vol. 17, 04 2014.

[38] L. Yao, Q. Z. Sheng, A. H. H. Ngu, and X. Li, "Things of interest recommendation by leveraging heterogeneous relations in the internet of things," *ACM Trans. Internet Technol.*, vol. 16, no. 2, Mar. 2016.

[39] J. yi Hong, E. ho Suh, and S.-J. Kim, "Context-aware systems: A literature review and classification," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.

[40] A. Zaslavsky and P. P. Jayaraman, "Discovery in the internet of things: The internet of things (ubiquity symposium)," *Ubiquity*, vol. 2015, no. October, Oct. 2015.

AUTHOR BIOGRAPHIES

Ivan MADJAROV is a senior lecture in Computer Science at Aix-Marseille University and a researcher at LIS (Laboratoire d'Informatique et Systemes) within the research group DIAMS (Data Integration, Analysis, and Management as Services). Previously, he worked as an assistant professor in the Department of Programming and Computer Systems Application at the Technical University of Sofia, Bulgaria. His research interests include service-oriented computing, data integration, multimedia document engineering, Internet of Things (IoT), and Cloud computing. He is conducting his research with application to domains of e-Learning and m-Learning systems and standards. He has been a leader or participant in several national and international projects.

Fatma SLAIMI received a Ph.D. degree in computer science from Aix-Marseille University, Marseille, France, within the research team DIMAG (Data, Information and content MAnagement Group) in LSIS (Laboratoire des sciences de l'information et des systèmes). His research thesis was based on Web Services discovery recommendations with graph databases application. She worked as a temporary lecturer and research assistant at the IUT-RT department.