CEIoT: A Framework for Interlinking Smart Things in the Internet of Things

Ali Shemshadi[1], Quan Z. Sheng[1], Yongrui Qin[2], and Ali Alzubaidi[3]

[1] School of Computer Science
The University of Adelaide, SA 5005, Australia
{ali.shemshadi, michael.sheng}@adelaide.edu.au
[2] School of Computing and Engineering, University of Huddersfield
yongrui.qin@hud.ac.uk
[3] College of Computer Science - Al Lith, Umm Al-Qura University
aakzubaidi@uqu.edu.sa

Abstract. In the emerging Internet of Things (IoT) environment, things are interconnected but not interlinked. Interlinking relevant things offers great opportunities to discover implicit relationships and enable potential interactions among things. To achieve this goal, implicit correlations between things need to be discovered. However, little work has been done on this important direction and the lack of correlation discovery has inevitably limited the power of interlinking things in IoT. With the rapidly growing number of things that are connected to the Internet, there are increasing needs for correlations formation and discovery so as to support interlinking relevant things together effectively. In this paper, we propose a novel approach based on Multi-Agent Systems (MAS) architecture to extract correlations between smart things. Our MAS system is able to identify correlations on demand due to the autonomous behaviors of object agents. Specifically, we introduce a novel open-sourced framework, namely CEIoT, to extract correlations in the context of IoT. Based on the attributes of things our IoT dataset, we identify three types of correlations in our system and propose a new approach to extract and represent the correlations between things. We implement our architecture using Java Agent Development Framework (JADE) and conduct experimental studies on both synthetic and real-world datasets. The results demonstrate that our approach can extract the correlations at a much higher speed than the naive pairwise computation method.

Keywords: Internet of Things, Correlation, Multi-Agent System

1 Introduction

With advances in the enabling technologies, such as Radio Frequency Identification (RFID), sensors, power harvest and IPv6, nowadays people can easily connect their everyday objects to the Internet. Thus, the paradigm of Internet of Things is a compelling and shifting vision of the future Internet. In the recent years, this paradigm has been tremendously growing. It is predicted that

Fig. 1. Example of Web mashup from ThingSpeak platform

by 2020, billions of devices will be connected to the Internet [7]. Even at the present time, numerous cloud based platforms are providing services for connecting and managing smart things. Taking advantage of the mashup paradigm, IoT data is commonly visualized and presented through simple Web mashups. For instance, Figure 1 shows an example of Web of Things mashup from the ThingSpeak platform [4]. As shown, none of the parts of the present mashup is referring to other mashups or correlated things on the Web. As a result, correlations remain implicit and IoT resources may remain isolated from each other if they are not interlinked. Interlinking relevant smart things will trigger improved user navigation as well as providing means for future IoT search engines. This is a very critical issue which resembles the role of hyperlinks in the Web.

A hyperlink is a reference to Web resources that the reader can directly follow by clicking on it. Usually, hyperlinks are associated with a textual description about their target, which is called hypertext. They play a key role in interlinking Web resources and provide navigation between different Web pages for users. Web crawlers are navigated using those hyperlinks in Web-based documents [3]. To enable the interlinking between resources in the context of interconnected networks of things, one eminent issue is that the traditional approach of interlinking Web documents, cannot fully unravel the benefits of interlinking IoT. Table 1 summarizes the differences between interlinking IoT resources vs. hyperlinks in the traditional Web.

Table 1. Requirements of traditional WWW hyperlinks vs. novel IoT-links

	Hyperlinks	**IoT-Links**
Establishment	manual and static	automatic and on-demand
Term	long-lasting and fixed	short term and highly dynamic
Connection Types	simple (single type)	various types
Weighted	No	Yes
Node Types	web pages	heterogeneous resources
Users	human users and crawlers	smart things, human users and crawlers

[4] https://thingspeak.com/channels/38629

Due to the highly dynamic and heterogeneous nature of IoT, correlations between entities may quickly become outdated due to the frequent changes in the status of things. Thus, one eminent issue is how to effectively and efficiently establish and maintain the links to the correlated resources. As the above Table shows, IoT-links have different requirements from hyperlinks in the traditional Web. Automatic maintenance of IoT-links requires a solid understanding of the implicit correlations between IoT resources in the physical world.

Although every pair of smart things around the world could potentially be correlated, in the context of IoT, correlations may not necessarily share the same type or the same weight.Thus, several types of correlations can be identified between interconnected things [2]. For instance, the type of the correlation that exists between two things that belong to the same person (`owned by` correlation) can be different from the correlation between two objects that are present in the same physical area (`co-located` correlation). Moreover, correlations of the same type, may not necessarily have the same weight. For example, the weight of co-located things may vary based on their distance from each other.

