Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2016, Vol. 12(7) 1–11

DOI: 10.1177/1550147716657925
ijdsn.sagepub.com
SAGE

Automated knowledge discovery and semantic annotation for network and web services

Szu-Yin Lin[1], Chia-Chen Chung[2], Wei-Che Hu[1], Chihli Hung[1], Shih-Lun Chen[3] and Ting-Lan Lin[3]

Abstract

With the rise of the Internet of things, the smart environmental issue is becoming increasingly important. Sensor web is one of the best solutions to this issue and provides the advantages of sensor networks and web services. Ontology web language for services (OWL-S) is an OWL-based web services ontology, which provides the ability to describe the semantics of web services and their capabilities in a formal and machine-processable manner. Moreover, it aids semantic service matching, selection and composition. However, automatically annotating semantic web services is a highly complicated and tedious task. In this study, we propose a methodology to uncover information in the history data and profiles of web services and then semantically annotate them. With the proposed approach, semantic relationships between web services could be extracted via a combination of association rules and input/output matching. Our results show that this hybrid automated knowledge-discovery approach works better than traditional approaches do. We also provide a scenario to explain how the proposed methodology works.

Keywords

Web services, semantic annotation, knowledge discovery in services, OWL-S

Date received: 3 March 2016; accepted: 7 June 2016

Academic Editor: Hongming Cai

Introduction

In recent years, with the increasing prevalence of the Internet of things (IoT), the smart environmental issue has become increasingly important. A 'sensor web', a type of sensor network, is one of the best solutions to this issue.[1,2] The terminology of the sensor web is also associated with sensing systems that heavily use the World Wide Web as a platform. Open Geospatial Consortium (OGC)'s sensor web enablement framework defines a suite of web service interfaces, standards and communication protocols abstracted from the heterogeneity of sensor network communication.[3] Therefore, service-oriented architecture (SOA) and web services technologies play important roles in the issue of sensor web monitoring in smart environments.

In the IoT environment, we can build integrated semantic service-oriented systems to support semantic interoperability.[4] Semantic technologies have been used in recent years as a key solution to provide formalized representations of real-world data. The advantage of applying semantic technologies to sensor data lies in the

[1]Department of Information Management, Chung Yuan Christian University, Taoyuan City, Taiwan
[2]Institute of Information Management, National Chiao Tung University, Hsin-Chu City, Taiwan
[3]Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan City, Taiwan

Corresponding author:
Szu-Yin Lin, Department of Information Management, Chung Yuan Christian University, Taoyuan City, Taiwan.
Email: szuyinlin@gmail.com

conceptualization and abstract representation of raw data, making them machine interpretable and interlinking the data with existing resources on the Web. Sensor data related to different events and occurrences can be analysed and turned into actionable knowledge. One of the primary goals of interconnecting devices and collecting/processing data from them is to create situational awareness and enable applications, machines and human users to better understand their surrounding environments. However, the data collected by different sensors and devices are usually multimodal and diverse in nature. Owing to the large amount of sensor data, it is difficult to determine the relationships between them.

For the reasons mentioned above, we will discuss how to combine the context of networks and web services with semantic technologies to contribute to the future intelligent environment. The goals of this study are to transform data into actionable knowledge and intelligence as well as to propose solutions for automated knowledge discovery and semantic annotation for network and web services.

Research on SOA and web service problems has become increasingly important in recent years due to the challenge of automating processes as well as the growing number of sensor networks and web services over the Internet. There are three basic roles for participants in SOA. The client (or consumer) uses a service, the provider offers a service and the broker (or registry) is a mediator who matches clients and providers. SOA can be achieved with several technologies, such as older common object request broker architecture. Recently, more researchers have become interested in a web services standard. Web services use the eXtensible markup language (XML) to access and describe services.

Web services are Internet-based software components that have the capability to deliver services across platforms and languages. The World Wide Web consortium (W3C) organization has defined a 'web service' as:

> a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language, WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.[5]

In addition, universal description discovery and integration (UDDI) works as an intermediary between service providers and users. Service providers can publish meta-information of services to locate web services, while users can send their requirements as queries to search for and find needed services. In other words, the purpose of SOA and web services is to address the requirements of loosely coupled, standards-based and protocol-independent distributed computing.[6] With this architecture, all services (or web services) can invoke or communicate with each other. It would not only be more flexible when simultaneously using resources but also be easier to integrate many different platforms and systems. Figure 1 shows the roles, techniques and interaction processes of the web services standard.

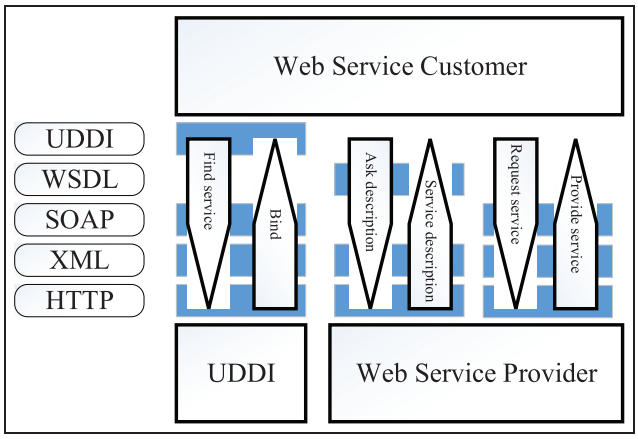

Figure 1. Roles of the web services standard.

