Service Mining for Internet of Things

Bing Huang, Athman Bouguettaya, Hai Dong(✉), and Liang Chen

School of Science, RMIT University,
Melbourne, Australia
{bing.huang,athman.bouguettaya,hai.dong,liang.chen}@rmit.edu.au

Abstract. A service mining framework is proposed that enables discovering interesting relationships in Internet of Things services bottom-up. The service relationships are modeled based on spatial-temporal aspects, environment, people, and operation. An ontology-based service model is proposed to describe services. We present a set of metrics to evaluate the interestingness of discovered service relationships. Analytical and simulation results are presented to show the effectiveness of the proposed evaluation measures.

Keywords: Service mining · Service recognition · Service relationship · Interestingness · Service description

1 Introduction

The current Internet is evolving from interconnecting computers to interconnecting things [3]. The Internet of Things (IoT) refers to numerous connected things that rely on sensory, communication, networking, and information processing technologies [3]. Real-world things are heterogeneous in terms of features with little standard descriptions [3]. Service-Oriented Computing (SOC) paradigm provides a promising solution for exposing features of real-world things as services in a standard form.

As a result of the prevalence of IoT, an increasing number of real-world things will be connected to future Internet, leading to a proliferation in services. This proliferation of services will contribute to the increasing opportunities of service composition. Service composition aims at providing value-added services through aggregating component services. One the one hand, as more diverse services are available, the opportunities of composing services will surpass anyone's imagination. On the other hand, we may not sometimes know which group of services in IoT is related to each other and can cooperate to achieve a common goal. Therefore, the expected large number of services in IoT, coupled with the need for composing services, call for a service mining tool that can uncover the intrinsic correlations among services. We define *service mining as the process of proactively discovering interesting relationships among services.*

Service mining aims at discovering any interesting service relationships without specific search goals for defining the exact service functionality. A general goal (i.e., smart home, which is a typical application domain in IoT) is provided at the beginning of service mining process to narrow down the mining scope. In the context of smart home, service mining may provide interesting correlations between different component services. For example, a TV service may hypothetically relate to a fridge service. There are three factors that provide the basis for the hypothetical relationship. The first factor is the collocation of both the TV set and the fridge [5]. The second factor is the temporal correlation (i.e., the fridge is sometimes accessed when the TV is on) [5]. The third factor is people (i.e., the TV and the fridge may be accessed by the same user) [6]. Another example is that an oven service may hypothetically relate to an air-conditioning service. An environment factor provides the basis for the hypothetical correlation [6]. When the oven is in use, temperature is rising. Then the air-conditioning is invoked to adjust room temperature. Thus, without any prior knowledge of specifying search goals, the service mining process has the potential of discovering interesting service relationships.

The key contribution is the service mining framework that aims at discovering service relationships in IoT. The paper is organized as follows: Sect. 2 proposes an ontology-based service description model. Section 3 elaborates the service mining framework. Section 4 shows experiment results. Section 5 concludes the paper and highlights some future work.

2 Ontology-Based Description of Services

Discovering service relationships requires the description of services so that service mining tools can understand services and form bonds among related services. We propose an ontology-based model for describing services (see Fig. 1). Unfilled nodes refer to ontology concepts that have been clearly defined in [4]. Gray nodes refer to extended ontology concepts. Numbers on edges denote the *has* relationship between ontology concepts. We formally define *service, environment, state, pre/postcondition, and people* as follows.

Definition 1: Service. A service S_i is defined by a tuple $\langle$ *name*, *des*, $Bind_i$, Cat_i, Ope_i, Sta_i, Peo_i $\rangle$ where:

- *name* and *des* are a name and a text summary about the service features, respectively.
- $Bind_i$ is a set of binding protocols supported by S_i.
- Cat_i is a set of categories that S_i belongs to.
- Ope_i is a set of operations provided by S_i (cf. Definition 5).
- Sta_i is a set of states that S_i represents (cf. Definition 3).
- Peo_i is people who consume the service S_i (cf. Definition 6).

Definition 2: Environment. The environment Env_i is defined as a tuple $\langle$ *name*, (val,uni,ts_i,loc_i) $\rangle$, where:

- *name* is an environmental variable of Env_i.