In this paper we propose the CEIoT (**C**orrelations **E**xtractor for **IoT**), a framework to facilitate automated correlation extraction for smart things in IoT. We use Multi-Agent System (MAS) architecture to design and implement our approach in order to be able to simulate the behaviors of smart things in real-world. We use MAS architecture mainly due to enable the solution to benefit from autonomous behaviors of the agents. Our framework regulates the extraction of different types of correlations. The correlation types that we cover in this paper are defined as follows [2]:

- Ownership object relationship (OOR): correlates objects with the same owner.
- Co-location object relationship (CLOR): correlates objects that are physically close to each other.
- Category based object relationship (CBOR): correlates objects which have the same describing tags.

We select the above correlation types mainly due to the nature and the structure of the IoT data that we have crawled using the ThingSeek crawler engine [11]. Each record in our dataset contains descriptive fields and sensor output fields. The sensor output in our dataset is includes the *data stream* field and the descriptive fields in our dataset include service description tags (which are defined by the owner and contain *tags*, *data stream unit*, *data stream symbol*, *type* and etc.), location (*latitude* and *longitude*) and owner (*user*). Due to diversity of the sensors in IoT and the lack of an standard ontology that correlates readings of various types of sensors, we cannot easily compare the values of sensor readings. For example, if sensor A reports the temperature as $12°C$ and sensor B reports it as $12°F$ sharing the same value 12 does not imply that their reading is similar. Thus, we opt out Sensor Reading Similarity [13] and rather focus on the descriptive fields.

In our CEIoT framework, we provide correlations with normalized weights to enable our model to represent more details from the real-world. Thus, our

approach employs a weighted undirected graph to model the heterogeneous network of things. In this paper, we assume that each thing is registered only to one network and belongs to one user only. We only focus on the publicly available things. We design a distributed and scalable framework to support the correlation extraction and use Open Linked Data to present the extracted correlations. Our contributions are summarized as follows:

- We propose our CEIoT framework to extract correlations in IoT. We support correlations with different types. We use a distributed architecture to enable our CEIoT framework to estimate the weights. To the best of our knowledge, existing approaches are limited in terms of the diversity and scale.
- We define the process of correlation discovery for IoT. We propose two novel algorithms for extracting and one algorithm for integrating the extracted correlations. In the CEIoT framework, we localize CBORs and estimate the weights of correlations for CLORs to increase the efficiency of the correlation extraction process. We increase the efficiency of correlation extraction and integration using a distributed architecture.
- We conduct experiments to evaluate our approach. We crawl an IoT platform to obtain the real-world data. We use both synthetic and real-world datasets to demonstrate the efficiency and effectiveness of our framework.

Our paper is organized as follows.

2 The CEIoT Approach

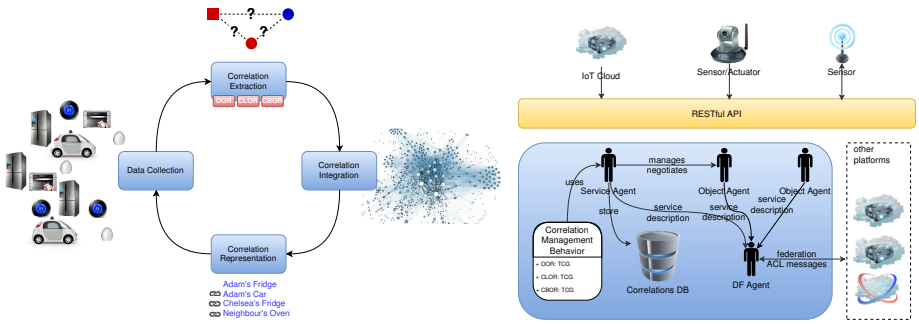
In this section, we present the details of our CEIoT architecture for automated extraction and representation of the correlations between things in IoT.

2.1 Correlation Discovery Process

Today, there are many online IoT platforms such as Xively[5] and Paraimpu[6]. Despite of their large scale and complexity, no means has been deployed to analyze or present the correlations between things. This includes the correlations of things of both inter and intra data sources.

We define correlation discovery as the process of extraction and representation of correlations of any types that exist between the resources in the IoT. Figure 2(a) illustrates the process of correlation discovery for the IoT consisting of four different phases: *Collection*, *Extraction*, *Integration* and *Presentation*. In the first phase, things' data is collected via RESTful application interfaces and maintained on a server. In the next step, *Extraction*, the similarity of given object pairs is examined based on different measures and criteria to extract the correlations. During the *Integration* phase, all of the extracted correlations are integrated to form a *Things Correlations Graph* (*TCG*) [12], which resembles the graphs in the traditional social networks. Finally, in the last phase, the edges of the TCG are converted into IoT hyperlinks.