Service-oriented computing (SOC) promotes the idea of assembling application components with standard interfaces into a network of services that can be loosely coupled to create flexible, dynamic business processes and agile applications that span organizations and computing platforms.[7] There are several pivotal, inherently related research themes in SOC, such as service matching[8] and discovery;[9] service composition; service management and monitoring and service-oriented engineering. Most of these issues, which are related to selection and composition, converge on the same fundamental problem: 'What additional information and methodologies are required to facilitate web service matching to increase accuracy?'

The semantic web approach is one of the best solutions to answer the above question. It provides a common framework that allows data to be shared and reused across applications, enterprises and community boundaries. It is a collaborative effort led by W3C with participation of numerous researchers and industrial partners. Ontology web language for services (OWL-S), a language based on the OWL, is used to describe semantic annotation. The rapid development of semantics in web services has led to an increased use of OWL-S for semantic modelling. OWL-S has three primary subclasses:[10] the service profile, the process model and the service grounding. The service profile is used to describe what the service does, including information about the service provider and functional descriptions of the services; the process model is used to describe how the service is used and the service grounding is used to describe how the service works, for example, the

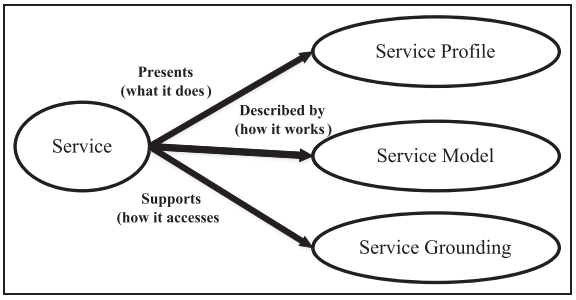

Figure 2. The OWL-S service ontology. OWL-S: ontology web language for services.

communication protocol. Figure 2 shows the relationship between the service ontology and its subclasses.

The semantic web approaches to web services not only give us the ability to describe the semantic relationships of the web services and their capabilities in a formal and machine-processable manner but also facilitate solving web services problems, such as service discovery, selection, matching, recommendation and composition. However, the manual procedure of semantic annotation for web services by experts is a complicated task. In previous studies, several researchers have tried to solve this problem.[11] For example, Paolucci et al.[12] proposed an approach to convert the web services description language (WSDL) of web services into DARPA agent mark-up language for services (DAML-S) descriptions. Web services providers could use DAML-S to unambiguously describe the concepts and functionalities of their web services. However, this method does not convert the semantic relationship into a DAML-S description owing to the different information records in WSDL and DAML-S; therefore, the translation output from WSDL does not have the complete process and profile information. This incomplete information may result in incorrect service matching. Il-Woong and Kyong-Ho[13] proposed another methodology to convert unified modelling language (UML) into OWL-S. However, the UML model needs to be built before generating the OWL-S description, and it would be complex to build or construct the UML model for every SOA system.

Therefore, the main purpose of this study is to propose a methodology for semantic knowledge discovery and automatic semantic annotation in web services. The concept of knowledge discovery in services (KDS)[14] is applied in this methodology and supports the extraction of semantic knowledge. There are two phases in the proposed methodology. The first phase of this study is semantic knowledge extraction, that is, extraction of semantic knowledge from the information in the service profile and service description. The service profile is recorded as the historical behaviours of services, such as how they are used and who has invoked them. The service description is recorded in the cloud database, which is provided by the service provider. Using the service profile and the service description, the proposed approach extracts the semantic relationships between the services or users and then obtains the relevant semantic knowledge. The second phase of this study implements a system to automatically generate semantic annotation from the semantic knowledge. This will help users select a suitable web service more quickly. This system can automatically generate annotation information and descriptions. This means that a user does not need to manually create and edit OWL-S files. The semantic annotation results are stored in several owl files: *service.owl, profile.owl, process.owl* and *grounding.owl*. Moreover, this approach generates control constructs in the process file, such as 'Sequence' and 'If-then-Else'.

The remainder of this article is organized as follows. The next section presents the proposed approach to automatically extract semantic knowledge and semantic annotations. Section 'Scenario case' demonstrates a scenario to explain the proposed novel approach. Then, sections 'Simulation and results' and 'The implemented system' show the experimental results and the implemented system. Finally, we provide some concluding remarks and future directions of study in 'Conclusion' section.

Materials and methods

In this section, we introduce the architecture and approaches that can facilitate knowledge discovery and automatic semantic annotation in web services.

Research architecture

The architecture of the proposed approach for the automatic semantic annotation of web services is shown in Figure 3. The system process consists of five steps.

Obtain information from web services, UDDI and user profiles. There are three main information sources in this framework. The service provider supplies web services, which are described in the WSDL and registry in agents, such as UDDI, and the historical continuous usage data from users, which are stored in the user profiles. Therefore, the information comes from three different places: (1) the agent (UDDI), for example, business name, business description, service name, service description and category; (2) the service description (WSDL), for example, service functional information, such as input, output and operation and (3) the user profiles, that is, the users with past records of using web services.

