PAPER

Specificity-Aware Ontology Generation for Improving Web Service Clustering

Rupasingha A. H. M. RUPASINGHA[†], Nonmember, Incheon PAIK^{†a)}, Member,
and Banage T. G. S. KUMARA^{††}, Nonmember

SUMMARY With the expansion of the Internet, the number of available Web services has increased. Web service clustering to identify functionally similar clusters has become a major approach to the efficient discovery of suitable Web services. In this study, we propose a Web service clustering approach that uses novel ontology learning and a similarity calculation method based on the specificity of an ontology in a domain with respect to information theory. Instead of using traditional methods, we generate the ontology using a novel method that considers the specificity and similarity of terms. The specificity of a term describes the amount of domain-specific information contained in that term. Although general terms contain little domain-specific information, specific terms may contain much more domain-related information. The generated ontology is used in the similarity calculations. New logic-based filters are introduced for the similarity-calculation procedure. If similarity calculations using the specified filters fail, then information-retrieval-based methods are applied to the similarity calculations. Finally, an agglomerative clustering algorithm, based on the calculated similarity values, is used for the clustering. We achieved highly efficient and accurate results with this clustering approach, as measured by improved average precision, recall, F-measure, purity and entropy values. According to the results, specificity of terms plays a major role when classifying domain information. Our novel ontology-based clustering approach outperforms comparable existing approaches that do not consider the specificity of terms.

key words: Web services, Web service clustering, term specificity, ontology learning, service similarity

1. Introduction

With increase of Web services by Internet of Things (IoT) systems, social media, and many life-oriented systems on Web, Web-service clustering contributes to improving quality of service discovery, composition and recommendation significantly. Web-service clustering is used to generate service groups within a large-scale group by considering similar characteristics and functionalities [1], [2]. Existing clustering approaches can be classified in terms of the properties used in the clustering process and include functionally-based [3], nonfunctionally-based [4] and social-criteria-based [5] clustering. This research used functional properties such as input, output, precondition, and effect, nonfunctional properties such as cost, response time

and reliability, and social properties such as sociability in clustering Web services. Here, we propose a new clustering approach based on functional properties, which is the most popular research approach. Our approach is based on extracting the service name, operation name, port name, input message and output message features expressed in Web Services Description Language (WSDL) [6].

Existing clustering approaches use several methods to compute Web-service similarity based on functional properties. These have included Information-Retrieval (IR) methods such as cosine similarity [7], Search-Engine-Based (SEB) methods [8] and Keyword Match [1]. In addition, ontology-based methods such as the Hybrid Term Similarity (HTS) method [8], [9] used an existing ontology or used their own method to generate the ontology. The Context-Aware Similarity (CAS) method [10] used support vector machine (SVM) in its similarity calculations. Existing approaches have provided encouraging results, but they still suffer from several drawbacks. IR-based methods are inadequate for the fine-grained measuring of semantic similarity between services because of the loss of the machine-interpretable semantics. Existing approaches have also failed because of using obsolete knowledge in identifying the latest relationships and words in a service corpus. The HTS method generates an ontology without considering domain-specific information, which plays a key role when classifying information than general terms. The CAS method ignores some term relationships when calculating similarities.

Some existing solutions [8], [9] have proposed ontology-based clustering approaches, but they consider only general terms when generating the ontology. However, domain-specific information is a significant factor in ontology generation and helps to improve the performance of the clustering process. Specific terms have the ability to supply more domain-specific information than do general terms. We propose new clustering approach based on a novel ontology-generation method that uses text-mining techniques such as two types of specific information, namely self-information and context-information. Self-information measures the specificity based on the modifiers in a compound term, which have the ability to describe the domain characteristics included in a term. Context-information can cover areas not addressed by self-information via the composition of multiword terms. For the similarity calculations, we introduce new machine filters by comparing generated

Manuscript received December 7, 2017.

Manuscript revised March 18, 2018.