[5] https://xively.com
[6] https://www.paraimpu.com



(a) (b)

Fig. 2. (a) Different phases of the correlation discovery process in IoT; and (b) The CEIoT framework

2.2 Framework Architecture and System Entities

Our CEIoT framework architecture is inspired by MAS framework. We design and implement a set of agent classes with built-in behaviors which facilitate the simulation of important entity types and their interactions in IoT correlation extraction problem. In the next step, agents are instantiated and deploy predesigned communication protocols to interact and submit/receive messages. An overview of the CEIoT framework is shown in Figure 2(b). Each platform in our framework operates independently from other platforms as IoT platforms operate in the real-world. The figure shows the main types of agents in each platform and how their instances interact with other parts of the system such as smart things, database and other platforms. Each platform maintains its own correlation database and Data Facilitator (DF) Service. As shown, agent classes include Service Agent, Object Agent and DF Agent. Due to the huge complexity of the nature of human user behaviors, we do not simulate them in our system and leave it for future works in this area. Existing agent classes and their roles are described in the following.

Object Agent. Object Agents are the main building block of the system; the smart things. These agents maintain the characteristics of the "things" that are connected to IoT and contain necessary behaviors to facilitate interconnections with other agents such as updating characteristics/readings and service registration. These agents constitute the largest number of agents in the system as each Object Agent is launched for one "thing" only. Each Object Agent models $A_i^o = (t, dt, lat, lon, u)$ such that $t \subset T$, $dt \in \mathbb{R}$, $lat \in \lfloor -90, 90 \rceil$ and $lon \in \lfloor -180, 180 \rceil$, $u \in U$ where T is the set of descriptive tags, dt is the latest datastream reading, u is the owner of the smart thing and lat, lon are the latitude and longitude of the object, respectively. Also, the Object Agent contains at least three default behaviors. One is to register their descriptive tags (t) into DF service. Two other behaviors are for updating location and other characteristics.

Service Agent. A Service Agent represents an IoT service provider (IoT platform) which facilitates the management of Object Agents. There is only one service agent per platform. It is responsible for coordinating and managing all agents present in its container as well as correlation discovery. These agents maintain all of the necessary information about their corresponding IoT platform. This includes host URL, port, Agent Communication Channel's address, platform ID and the DF Agent. Moreover, the Service Agent can launch, suspend or destroy Object Agents if required. The main responsibility of the Service Agent is to enquire other agents and update the correlation database frequently.

DF Agent. DF service facilitates the address book of each platform. Intra-platform agents can enquire the DF service to find agents with the specified services. The DF Agent stores tuples of services and agent URIs in the form of $\{(t, a)\}$ such that $t \in T$ and $a \in A$ where T is the set of descriptive tags and A is the set of all agents in the platform. Usually, DF service is provided only for the platform agents internally and is not designed to be shared across multiple platforms. Hence, inter-platform agent communication cannot be established in this situation. To avoid having a number of isolated MAS platforms, we devise a medium to share DF data across authorized platforms.

2.3 Correlation Extraction

Correlation extraction is the key step in the IoT correlation discovery process. In this step, our aim is to set up a efficient approach to extract the three types of correlations discussed earlier. As each type of correlation is discovered independently, firstly, we propose a separate approach for each correlation type. Then secondly, we investigate how we can integrate the process and the results.

CLOR. Given a pair of Object Agents (A_i^o, A_j^o) and a threshold $t \in (0, 1]$, CLOR can be defined as follows:

$$\texttt{clor}(A_i^o, A_j^o) = \begin{cases} \frac{\Delta(A_i^o.l, A_j^o.l)}{max\{\Delta(A^o.l, A^o.l)\}} & if\ \ \Delta(A_i^o.l, A_j^o.l) \geq t; \\ 0 \qquad otherwise \end{cases}$$

where $\Delta : (latitude, longitude)^2 \rightarrow R$ returns the distance between two points (Manhattan, Euclidean, Haversine and etc.).

A naive approach for extracting CLORs would require mutual comparison between every pair of things. Therefore, the complexity for extracting the CLORs for N things using a naive approach is $\mathbf{O}(N^2)$, which is not suitable for a large number of things as in IoT. We introduce a weight estimation strategy for CLORs. We use R-Tree data structure and capping the distance granularity to extract CLORs. For distance granularity limit, we consider the area of the parent rectangle (T) with a diagonal length $\texttt{D}(T)$ in which the distance of objects is bounded by $0 \leq d \leq \texttt{D}(T)$. Thus, if we limit the granularity of the distance to $\texttt{D}(T)$ (means that the distance can only be 0 or $\texttt{D}(T)$), then the CLOR between