Locate semantic knowledge via KDS. The technique of KDS can be applied to extract semantic knowledge. It

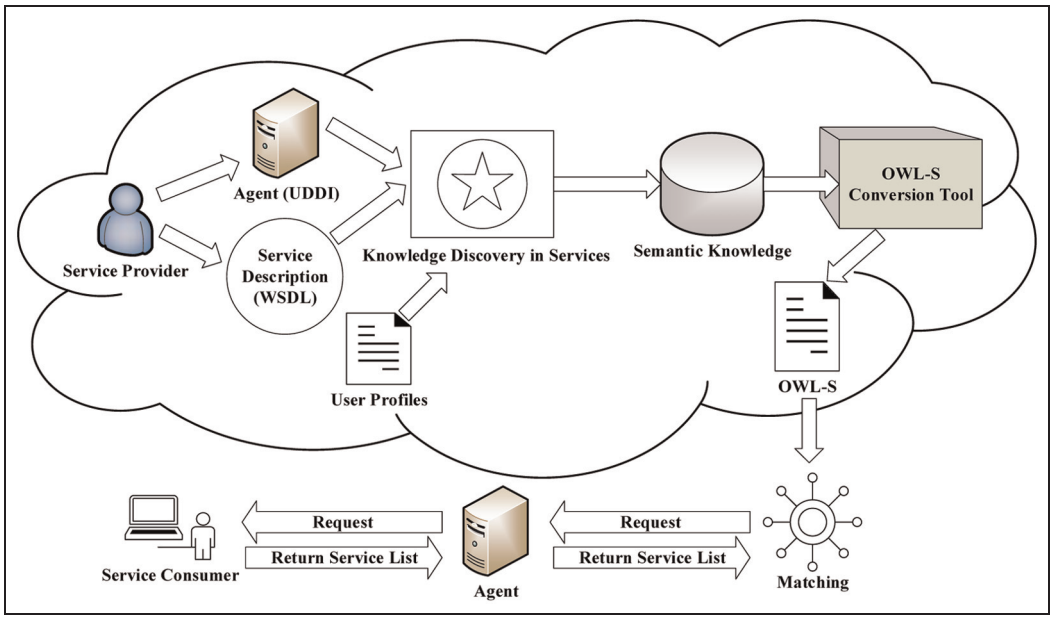

Figure 3. Architecture of the proposed approach for the automatic semantic annotation of web services.

is the core semantic knowledge extraction engine in this study. Further detailed KDS methodology will be introduced in section 'Methodology'.

Generate semantic rules. The results of semantic knowledge extracted from (1) the association rules (ARs), (2) the affinity and (3) the hybrid approach are combined into a XML and semantic knowledge mark-up in semantic web rule language (SWRL) file.

Translate semantic rules into OWL-S. The above rules of semantic knowledge, which are represented in SWRL, are transformed to generate semantic annotation OWL-S files.

Support semi-automatic OWL-S generation with a simple editing interface. Finally, a simple editing interface was designed to support semi-automatic OWL-S generation by users. First, the consumer makes a request to the agent for services, and then the agent uses the information given by the consumer to match them with services based on the OWL-S language. Finally, a list of appropriate recommended services is returned.

Methodology

The concept of KDS can be applied to extract semantic knowledge in our proposed methodology. KDS is a technology to acquire knowledge from services.[14] To generate semantic annotation automatically in web services, this study uses KDS to extract semantic knowledge and then translate it into a standard semantic information format.

Knowledge discovery in databases (KDD) is a well-known method in data mining, which is used to extract knowledge from databases. KDS has five phases that are essentially similar to KDD.[14–16]

1. Discovery phase: Find services in the service repository and extract service basic information, such as the service name.
2. Categorization phase: Group services using similar terms.
3. Equivalence processing and clustering phase: Link services together via their input and output (IO).
4. Filtering phase: Obtain or filter the most useful services.
5. Presentation phase: Display the new knowledge.

In this study, KDS can be applied in our architecture as a core methodology to extract semantic knowledge. Figure 4 shows the detailed process proposed by this study, which uses KDS. The phases are (A) discovery and extraction, Bb) IO matching, (C) ARs, (D) mash-up processes, (E) semantic knowledge and (F) generate semantic annotation.

Discovery and extraction. Blake[14] integrated multiple repositories in KDS's discovery phase with the help of a common standard. In this study, services information can be extracted from the service profile, such as the

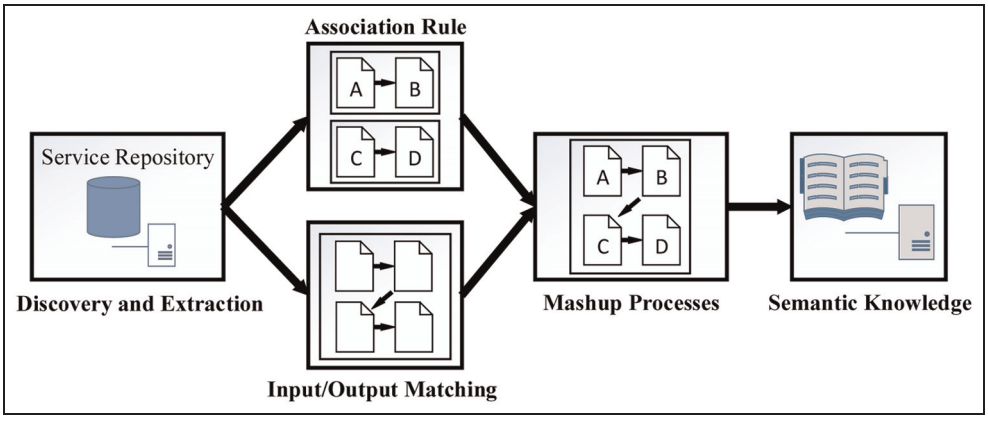

Figure 4. Methodology process.

service name, service description or contact information. Then, using the domain ontology, the services that satisfy the request can be discovered.