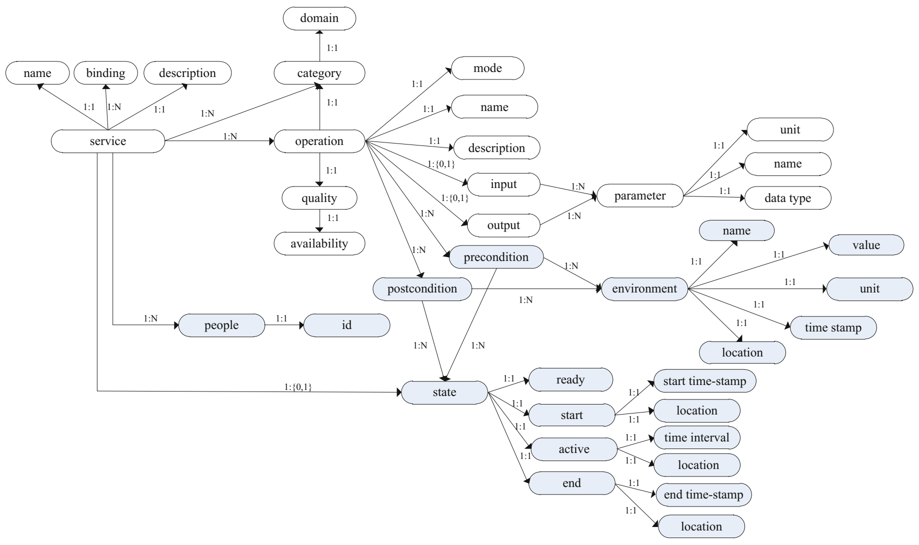

Fig. 1. Ontology-based description of services

- (val,uni,ts_i,loc_i) shows the recorded readings of the environmental variable. The readings include value val, unit uni, time-stamp ts_i and location loc_i. The location is uniformly defined as a GPS point with a user-defined spatial radius r. For simplicity, we use v_t to refer to the value of the environment variable at time point t.

Definition 3: State. The state Sta_i can only be an attribute in the tuple $\langle$ *ready*, *start*, *active*, *end* $\rangle$ at a given time where:

- *ready*: the service is in the *ready* state if an invoking request has not been made.
- *start*: the *start* state means that the service execution has been initiated. The start state is defined by a tuple $\langle\ st_i,\ loc_i\ \rangle$ where st_i and loc_i are the start time stamp and the location of initiating the service, respectively.
- *active*: the service is in the *active* state if the service is executing. The active state is defined by a tuple $\langle\ tin_i,\ loc_i\ \rangle$ where tin_i and loc_i are the time interval and the location of the service execution, respectively, and $tin_i = et_i\text{-}st_i$.
- *end*: the service is in the *end* state if the service execution is terminated. The end state is defined by a tuple $\langle\ et_i,\ loc_i\ \rangle$ where et_i and loc_i are the end time stamp and the location of terminating the service, respectively.

Definition 4: Precondition and Postcondition. The precondition Pre_i is defined as a tuple $\langle\ (Sta_1,\ Sta_2 \cdots Sta_i),\ (Env_1,\ Env_2 \cdots Env_i)\ \rangle$ where:

- Sta_i is the required *state* before executing an operation (cf. Definition 3).
- Env_i is the required *environment* before executing an operation (cf. Definition 2).

The postcondition Pos_i is defined as a tuple $\langle$ $(Sta_1, Sta_2 \cdots Sta_i)$, $(Env_1, Env_2 \cdots Env_i)$ $\rangle$ where:

- Sta_i is the effect state after executing an operation (cf. Definition 3).
- Env_i is the effect environment after executing an operation(cf. Definition 2).

Definition 5: Operation. An operation Ope_i is defined by a tuple $\langle$ $name$, des, Cat_i, Mod_i, Inp_i, Out_i, Qua_i, Pre_i, Pos_i $\rangle$ where:

- $name$, des, Cat_i, Mod_i, Inp_i, Out_i, Qua_i refer to names, descriptions, categories, modes, input/output messages, and qualities, respectively [4].
- Pre_i gives the preconditions of an operation (cf. Definition 4).
- Pos_i gives the postconditions of an operation (cf. Definition 4).

Definition 6: People. Peo_i is defined as a tuple $\langle$ id $\rangle$ where

- id is the unique identifier of people.