Algorithm 1: EXTRACT_CLOR

input : Granularity level l, max level l_m, current sub-tree rectangle T, *global* adjacency matrix M, set of object agents A

if $|A^o \in T| \geq 2$ **then**
  $\forall (A_i^o, A_j^o) \in T : M(A_i^o, A_j^o) = M(A_i^o, A_j^o) + \frac{2^{l_m}}{\mathtt{D}(t)}$
  if $l \leq l_m$ **then**
    $T' = \mathtt{subtrees}(T)$
    foreach $t \in T'$ **do** **if** $|A^o \in t| \geq 2$ **then**
      call $\mathtt{EXTRACT\_CLOR}(l+1, l_m, M)$

all objects located in T form a complete graph $G_T = (V, E, w)$ where V is the set of objects, $E = V \times V$ is the set of edges defined between all objects in T and $w = \mathtt{D}(T)$. For a better precision of the weights w, we can divide the area T and strengthen the weights of edges in the same sub-areas. Thus, objects in the same sub-area $t \in T'$, will have a maximum distance of $\mathtt{D}(t)$ yielding a correlation which is $\frac{\mathtt{D}(T)}{\mathtt{D}(t)}$ stronger. For example, objects surrounded by a rectangle with $1km$ diagonal have a correlation twice stronger than objects surrounded by a larger rectangle with a $2km$ diagonal. Figure 3(a) depicts this idea.

Algorithm 1 describes our approach in further details. The `Extract_CLOR` is a recursive algorithm to estimate the strength of correlations between object agents. In the first level, the algorithm is launched with initial adjacency matrix $M = \{0\}_{m,m}$ where m is the number of object agents. In each recursion round (level l), the algorithm assigns correlation weights to all object agents included in the target area T. Thereafter, the algorithm would stop recursion for empty areas or if it reaches to the maximum level of granularity. The order of the algorithm would mainly depend on the distance granularity level rather than the number of object agents.

OOR. Given a pair of object agents (A_i^o, A_j^o), they have an OOR if and only if

$$\mathtt{oor}(A_i^o, A_j^o) = \begin{cases} 1 & if\ A_i^o.u = A_j^o.u; \\ 0 & otherwise \end{cases}$$

where $\mathtt{oor} : A^o \times A^o \rightarrow [0, 1]$ is the OOR score function. To obtain the correlation defined above, each Service Agent can reach the federated DFs and enquire the existing agents as well as their owners. The result set can be sorted based on the owners using a quick sort algorithm. Thus, OORs can be constructed for agents with the same owners which are in the same group. The complexity of such algorithm would be $\mathcal{O}(n)$ if a hashmap is used. The results are indexed by Service Agent to accelerate the retrieval.

CBOR. As defined earlier, each Object Agent A^o is assigned a set of textual tags $A^o.t = \{t_1, t_2, ..., t_k\}$ where each tag denotes a descriptive feature of the

object such as its functionality or datastream unit. For instance, an Air Quality Egg which is designed to measures the indoor air quality and temperature, can be assigned textual tags such as "^oC", "air quality" and "indoor". The tags are assigned by the users of the IoT platform and thus, can vary significantly based on their count, keyword selection, dictation and used symbols. We assume that the tag set for each object can be used for the purpose of categorization. For a given pair of Object Agents (A_i^o, A_j^o), a text similarity function $\sigma : T^2 \rightarrow [0, 1]$ and a similarity threshold $\tau \in (0, 1]$ we define the CBOR as follows:

$$\texttt{cbor}(A_i^o, A_j^o) = \begin{cases} \prod \sigma(A_i^o.t, A_j^o.t) & if\ \ \sigma(A_i^o.t, A_j^o.t) > \tau; \\ 0 & otherwise \end{cases}$$

where $\texttt{cbor} : A^2 \rightarrow [0, 1]$ is the weight function of the CBOR correlation between (A_i^o, A_j^o). Algorithm 2 shows our approach to identify and extract CBORs amongst a set of given Object Agents. Using a naive approach for finding the similarity between all pairs of object agents is time consuming and complex. Due to limiting the recursions and through the use of dynamic programming, the runtime of this algorithm would be linear. Thereupon, the three scenarios of searching for similar objects using CBOR are:

- There is no matching result and a Null value returned.
- The number of objects in the list $\leq$ *Max_results*. Thus, all objects in the list are returned.
- The number of objects exceeds the max criterion. Provided that the list is descendingly sorted, a sub list with size of *Max_results* is cut from the first element and retrieved as an answer for the query indicating that there is high potentiality to connect, correlate, cooperate, and any action can be taken with these objects.

2.4 Correlation Integration

In the Integration phase, different TCGs are merged to form a universal graph. In the Aggregated Correlations Graph (ACG), each edge has a weight that indicates the strength of their correlation. For a given set of Object Agents A, we assume that the result of the correlation extraction process in all types of correlations maintain the same format and size. Thus, the result of CLOR, OOR and CBOR correlations are modelled as $M^L_{|A|\times|A|}$, $M^O_{|A|\times|A|}$ and $M^B_{|A|\times|A|}$, respectively. In the simplest form of integration, TCGs of different types can be integrated through a weighted matrix integration as follows:

$$\overline{M} = \frac{1}{3} \sum_{M \in \{M^L, M^O, M^B\}} w_i.M \tag{1}$$

where $w_i \in [0, 1]$ is the weight that is assigned to each correlation and $\sum w_i = 3$. However, this kind of correlation may result in the loss of correlation types and not very suitable for cases in which the type of each correlation is important.