IO matching. If there is no profile that records how the services are being used, 'IO matching' proposed by Kun Yue et al.[15] can be used to calculate the relationship between services as the semantic annotation. For any two services A and B($A \neq B$), the association between A and B is defined by equation (1):

$$\mathrm{aff}(A, B) = \frac{|A.\text{Onput type list} \cap B.\text{Input type list}|}{|B.\text{Input type list}|}, \quad (0 \leq \mathrm{aff}(A, B) \leq 1). \tag{1}$$

In equation (1), the range of $\mathrm{aff}(A, B)$ is $(0 \leq \mathrm{aff}(A, B) \leq 1)$, where $\mathrm{aff}(A, B)$ indicates the affinity when A invokes B and Input type list (orOnput type list) indicates the type of input or output. If the services have higher affinity (higher than the threshold), they can be grouped into a set, which means they are more likely to be composited together.

Association rules. An AR is a method of data mining to find associations between items in a large database. Because ARs can find the relationships between items, the web services relationships, such as how they can be used or their sequence, can be found using this method. The Apriori algorithm is used to examine the relationship between services. The minimum support threshold and minimum confidence threshold are used to examine the association between two services. For any two services A and B($A \neq B$), the support and the confidence for A and B ($A \Rightarrow B$) are defined as equations (2) and (3), respectively:

$$\mathrm{Support}\left(A \Rightarrow B\right) = P(A \cup B) = \frac{\text{Number of services containing both A and B}}{\text{Total number of services}}, \tag{2}$$

$$\mathrm{Confidence}\left(A \Rightarrow B\right) = P(B|A) = \frac{\text{Number of services containing both A and B}}{\text{Number of services containing A}}. \tag{3}$$

The support is the percentage of the population that satisfies the rule. The confidence is defined as the measure of certainty or trustworthiness associated with each discovered pattern.

Mash-up processes. This phase combines or mashes up processes based on the result from the IO matching and the 'ARs'. Let m_α be a rule that has high similarity for matching IO lists and $M_\alpha = \{m_1, \ldots, m_\alpha\}$ be a set of rules obtained from the IO matching. Let a_β be a rule that has relationships with high probability and $A_\beta = \{a_1, \ldots, a_\beta\}$ be a set of rules obtained from the ARs. Then, the mash-up process that merges the above results can be represented as $P_\gamma = \{M_\alpha \cup A_\beta\}$.

Semantic knowledge. In this phase, those services that can be combined together will be recorded in the semantic knowledge database. Then, the semantic annotation can be generated based on the semantic knowledge. Let o_α be a control construct, such as a sequence or if-then-else and $O_\alpha = \{o_1, \ldots, o_\alpha\}$ be a set of control constructs in the process model. Let c_β be a condition of a control construct and $C_\beta = \{c_1, \ldots, c_\beta\}$ be a set of conditions. Let e_γ be an effect of a control construct and $E_\gamma = \{e_1, \ldots, e_\gamma\}$ be a set of effects.

Table 1. The user profiles using the web services.

Record	Web services
1	W_1, W_2
2	W_1, W_3
3	W_1, W_3
4	W_5, W_1, W_3
5	W_5, W_4

Table 2. Input and output lists for the web services.

Service	Input list	Output list
W_1	City	Longitude, latitude
W_2	Longitude, latitude	Weather
W_3	Road, zip code	City, scenic spot
W_4	City, longitude, latitude	Road, zip code
W_5	City, zip code, road	Weather

Then, the semantic knowledge can be represented by S_δ, where $S_\delta = \{o_\alpha, c_\beta, e_\gamma\}$.

Generate semantic annotation. After the semantic knowledge is generated, it can be applied to automatically generate semantic annotation in the OWL-S format. This helps users find web services more quickly and accurately via knowledge of standard semantic relationships. Further, we can obtain OWL-S descriptions more easily without involving experts to edit specific platforms.

Table 3. The affinity in the scenario.

Previous \ Following	W_1	W_2	W_3	W_4	W_5
W_1	×	1*	0	0.67*	0
W_2	0	×	0	0	0
W_3	1*	0	×	0.33	0.33
W_4	0	0	1*	×	0.67*
W_5	0	0	0	0	×

*The affinity is higher than the threshold.

Scenario case

In this section, a scenario is described in detail according to the methodology of section 'Materials and methods'. The main phases to extract semantic knowledge in this scenario are given below.

Discovery and extraction

Assume that there are five web services named W_1, W_2, W_3, W_4 and W_5. The user profiles using these services in the database are shown in Table 1.

IO matching

Table 2 shows the IO lists of W_1, W_2, W_3, W_4 and W_5.

Assume that the threshold of the IO matching is 60%. If the affinity is higher than 60%, they have a higher probability of being used together. Table 3 shows some examples to calculate the affinity between services. From the results in Table 3, several rules can be deduced, that is, some web services have a higher probability of being clustered together because they have higher IO matching values. There are some rules ($W_1 \rightarrow W_2, W_3 \rightarrow W_1, W_1 \rightarrow W_4, W_4 \rightarrow W_3, W_4 \rightarrow W_5$) that can be found in this phase.

Association rules

Assume that the minimum support is 40% and the minimum confidence is 70%. According to equation (2), the index 'support' is a kind of AR that involves two web services. In Figure 5, first, to compute the large item sets in the database, Apriori algorithm first generates the candidate set C1 = {W_1, W_2, W_3, W_4, W_5}, then scans the database to obtain the support of each item set in C1; then, to compare their support with the minimum support and obtain a set of frequent item sets. Second, generate candidate item sets and next, to calculate their confidence and compare them with the minimum confidence. Finally, generate string ARs. Several instances are presented in Figure 5.