Manuscript publicized May 18, 2018.

[†]The authors are with School of Computer Science and Engineering, University of Aizu, Aizu-Wakamatsu-shi, 965–8580 Japan.

^{††}The author is with Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka, Belihuloya, Sri Lanka.

a) E-mail: paikic@u-aizu.ac.jp

DOI: 10.1587/transinf.2017EDP7395

ontology relationships. If similarity calculations using these filters fail, we use IR-based methods such as thesaurus-based term similarity and SEB methods. Clustering being achieved via an agglomerative clustering algorithm, based on the cluster centroid method [8], which compares calculated similarity values. Experimental results indicate that our new approach helps to address the issues associated with previous approaches by improving the clustering performance.

The remainder of the paper is organized as follows. In Sect. 2, we explain related work. Section 3 discusses a motivating example involving domain specificity and Sect. 4 describes our proposed approach. Section 5 discusses our experiments and their evaluation. Section 6 concludes the paper and outlines future work.

2. Related Work

2.1 Ontology Learning

In the [8] and [9], ontology is generated via the hidden semantic patterns of the extracted features from the WSDL documents. This is achieved by splitting complex terms into individual words before generating the ontology. C.D. Maio et al. [11] also proposed an ontology generation based on the above two relations. The approach is starting from Web resources, content with a high level of abstraction is obtained: concepts, connections between concepts, and instance-population are identified and arranged into ontology. Fang et al. [12] proposed an agility-oriented and fuzziness-embedded cloud-service ontology model, which adopted agility-centric design. The model enabled comprehensive service specification by capturing cloud concept details and their interactions. In Xie et al. [13], a domain-ontology hierarchy was defined to describe the conceptual semantic information. A domain ontology was developed using semantic dictionaries and existing ontologies from the Internet.

As other approaches, in [14] explained four ontology engineering environments such as OntoEdit, WebODE, Protégé, and Hozo. These tools build ontology using relations between concepts based on role and a more convincing is-a hierarchy. Ojamaa et al. [15] also presented an approach that introduces formal domain ontologies into the Domain Specific Languages (DSL) development process and allows to automatically generate design templates of a DSL meta-model from a given domain ontology. Other research [16] presented four different extensions to the Term Frequency–Inverse Document Frequency (TF-IDF) information retrieval model, that would allow it to be better adapted to specific domains. Here, TF-IDF measure can be viewed as a term specificity measure but only based on document frequency information and cannot apply for the ontology term clustering.

The above mentioned all ontology generation methods based on the formal concepts and super-sub relations of the general terms and did not consider the specificity of

ontology terms. But, our approach generates an ontology based on the specificity of each ontology terms by getting the advantage of the domain-specific information contained in the terms with respect to information theory. It helps to generate the more detailed hierarchy and improve the performance of similarity calculations and service clustering than existing methods.

2.2 Similarity Calculation

Kumara et al. [8] used ontology-based similarity calculations that involved proposed filter values to identify the ontology relationships. By assigning different weights to the filters, they calculated similarity via edge-count-based similarity calculations. Banage et al. [10] proposed new similarity calculations using machine-learning methods such as SVMs. Their method generated feature vectors through extracted terms from Google and Wikipedia. Similarity calculations involved converting the SVM output into posterior probabilities. Our previous approach [9] proposed a new similarity calculation method that combined an ontology-based method [8] and an SVM-based method [10] that gave more-efficient results. Shi et al. [17] acquired the semantic similarity between Web-application-program interfaces and mash-ups by proposing an enhanced cosine-similarity calculation method, where a penalty term for the dissimilarity of two vectors was introduced. Lei et al. [18] proposed both a Web-service similarity measurement method and a recommendation method based on ontology and IR techniques. Their method calculated similarities and classified services according to their topics, functionality and semantics.