3 Service Mining Framework

The service mining framework consists of three phases. The framework starts with the scope specification phase which consists of two steps. In the first step, the user specifies a list of domains termed as *mining context*(MC). Domains are specified by a set of ontologies [2]. Hence, ontologies related to MC can be determined, which is denoted as $Ont(MC)$. In the second step, a list of services related to the mining context are identified, which are termed as *mining library*(ML). Due to page limit, the detailed process of scope specification is omitted. The scope specification is followed by the automatic service recognition phase. In this phase, services will go through service recognition process for identifying related services which are represented as service composition leads. In the evaluation phase, evaluation methodologies are proposed to identify interesting service composition leads.

3.1 Service Recognition

We model service relationships from *states*, *environments*, *people*, and *operations* perspectives.

(1) Two services S_i and S_k have *state* relationship if they are spatial-temporal dependent. S_i and S_k have spatial dependency if $S_i.active.loc_i$ is located inside the spatial circle centred at $S_k.active.loc_k$ with a geographic radius r [1]. For example, the TV located in home A has a spatial dependency with the fridge located in home A. This TV has no spatial dependency with the fridge located in home B. We formalize the spatial dependency between S_i and S_k in Eq. (1).

$$distance(S_i, S_k) = 2arcsin\sqrt{sin^2\frac{a}{2} + cos(Lat_i) \times cos(Lat_k) \times sin^2\frac{b}{2}} \times r_e \tag{1}$$

where $a = Lat_i - Lat_k, b = Long_i - Long_k$, r_e is the radius of earth. Lat_i and $Long_i$ refer to the latitude and longitude of the location $S_i.active.loc_i$. $S_i.active.loc_i$ is GPS points of S_i. r is a user defined radius. If $distance(S_i, S_k) \leq r$, S_i and S_k are spatial dependent. S_i and S_k have temporal dependency if one of the following conditions is satisfied.

- S_i is invoked in the execution time interval of S_k. That is $S_i.start.st_i \geq S_k.start.st_k$ and $S_i.end.et_i \leq S_k.end.et_k$. For example, the fridge door is opened and closed during the time interval when TV is on.
- S_i is invoked before S_k. That is $S_i.end.et_i \leq S_k.start.st_k$. For example, the fridge door is closed and then TV is on.
- S_i is invoked before S_k and becomes the end state after S_k is invoked. That is $S_k.start.st_k < S_i.end.et_i < S_k.end.et_k$ and $S_i.start.st_i < S_k.start.st_k$. For example, the oven is in use for a while, then the air-conditioning is turned on. The air-conditioning keeps the active state for a time interval after the oven is turned off.

(2) Two services S_i and S_k have environment relationships if the following two conditions are satisfied. We use $\rightarrow$ to denote state changes.

- the value of environment variable Env_j is changed during the execution time interval t_{in} of S_i. That is $S_i.state = active$ and $Env_j.v_{st} \neq Env_j.v_{st+t_{in}}$.
- the state of S_k is transformed due to the environment changes. That is $S_k.ready \rightarrow S_k.start \rightarrow S_k.active$ or $S_k.active \rightarrow S_k.end \rightarrow S_k.ready$.

(3) Two services S_i and S_k are related through people if S_i and S_k are consumed by the same person in a time interval. That is $S_i.Peo_i = S_k.Peo_k$. For example, the TV and the fridge are used by the same people in a time interval.

(4) Two services S_i and S_k have operation relationships if the two operations $S_i.Ope_i$ and $S_k.Ope_k$ are syntactically compatible. The following two conditions must be satisfied.

- $S_i.Ope_i.Mod_i =$ notification and $S_k.Ope_k.Mod_k =$ one-way; or $S_i.Ope_i.Mod_i =$ one-way and $S_k.Ope_k.Mod_k =$ notification; or $S_i.Ope_i.Mod_i =$ solicit-response and $S_k.Ope_k.Mod_k =$ request-response; or $S_i.Ope_i.Mod_i =$ request-response and $S_k.Ope_k.Mod_k =$ solicit-response [4]. This condition implies that a one-way operation must be mapped to a notification operation and a solicit-response operation must be mapped to a request-response operation.
- $S_i.Ope_i.Out_i \supseteq S_k.Ope_k.Inp_k$.