From the above result, the rule ($W_1 \rightarrow W_3$) can be identified in this phase.

Mash-up processes

Table 4 shows the input data and the result for each phase. The results from IO matching and ARs are then merged and combined via the mash-up processes.

Generate semantic knowledge

The extracted results of the semantic knowledge, (1) the ARs, (2) the affinity and (3) the hybrid approach, are combined into a XML and SWRL file. Table 5 shows an example of a knowledge rules mark-up in the XML language and Table 6 shows another example of a semantic knowledge mark-up in SWRL.

Generate semantic annotation

Finally, we use the semantic knowledge to generate the semantic annotation formed by OWL-S. Tables 7 and 8 are a simple demonstration of transforming knowledge into an annotation result.

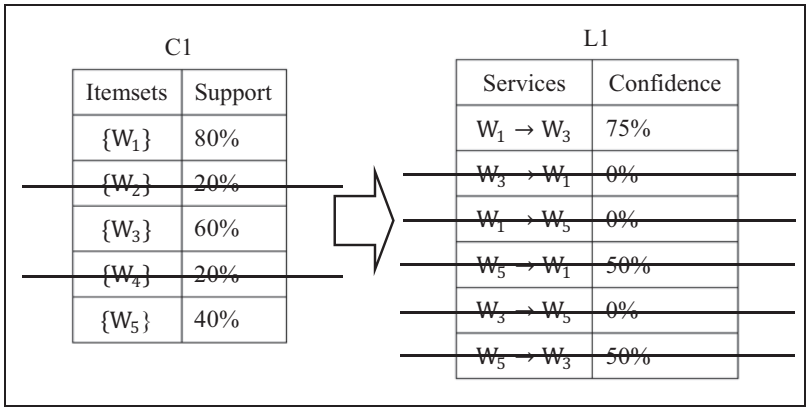

Figure 5. Examples of ARs. ARs: association rules.

Table 4. Phases to extract semantic knowledge.

Phase	Input data	Result
Discovery and extraction	Business name, service name, service description, category	W_1, W_2, W_3, W_4, W_5
Input and output matching	Inputs and outputs of W_1, W_2, W_3, W_4, W_5	{W_1, W_2}{W_3, W_1}{W_1, W_4}{W_4, W_3}{W_4, W_5}
Association rules	Profiles of W_1, W_2, W_3, W_4, W_5	{W_1, W_3}
Mash-up processes	Results of two main methods	Merge together: {W_1, W_2}{ W_3, W_1}{ W_1, W_4} {W_4, W_3}{ W_4, W_5}{ W_1, W_3}
Semantic knowledge	Result of mash-up and services' operation	Example: If W_1 is invoked, then W_3 may be invoked next.

Table 5. Knowledge rules mark-up in the XML language.

```
<Rule1>
<method> Association</method>
<probability> 0.75 </probability >
<previous_ws> W1 </previous_ws >
<next_ws> W3 </next_ws >
</Rule1>
<Rule2>
<method> Affinity </method>
<probability> 0.67 </probability >
<previous_ws> W1 </previous_ws >
<next_ws> W4 </next_ws >
</Rule2>
```

Table 6. Semantic knowledge mark-up in the SWRL.

```
W1(?Input1,?Input2,?Ouput2) ⇒ W3(?Input2,?Input3)
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
  <swrlx:individualPropertyAtom swrlx:property="W1">
  <ruleml:var>Input1</ruleml:var>
  <ruleml:var>Input2</ruleml:var>
  <ruleml:var>Ouput2</ruleml:var>
  </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
  <swrlx:individualPropertyAtom swrlx:property="W3">
  <ruleml:var>Input2</ruleml:var>
  <ruleml:var>Input3</ruleml:var>
  </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

SWRL: semantic web rule language.

Simulation and results

Simulation design

In the real world, network or web services are provided by vendors and registered in a registration centre where consumers can find web services to meet their needs. The experiment environment supposed that there are four type groups for web services. All web services in this simulation could be assigned to one of the four groups. These groups include web services with (1) high IO matching and a strong relationship between services, (2) high I/O matching but a weak relationship between services, (3) low I/O matching but a strong relationship between services and (4) low I/O matching and a weak relationship between services. The next section shows the results of an environment with each group equally distributed, implying that the probability of a service belonging to any group is the same.

Table 7. Semantic knowledge.

If W_1 is invoked, then W_3 has an approximately 75% probability of being invoked next.

Table 8. Semantic annotation.