Algorithm 2: EXTRACT˙CBOR

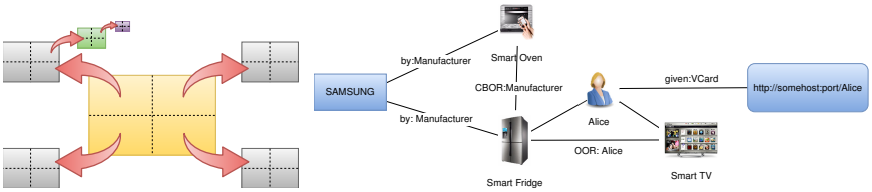
Input : Requester ID, Type, Sensor_set, Actuators_set, Max_results
Output: Results-list : List of relevant objects agents

Results = Search DF based on Type
if $Results \neq \emptyset$ **then**
  foreach $Object \in result$ **do**
    if $Sensors \neq \emptyset \;||\; Actuators \neq \emptyset$ **then**
      $S = \{Sensors\}$
      $A = \{Actuators\}$
      $S\_SIM(Sensor\_set, S) = \lceil Sensor\_set \cap S \rceil / \lceil Sensor\_set \cup S \rceil$
      $A\_SIM(Actuators\_set, A) =$
      $\lceil Actuators\_set \cap A \rceil / \lceil Actuators\_set \cup A \rceil$
      $Similarity = S\_SIM + A\_SIM$
      Assign Similarity to Object
      Add Object to *Results-list*
  Sort *Results-list* descendingly
  if $\{Results - list\} \geq Max\_results$ **then**
    Shrink the result by excluding elements from *Max_results* until the end of the list
  return *Results-list*
else
  return Null

2.5 Correlation Representation

Correlation representation is a part of the process in which all extracted correlations are presented in a standard format. We use RDF to represent correlations in IoT. Thus, we can maintain the connections between things while a large portion of them are quickly evolving. Through the use of specifically designed ontologies along with RDF correlations, our approach can be deployed to empower pattern queries for IoT search engines. In this regard, we use triple space computing (TSC) to facilitate the communication and store relationships triples. Figure 3(b) depicts an example of how things can be correlated using RDF triples, where each object is considered as a resource. Each statement is identified via a unique URI. A statement consists of three elements: *subject*, *predicate*, and *object*. A *subject* (a thing) is linked with an *object* (another thing). The connection between two *objects* is called a *predicate*. The *predicate* explains the relationship between the *subject* and the *object* of the statement.

Two objects will have OOR if and only if they have the same owner. The result is retrieved from federated DFs as graph of RDF triples similar to the Listing 1.1. In this type of relationships, we point out the possibility of connecting objects under different regimes based on a criterion such as a common owner.



(a) (b)

Fig. 3. (a) The surrounding rectangular area recursively breaks into four rectangles; and (b) An example of 2 types of relationships established among objects A, B, and C based on their common owners (OOR) and common tags for production batch (CBOR)

Listing 1.1. Example of OOR correlated object agents

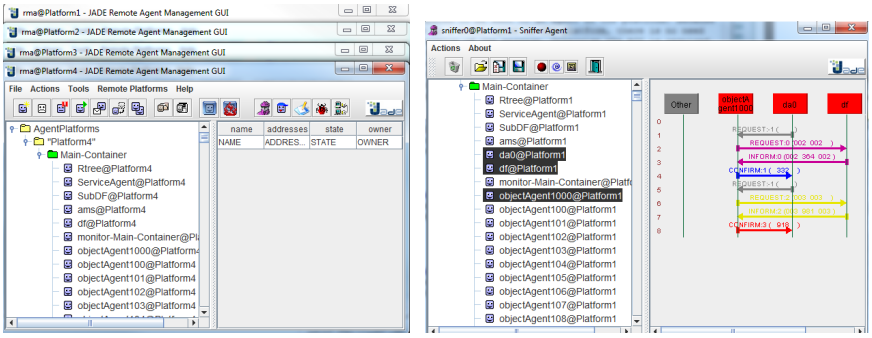
```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22 rdf syntax ns#"
    xmlns:RT="http://uqucs.com/RT"
    xmlns:vcard="http://www.w3.org/2001/vcard rdf/3.0#"
    xmlns:dc="http://purl.org/dc/elements/1.1/" >
  <rdf:Description rdf:about="http://uqucs.com/objectAgent10@Platform2">
    <RT:OOR>objectAgent43@Platform2</RT:OOR>
    <RT:OOR>objectAgent18@Platform2</RT:OOR>
    <RT:OOR>objectAgent5@Platform1</RT:OOR>
    <RT:OOR>objectAgent61@Platform2</RT:OOR>
    <RT:OOR>objectAgent42@Platform2</RT:OOR>
    <RT:OOR>objectAgent29@Platform1</RT:OOR>
    <RT:OOR>objectAgent48@Platform2</RT:OOR>
    <vcard:N>Tahani</vcard:N>
  </rdf:Description>
</rdf:RDF>
```