3.2 Evaluation

Not all discovered service composition leads are necessarily interesting in the service recognition phase. An evaluation measure is needed to filter out uninteresting service composition leads. The evaluation phase consists of two steps.

The first step is *Correlation Degree* (CD) filtering which measures the relationship strength between two services. *Correlation Degree* (CD) is defined in Eq. (2).

$$CD(S_i, S_k) = \eta_1 \cdot State(S_i, S_k) + \eta_2 \cdot Env(S_i, S_k) + \eta_3 \cdot Peo(S_i, S_k) + \eta_4 \cdot Ope(S_i, S_k) \tag{2}$$

where $\eta_1 + \eta_2 + \eta_3 + \eta_4 = 1$, $State(S_i, S_k), Env(S_i, S_k), Peo(S_i, S_k), Ope(S_i, S_k)$ are binary values returned from the service recognition phase. We define a *correlation threshold* (ζ) which is the minimum value allowed for the CD. If $CD(S_i, S_k) \geq \zeta$, then leads of composed services S_i and S_k are selected for further evaluation. Otherwise S_i and S_k are filtered out.

The second step is *interestingness* evaluation. We first give the concept of *Availability* (Ava), *Domain Correlation*(DC), and *Diversity*(Div), and then give the *interestingness* definition.

Availability. Ava is defined as a binary value (i.e., 1 for available, 0 for unavailable). The source of availability information can be a registry that keeps tack of the availability of services [2].

Domain Correlation. DC measures the relevance of two domains that services S_i and S_k belong to, respectively. The domain correlation of S_i and S_k is equivalent to the ontology correlation of $Ont(S_i)$ and $Ont(S_k)$. We formally define DC in Eq. (3).

$$DC(S_i, S_k) = e^{-\frac{1}{\lambda_0 \cdot \{sim(Ont(S_i), Ont(S_k)) + 1\}}} \tag{3}$$

Where

$$Sim(Ont(S_i), Ont(S_k)) =$$
$$w_1 \cdot (\frac{|N_{pre}(S_i)| \cap |N_{pre}(S_k)|}{|N_{pre}(S_i)| \cup |N_{pre}(S_k)|}) + w_2 \cdot (\frac{|N_{pos}(S_i)| \cap |N_{pos}(S_k)|}{|N_{pos}(S_i)| \cup |N_{pos}(S_k)|})$$
$$+ w_3 \cdot (\frac{|N_{in}(S_i)| \cap |N_{in}(S_k)|}{|N_{in}(S_i)| \cup |N_{in}(S_k)|}) + w_4 \cdot (\frac{|N_{out}(S_i)| \cap |N_{out}(S_k)|}{|N_{out}(S_i)| \cup |N_{out}(S_k)|})$$

where $S_i, S_k \in ML, w_i (i = 1, 2, 3, 4)$ is a weight such that $w_i \in [0, 1]$ and $\sum_1^4 w_i = 1$. $|N_{pre}(S_i)|$, $|N_{pos}(S_i)|$, $|N_{in}(S_i)|$, and $|N_{out}(S_i)|$ refer to the set of preconditions, postconditions, input parameters, and output parameters of concepts, respectively. Operators $\cap$ and $\cup$ are ontological overlap and union of two concepts. When $Sim(Ont(S_i), Ont(S_k)) = 0$, the correlation between two domains is assigned with an initial value $r_0 = e^{-\frac{1}{\lambda_0}}$. Equation (3) shows that $DC(S_i, S_k)$ approaches 1 as $Sim(Ont(S_i), Ont(S_k))$ increases. We define Div as the multiplicative inverse of the domain correlation. We bound the maximum value of Div to 1. The Div is formally defined as follows.

$$Div = \frac{r_0}{DC(S_i, S_k)} \tag{4}$$

The *interestingness* is defined as follows.

$$Interestingness = Ava \cdot w_1 + Div \cdot w_2 \tag{5}$$

where w_1, w_2 are weights for Availability and Diversity respectively, $w_i \in [0,1](i = 1,2)$, and $\sum_1^2 w_i = 1$. We define an interestingness threshold (ξ) which gives the minimum value allowed for interesting service composition leads. If the value of $interestingness \geq \xi$, then leads of composed services are considered interesting. Otherwise the service composition leads are considered uninteresting.