```
<owl:ObjectProperty rdf:ID="ifCondition">
  <rdfs:comment> The if condition of an if-then-else</
rdfs:comment>
  <rdfs:domain rdf:resource="#If-Then-Else"/>
  <rdfs:range rdf:resource="&expr;#W1"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="then">
  <rdfs:domain rdf:resource="#If-Then-Else"/>
  <rdfs:range rdf:resource="#W3"/>
</owl:ObjectProperty>
```

The data set used in the experiments was gathered by a web service discovery and composition challenge contest held by the second IEEE International Conference on e-Business Engineering (2005) in Beijing. Web services discovery and composition are two indispensable capabilities required by emerging SOAs. This competition was limited to syntactical matching based on the WSDL. Participants needed to identify and compose WSDL-specified services based on IO messages as specified in a directory of WSDL documents. As a result, the web service in this data set contains the IO relationships. However, the user profile data are still needed to design the experiments. In this case, the web services amount to 50, and the profiles are increased by 200 for each round.

Simulation results

The experiment compares three experimental methods: (1) the IO method, which uses IO matching to find relationships; (2) the ARs method, which uses ARs to find relationships and (3) the IO + AR method, which combines the IO and AR methods to find the relationships between services. This case was designed to show the performance of this hybrid approach and how this method could improve the recall for service matching with increasing numbers of user profile records.

In this case, as the number of user profiles increases, the precision of the proposed hybrid approach also increases. At first, the precision of the IO approach may perform better than the others because it has some initial clues (IO matching information) to find semantic relationships, while the others do not. Then, the precision of the IO method remains constant because the increasing number of user profiles does not affect the predefined IO relationships. However, with the increase

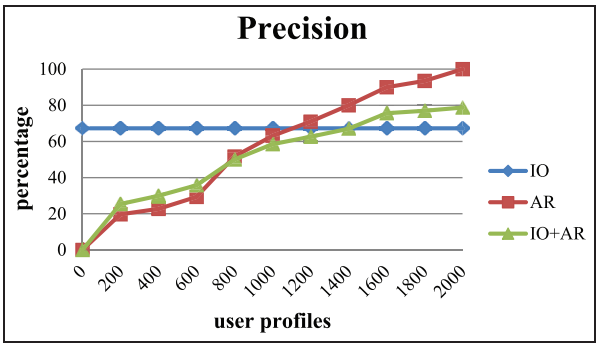

Figure 6. Precision results for the three methods.

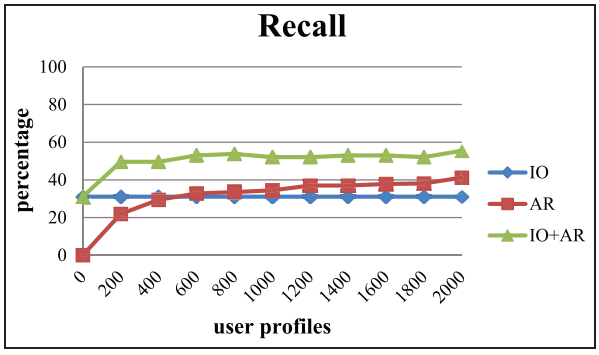

Figure 7. The recall results.

in the number of profile records, the AR and IO + AR methods improve their precision gradually. Eventually, the performances of AR and IO + AR will be better than that of the IO method. Figure 6 shows the precision results for the three methods.

Figure 7 shows the recall results. The IO method's recall result does not change because the increasing number of user profile records does not affect the IO method. For the AR method, more possible correct solutions are gradually found with the increase in the number of user profile records. Because the IO + AR approach combines these two methods, it can retrieve more relationships than the other methods. Therefore, the performance of the recall for the IO + AR method will obviously be better than that of the others. Figure 8 shows the *F*-measure for this case, which demonstrates that the performance of the IO + AR method, is better on average than the IO and AR methods.