3 Experimental Results

In this section, we present the evaluation results for the proposed CEIoT framework. We conduct the experiments on a PC with a Core i7 2.20 GHz, 4 GB memory and Windows 7 64-bit. The details of the used datasets are as follows:

1. Synthetic dataset: we simulate a set of four IoT service providers where each service provider is supplied with one Service Agents and 1,000 Object Agents. Furthermore, each service provider is initialized on a separate platform, which can run on an independent machine or share the same machine with other service providers. We use this simulation to evaluate the framework on a distributed infrastructure and for multiple service providers. Figure 4(a) shows four RMA GUIs visualizing all four platforms.
2. Real-world dataset: we crawled the datastream of public sensors on Xively[7] using the ThingSeek crawler [11], previously known as Pachube, which is a pioneering IoT platforms on the Internet with one of the largest collections of publicly available sensors. The dataset contains around 67,000 things and their most recent sensor readings. However, after filtering records with incomplete data, only 11,894 records remain in our dataset. A primary analysis of the tag sets reveals that the tags are scattered (Figure 5(a)) but densely

[7] https://xively.com/



(a) (b)

Fig. 4. (a) RMA view of launched platforms for each service agent; (b) Sniffed communications between two object agents

focused on some tags (Figure 5(b)). Moreover, only less than 10% of the tags have been assigned to more than 60% of things (Figure 5(c)). Thus, a single label based approach without considering location based correlations will not be helpful in application.

We use tools provided by *RMA GUI* such as *Sniffer* and *Dummy* for debugging and testing purposes. Dummy agent is used to communicate with agents in the platform and command them to execute specific behaviors. We prepare the agents with *Cyclic behaviors* that are responsible for receiving these commands and their execution. Otherwise, these behaviors are blocked until a command is received to prevent infinite loop. Blocking the behaviors in agents does not block the entire agent. Additional aim of implementing agents behaviors and communication is to use them as self-explained examples of how agents can be communicated and commanded by end users of the framework using third-party agent such as *Dummy* agent.

3.1 System Performance

Due to the scale and dynamics of the IoT, the overall performance of the system is important. Mainly, the following steps are paved by the system: i) launching all independent platforms; ii) launching all agents mentioned above; iii) federating DF service; iv) communication for exchanging stetting information; v) generating RDFs and message de/serialization; and vi) R-tree insertion. Using the fixed parameters with an RMA launched for each platform, it takes the system roughly 25 seconds to perform all the mentioned key tasks. This is due to that each *Service Agent* waits for about 10 seconds to ensure all platforms are established, as well as additional 10 seconds for federation. The 20 seconds delay is used to prepare all platforms for the experiment.

We launch 4,000 object agents which are distributed over four different platforms. For the experiment, each object agent has a cycle behavior that receives

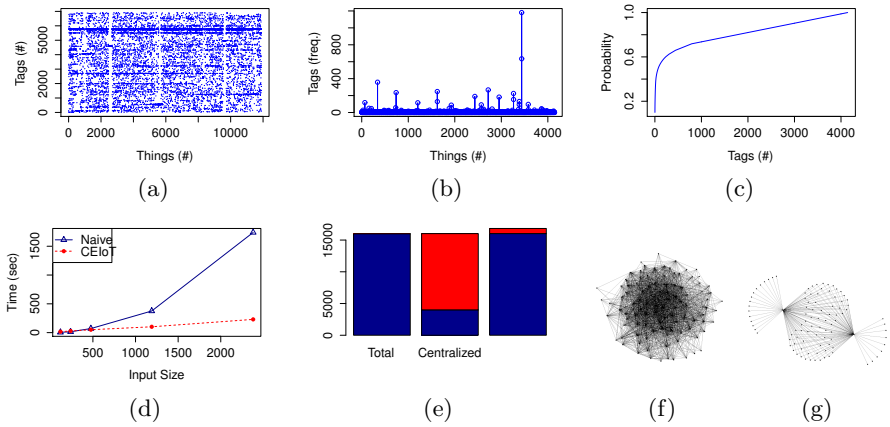


Fig. 5. (a) Things and tags relation; (b) Tags frequency; (c) Probability of reusing a tag; (d) Runtime; (e) Transacted messages for the synthetic dataset; and (e),(f) CLOR graph for things with different thresholds

a message with a *Request* per-formative act. It recognizes commands OOR, and CBOR. Otherwise, it responds with a *non-understood* message associated with commands which it can understand. If the commands are understood, it complies with them to search for other agents matching the type in the command.