4 Experiment Results

We study the effect of variables listed in Table 1 on the total number of discovered service composition leads, the number of service composition leads after Correlation Degree filtering, and the number of interesting service composition leads. First, we model the ontology using a class whose domain attribute stands for the domain of the ontology. We can obtain services with different domain attributes through initializing the class by assigning different values to domains. The rules of generating values shown in Table 1. For example, we randomly generate its input/output parameters such that the number of these parameters uniformly falls in the range of 0 to 5. The data type of each parameter is integer. The bound of each parameter is 0 to 100. For simplicity, we only consider the exact input/output parameter data type match.

Figure 2 (line a) shows that the total number of service composition leads increases significantly as the number of services increases. This is an expected result because as more services are introduced to the mining process, a service has a higher chance of relating to other services. Figure 2 (line b) shows the effect of the *CD* on filtering out uninteresting service composition leads.

Table 1. Experiment settings

Variable	Value or Range
Number of input parameters per operation	0–5
Number of output parameters per operation	0–5
Number of pre/pos-condition per operation	0–3
Range of input/output per operation (integer)	0–100
Range of pre/pos-condition per operation (float)	0–1
Temporal range (time-stamp)	0–24
Availability	0/1
r	10
$\eta_1/\eta_2/\eta_3/\eta_4$	0.1/0.2/0.3/0.4
w_1/w_2	0.3/0.7
λ_0	0.1
ζ	0.3
ξ	0.6

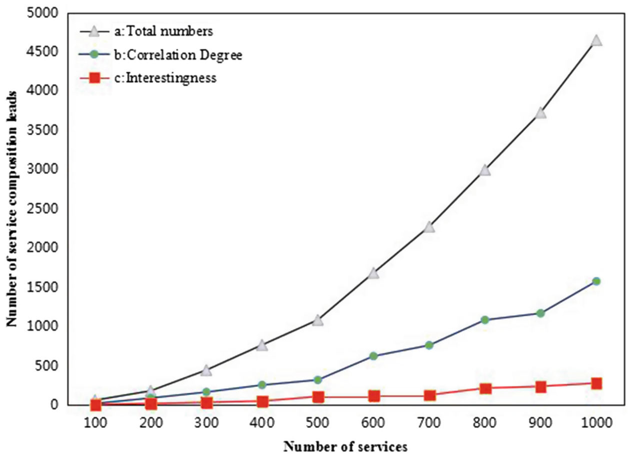

Fig. 2. Effects of key variables

With the number of services increasing, the *CD* can filter out uninteresting service composition leads to a large extent. Although the *CD* filtering technique reduces the number of service composition leads to a much smaller size, there are service composition leads that exhibit high *CD* values but are commonly known or useless. The interestingness measure (Fig. 2 (line c)) further filters out uninteresting service composition leads after the *CD* filtering. Compared with Fig. 2 (line b), interestingness measures can reduce uninteresting service composition leads significantly.

5 Conclusion

We propose a service mining framework that enables the proactive discovery of interesting service relationships. We propose an interestingness evaluation measure to sort out interesting service composition leads. We also proposes an ontology model for describing services. Future work includes developing a tool aiming at providing visual aids toward representing the discovered service composition leads. We also plan to improve the agility of our service mining framework to accommodate for the dynamic expansion of services.

Acknowledgements. This research was made possible by NPRP 7-481-1-088 grant from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. Neiat, A.G., Bouguettaya, A., Sellis, T., Dong, H.: Failure-proof spatio-temporal composition of sensor cloud services. In: Barros, A., Grigori, D., Narendra, N.C., Dam, H.K. (eds.) ICSOC 2015. LNCS, vol. 9435, pp. 368–377. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45391-9_26
2. Zheng, G., Bouguettaya, A.: Service mining on the web. IEEE Trans. Serv. Comput. **2**(1), 65–78 (2009)
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. J. Comput. Netw. **54**(15), 2787–2805 (2010)
4. Medjahed, B., Bouguettaya, A.: Composing web services on the semantic web. J. VLDB **12**(4), 333–351 (2003)
5. Baldauf, M., Dustdar, S.: A survey on context-aware systems. J. Ad Hoc Ubiquit. Comput. **2**(4), 263–277 (2007)
6. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Proceedings of International Symposium on Handheld and Ubiquitous Computing, pp. 304–307 (1999)