The implemented system. This study implements a simple user interface and a convenient way to generate semantic files for web services. From this simple system, a user can obtain an OWL file that includes the service, profile, process and grounding information. Details concerning the OWL files are shown in Figures 9 and 10.

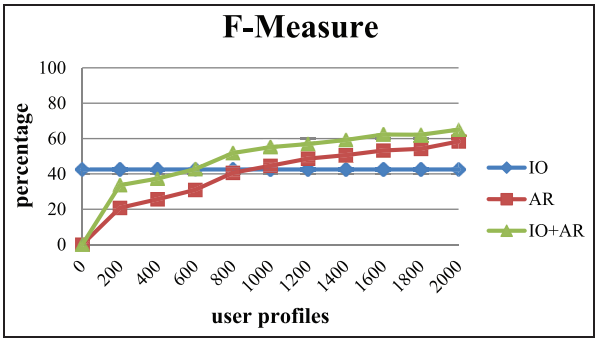

Figure 8. The *F*-measure results.

The *service.owl* file is used to define the service's basic information, such as the service name. This file also defines which profile, process and grounding file linked to. Therefore, with this *service.owl* file, it is easy to find corresponding semantic files that could help with semantic service matching.

The *profile.owl* file defines the functional and non-functional properties. The functional property helps specify what the service provides, such as the input/output/precondition/effects. The *process.owl* files describe how the web service is used. Further, this file defines not only the IO relationships but also the constructor. The *grounding.owl* file describes how the services work, such as the accompanying protocol or port for the channel. With these semantic files, the user can find the semantic relationships, including the input, output and processes, between web services.

Conclusions

In the IoT environment, sensors are used to continually collect data. However, transforming these data into machine-readable and machine-interpretable forms is difficult. Therefore, combining the context of IoT networks with web services and semantic technologies is beneficial for the facilitation of future intelligent environments. OWL-S is often applied to semantic annotation in web services for semantic services matching, selection and composition. However, semantic web services annotation is an extremely complicated task. This study proposes an approach to automatically generate semantic annotation information from semantic knowledge in the OWL-S format. With the proposed method

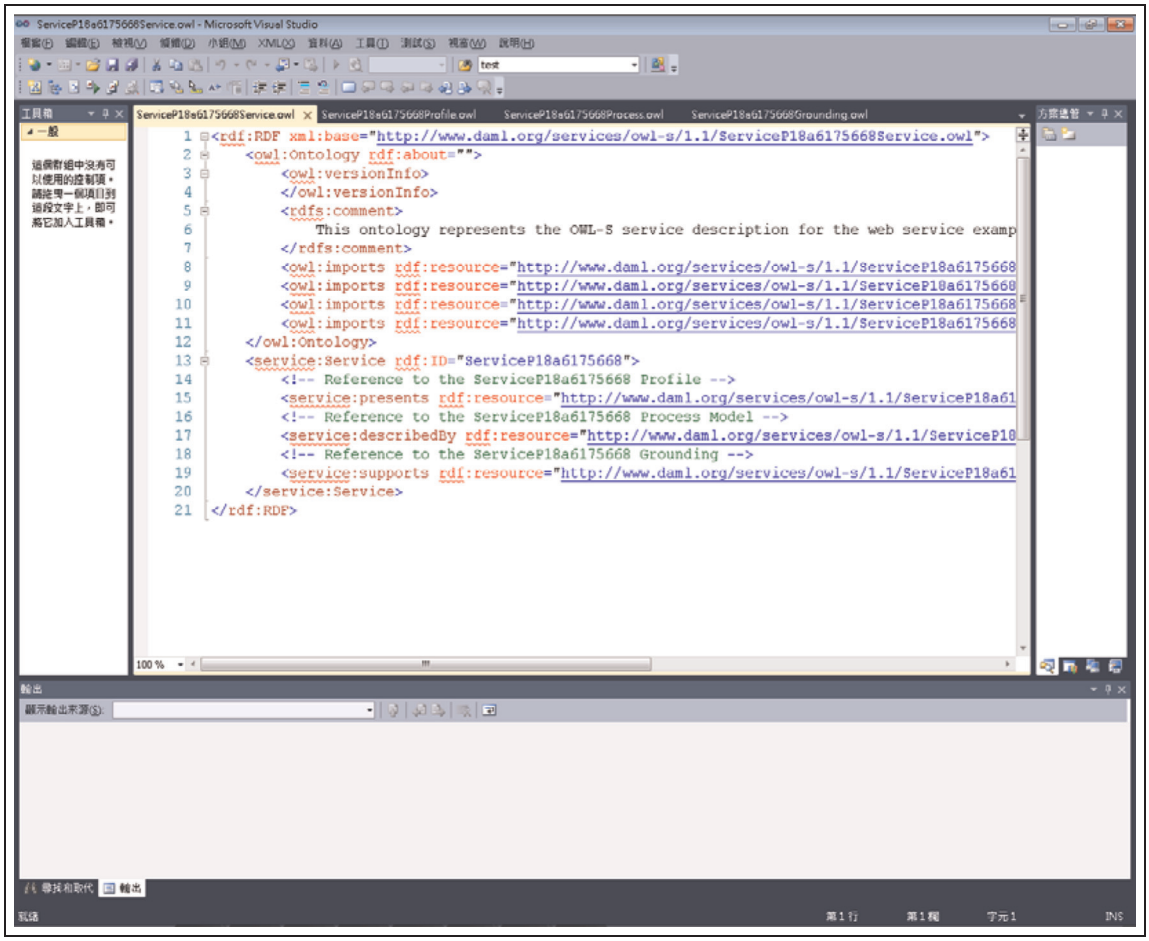

Figure 9. The OWL-S file for the service information. OWL-S: ontology web language for services.

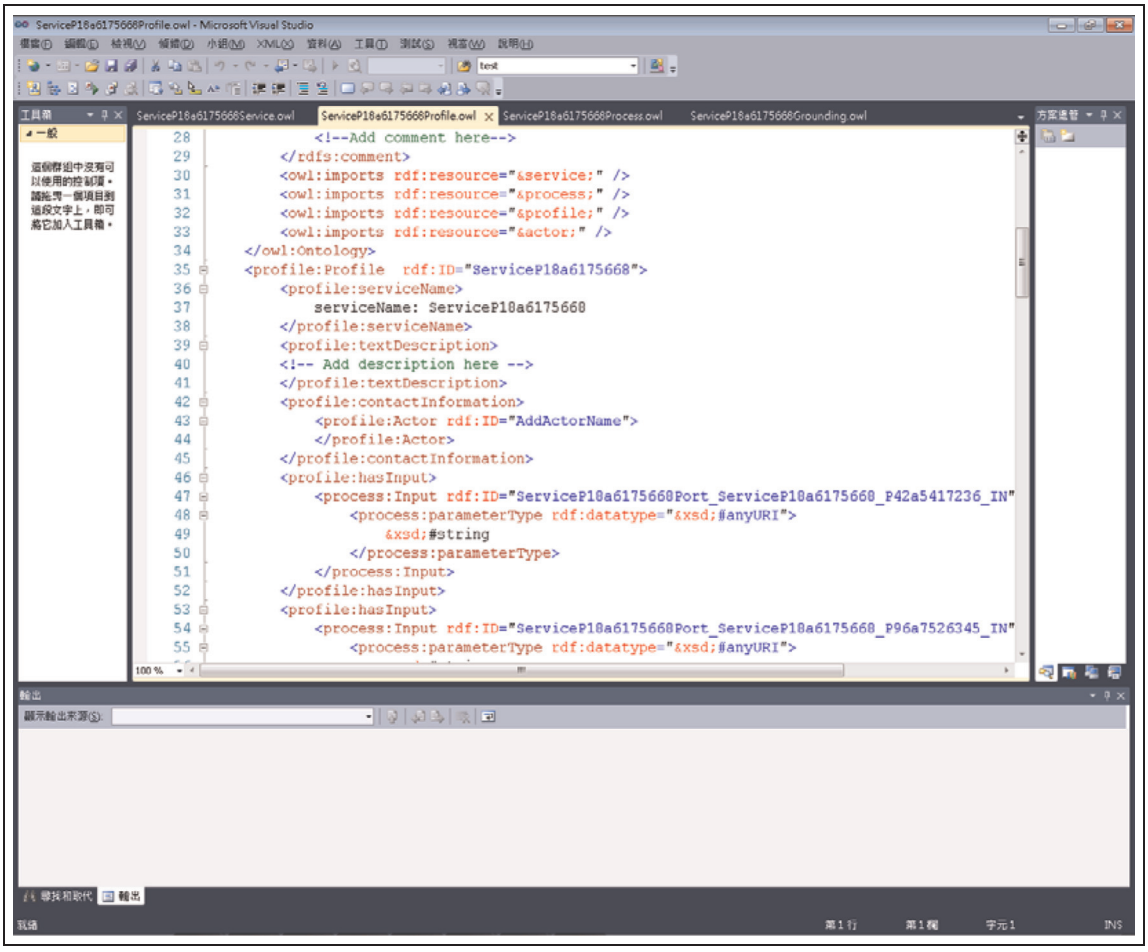

Figure 10. The OWL-S file for the profile information. OWL-S: ontology web language for services.

based on the KDS process, semantic relationships between web services can be extracted by combining ARs and IO matching. Furthermore, the automatic generation of semantic annotation in OWL-S helps to improve the service matching efficiency.

To examine the full potential of the proposed approach in a complex environment, future work will involve measurements and evaluations of this approach in situations involving cloud-based web services. Moreover, other methods, such as cluster or classifier approaches, should be applied to find additional different and complex relationships between services. The higher the number and complexity of these relationships, the higher the scope for knowledge extraction will be; this will help solve the more complex problems of service matching and selection.

Most studies used semantic sensor network (SSN) ontology as a base in the IoT environment, combined with other domain ontologies, as a method to determine relationships between the sensor data. This method is used as a foundation to drive semantic annotation. The SSN ontology can be used along with other ontologies, such as quantity kinds, units ontology and Semantic Web for Earth and Environment Technology (SWEET) ontology. The SSN has also been used with domain ontologies to develop various smart thing ontologies, such as the smart product ontology.[17]