The communication with Object Agent is performed as expected (Figure 4(b)). The figure shows the UML sequence for sending relationship requests by the dummy agent (d0) to an object agent. Agent d0 is used in our experiment for requesting object agents to perform a relationship on demand which are OOR or CBOR. Lines 1-4 shows that sending the OOR request, the agent received the request, then it searches the DF in Line 2, and DF returns the results to the agent in Line 3. Finally, the agent confirms to the dummy agent the success of the relationship establishment. Additionally, there should appear the RDF description for the relationships of the object agent. The same process is done with CBOR in lines 5-8. In OOR relationships extraction, it takes 200 *milliseconds* to retrieve the results and correlate them with the requester object as an RDF triple. However, in the CBOR it took approximately 1,300 *milliseconds*. For CLOR correlation, we observe the proposed system's response under a varying number of input sizes to ensure the system's scalability. We compare our approach with the naive method. As Figure 5(d) shows, the naive approach may take less time for results with small sizes. However, as the size of the input increases, for inputs with 500 or more things, our algorithm's runtime outperforms the naive approach. We find out that using the naive approach for an input size with the real-world dataset would be impractical, particularly when we consider the scope and the dynamics of the IoT.

3.2 Things Correlation Graph

For Algorithm 1, we apply different thresholds to simulate a search engine harvesting the graph. For example, it can be interesting in relationship matching a certain criteria. Thus, the threshold is used to reduce the search space and to limit it to connections with values equal or larger than the threshold. We visualize this by passing the symmetric weighted graph to the *GraphVis class*. The graph is the file holding all CLOR relationships among the Object Agents resulted from Algorithm 1. The higher the value of a threshold, the smaller is the search space. Figures 5(f) and 5(g) illustrates the samples of graphs produced based on different thresholds, respectively.

3.3 Message Volume

One of the key factors is to minimize the number of transacted messages between the machines. In our CEIoT framework, the messages that should be transacted between different platforms are summarized. Thus, we expect a dramatic reduction in the number of transacted messaged compare to a centralized scenario. Figure 5(e) shows the number of both internal and external messages which are transacted between agents during the experiment on the synthetic dataset. In this Figure, the As it is shown, although the red stack (top) shows the number of inter-platform messages and the blue (bottom) shows the number of intra-platform messages in our experiment. As shown, despite the fact that using the CEIoT approach in a distributed mode increases the number of transacted messages by a small amount, it may enhance the total throughput of the system by reducing the ratio of inter-platform messages which are quite expensive. Moreover, in the distributed mode the messages are being processed by a larger number of machines than the centralized approach.

4 Related Work

There are some studies which promote that the IoT can be implemented as the Internet of agents [16]. This is due to the ability of agents to mimic human activities such as willingness to achieve certain goals, and the social ability to interact with each other and with human as well. In a proposal by Fortino et al [5], MAS is also used in developing smart environments for objects. The heterogeneity and disparity can intensify the problem of dealing with smart things in an effective way. Things data is never under centralized control. Hence, datasets are usually stored in distributed locations. Using central location for data storage is an obstacle for materializing an effective solution. The diversity of sensors infrastructures means that, data are structured and documented in different ways, which complicates combining their datasets in an easy way. Thus, one of the solutions called *Concinnity* [9] takes the advantage of Semantic Web technologies such as RDF, Ontology, and SPARQL.

There is an interesting paradigm called *Triple Space Computing* [4] that has the potential to facilitate storage and communication in the Internet of Things

context. It is basically a dedicated Web for machines, which is combined of space-based computing and the Semantic Web. It uses RDF triples for data representation to exchange knowledge using shared space, just like HTML representation in human-driven Web [6]. TSC provides asynchronous communication such that consumers neither need to recognize each other through identifiers, nor need to concurrently consume data.

A provisional approach is the Social Internet of Things (SIoT) [2, 1, 8]. To socialize things in the IoT, unlike the solutions for socializing smart things that depend totally on their owners relationships, this approach seeks a solution to enable smart things to be interlinked by themselves. Objects have their own profiles and IDs, so they can discover other objects of interest and establish a friendship with each other according to their owners constraints. Additionally, this approach defines some types of relationships established among objects. For instance, *Parental Object Relationship* (*POR*), *Co-Location Object relationship* (*CLOR*), *Co-work Object Relationship* (*C-WOR*), and *Social Object Relationship* (*SOR*). However, no realistic and scalable solution given on how to identify and extract these relationships. Correlating things in IoT has various benefits such as better navigability experience, query result diversification and matters of an interest can be effectively discovered with the least effort possible [12]. As discussed before, Atzori et al. [2] discuss the characteristics of *Social Internet of Things* and define polices for relationships establishment among connected things. Unlike the approaches that socializes things according to what relationships their owners [8, 10], this approach is more focused on things as key players in relationships establishment, which limits the roles of their owners to managing them and setting appropriate rules for their relationships. However, to the best of our knowledge, this vision has not been implemented yet. Additionally, currently no technical details have been given on how to automate the establishment of relationships between objects when they are aware of each other. Automated correlation extraction is limited to one type of correlation for a small number of objects [15, 14].