In the future, we may integrate OWL-S and SSN with network and web services to use data mining approaches to analyse all the web sensor data, to find the relationships between the data services and to use these relationships to update and annotate the attribute and the field of the ontology.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research work was supported by the Ministry of

Science and Technology, Taiwan under the grant 104-2410-H-033-025 and 105-2410-H-033-027.

References

1. Delin KA. Sensor webs in the wild. In: *Wireless sensor networks: a systems perspective*, 2005, p. 238. Norwood, MA: Artech House.
2. Delin KA and Jackson SP. Sensor web for in situ exploration of gaseous biosignatures. In: *2000 IEEE Aerospace Conference Proceedings*, Big Sky, MT, March 2000, vol. 7, pp. 465–472.
3. Botts M, Percivall G, Reed C, et al. OGC® sensor web enablement: overview and high level architecture. In: *GeoSensor networks*. Berlin, Heidelberg: Springer, 2006, pp. 175–190.
4. Karnouskos S and Sandro M. Asset monitoring in the service-oriented Internet of things empowered smartgrid. *SOCA* 2012; 6(3): 207–214.
5. W3C. *Web Services Glossary*, 2004, http://www.w3.org/TR/ws-gloss/ (accessed 30 June 2016).
6. Papazoglou M and van den Heuvel W-J. Service oriented architectures: approaches, technologies and research issues. *VLDB J* 2007; 16: 389–415.
7. Papazoglou MP, Traverso P, Dustdar S, et al. Service-oriented computing: state of the art and research challenges. *Computer* 2007; 40: 38–45.
8. Jiang B and Luo Z. A new algorithm for semantic web service matching. *J Softw* 2013; 8(2): 351–356.
9. Ngan LD and Rajaraman K. Semantic web service discovery: state-of-the-art and research challenges. *Pers Ubiquit Comput* 2012; 17(8): 1741–1752.
10. Martin D, Paolucci M and Wagner M. Bringing semantic annotations to web services: OWL-S from the SAWSDL perspective. *Comput Sci* 2007; 4825: 340–352.
11. Tosi D and Sandro M. Supporting the semi-automatic semantic annotation of web services: a systematic literature review. *Inf Softw Technol* 2015; 61: 16–32.
12. Paolucci M, Srinivasan N, Sycara K, et al. Toward a semantic choreography of web services: from WSDL to DAML-S. In: *International Conference on Web Services (ICWS)*, Las Vegas, NV, USA, June 2003.
13. Il-Woong K and Kyong-Ho L. A model-driven approach for describing semantic web services: from UML to OWL-S. *IEEE T Syst Man Cyber, Part C* 2009; 39: 637–646.
14. Blake MB. Knowledge discovery in services. *IEEE Internet Comput* 2009; 13: 88–91.
15. Kun Yue MY, Liu W and Li X. A graph-based approach for type matching in web service composition. *Comput Inf Syst* 2010; 6: 2141–2149.
16. Blake MB and Nowlan ME. Knowledge discovery in services (KDS): aggregating software services to discover enterprise mashups. *IEEE Trans Knowledge Data Eng* 2011; 23: 889–901.
17. Barnaghi P, Wang W, Henson C, et al. Semantics for the internet of things: early progress and back to the future. *Int J Seman Web Inf Syst* 2012; 8(1): 1–21.