5 Conclusion

One of the missing components in the IoT is something similar to hyperlinks in the World Wide Web. Unlike the traditional Web, establishing correlations in IoT can be challenging as it must be automated. In this paper, we have proposed the CEIoT framework for extraction and representation of correlations between things in the IoT. The framework is implemented in Java and is open-sourced. We have used a distributed architecture and local correlation filtering to stabilize the performance of the library in different conditions. One of the limitations of our work is dismissing incomplete and uncertain data entries. We plan to extend the CEIoT framework to support incomplete and uncertain data to address a large portion of the records in our dataset.

References

1. Atzori, L., Iera, A., Morabito, G.: Siot: Giving a social structure to the internet of things. Communications Letters 15(11), 1193–1195 (2011)
2. Atzori, L., Iera, A., Morabito, G., Nitti, M.: The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization. Computer Networks 56(16), 3594–3608 (2012)
3. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Proc. of the 7th Intl. World-Wide Web Conf. (WWW) (1998), `http://ilpubs.stanford.edu:8090/361/`
4. Fensel, D., Krummenacher, R., Shafiq, O., Kuehn, E., Riemer, J., Ding, Y., Draxler, B.: Tsc–triple space computing. e & i Elektrotechnik und Informationstechnik 124(1-2), 31–38 (2007)
5. Fortino, G., Guerrieri, A., Russo, W.: Agent-oriented smart objects development. In: Proc. of the 16th Intl. Conf. onComputer Supported Cooperative Work in Design (CSCWD 2012). pp. 907–912. IEEE (2012)
6. Gómez-Goiri, A., López-de Ipiña, D.: On the complementarity of triple spaces and the web of things. In: Proc. of the 2nd Intl. Workshop on Web of Things. p. 12. ACM (2011)
7. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (iot): A vision, architectural elements, and future directions. Future Generation Computer Systems 29(7), 1645–1660 (2013)
8. Guinard, D., Fischer, M., Trifa, V.: Sharing using social networks in a composable web of things. In: Proc. of the 8th IEEE Intl. Conf. on Pervasive Computing and Communications Workshops (PERCOM 2010). pp. 702–707. IEEE (2010)
9. Lee, C.H., Birch, D., Wu, C., Silva, D., Tsinalis, O., Li, Y., Yan, S., Ghanem, M., Guo, Y.: Building a generic platform for big sensor data application. In: Proc. of 2013 IEEE Intl. Conf. on Big Data (IEEE BigData). pp. 94–102. IEEE (2013)
10. Pintus, A., Carboni, D., Piras, A.: The anatomy of a large scale social web for internet enabled objects. In: Proc. of the 2nd Intl. Workshop on Web of Things. p. 6. ACM (2011)
11. Shemshadi, A., Sheng, Q.Z., Qin, Y.: Thingseek: A crawler and search engine for the internet of things. In: Proc. of the 39th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR). pp. 1149–1152. ACM (2016)
12. Shemshadi, A., Yao, L., Qin, Y., Sheng, Q.Z., Zhang, Y.: Ecs: A framework for diversified and relevant search in the internet of things. In: Proc. of the 16th Intl. Conf. on Web Information Systems Engineering (WISE 2015), pp. 448–462. Springer (2015)
13. Truong, C., Römer, K., Chen, K.: Sensor similarity search in the web of things. In: 2012 IEEE Intl. Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM). pp. 1–6. IEEE (2012)
14. Yao, L., Sheng, Q.Z.: Correlation discovery in web of things. In: Proc. of the 22nd international Conf. on World Wide Web Companion (WWW 2013). pp. 215–216. Intl. World Wide Web Conf.s Steering Committee (2013)
15. Yao, L., Sheng, Q.Z., Gao, B.J., Ngu, A.H., Li, X.: A model for discovering correlations of ubiquitous things. In: Proc. of 13th Intl. Conf. on Data Mining (ICDM 2013). pp. 1253–1258. IEEE (2013)
16. Yu, H., Shen, Z., Leung, C.: From internet of things to internet of agents. In: Proc. of the 2013 IEEE Intl. Conf. on Green Computing and Communications (GreenCom). pp. 1054–1057. IEEE (2013)