2.3 Web-Service Clustering

In previous studies [8], [9], an agglomerative clustering algorithm has been used in the cluster-center identification approach to clustering the Web services. The CAS method uses a spatial clustering technique called the associated keyword space, which was effective for noisy data [10]. It projected the clustering results for a three-dimensional (3D) sphere onto a two-dimensional (2D) spherical surface for 2D visualization. Zhou et al. [19] used unsupervised clustering with K-means clustering on attrition rates to determine an appropriate segmentation and number of segments. The clustering method starts by choosing appropriate attributes. Mabrouk et al. [20] investigated clustering techniques that used a K-means algorithm based on a quality of service (QoS)-aware service selection algorithm (QASSA). It grouped service candidates associated with an activity into several clusters according to their QoS values.

3. Motivating Example Involving Domain Specificity

3.1 Term Specificity and Term Similarity

Research approaches in the last decade have focused on general terms in generating an ontology and did not take

advantage of the domain-specific information contained in the terms. However, the specificity of a term explains the quantity of domain-specific information contained in the term. Specificity can be measured for each term as its included information quantity. Specificity plays a significant role in generating hierarchical relationships between terms [21], and highly specific terms tend to be located at deeper levels of the hierarchy.

In this paper, we introduce a new term-specificity measuring method by classifying information into two categories, namely self-information and context-information, based on the composition of component words. Most terms are compound terms with a set of modifiers, enabling self-information to be significant for representing a set of internal characteristics in a domain corpus. Context-information helps to cover any shortage of self-information. The final specificity value is measured as a combination of self-information and context-information.

3.2 Self-Information

Self-information includes information in compositional words and the internal anatomy of terms. Most domain-specific terms are compound terms, and that helps represent the meaning of terms. The characteristics of each component word and/or the internal anatomy of terms are both useful for measuring term specificity. If the term contains more than one word, the specificity of the term is always larger than that of the head term. With this condition, we can assume that a more specific term has higher specificity.

Let me explain in detailed. Self-specificity is based on a set of compound terms. Consider three terms $t_1 = NovelAuthor$, $t_2 = FictionNovelAuthor$ and $t_3 = ScienceFictionNovelAuthor$ as modifier-head structures. In each term, *Author* represents the head and *Novel*, *Fiction*, and *Science* represent modifiers. t_1 has one modifier, t_2 has two modifiers and t_3 has three modifiers. The compound term t_1 is created by adding the modifier *Novel* to the existing word *Author*. Here, *Author* is considered an ancestor of *Novel*. The meaning of the compound term t_1 can be predicted by using the two compounding words *Novel* and *Author* that describe their unique characteristics. t_2 is created by adding a new modifier *Fiction* to the term t_1 and t_3 is created by adding a new modifier *Science* to the term t_2 . In this manner, as the number of modifiers is increased, the compound term can achieve a more specific meaning. Therefore, multiword terms have a higher specificity value than single-word terms. The specificity of the terms t_1 , t_2 and t_3 are ordered as $Spec(t_1) < Spec(t_2) < Spec(t_3)$.

3.3 Context-Information

Some information cannot be accessed by self-information derived from the composition of multiword terms. Some terms have the ability to describe their own characteristics independently without sharing common words, such as $t_1 = FilmInformation$ and $t_2 = MovieDetails$. Consider

these two terms t_1 and t_2 , which do not share any common words. These new terms would have been created independently of existing terms. In this word format, it is ineffective to measure the specificity values using self-information because the compounding words of t_1 and t_2 are completely different from each other. In such cases, assessing compound words independently cannot give the correct information for the compound term.

These limitations can be overcome using context-information that represents the characteristics of the terms indirectly. General terms are usually modified by other words, but domain-specific terms are rarely modified by other words because they already contain sufficient information [21]. Using this idea, we can use the probabilistic distribution of modifiers as context-information to measure the specificity of terms [21].

Based on this theory of information and basic similarity calculation, we propose a new method to improve the accuracy of the ontology hierarchy, Web-service similarity calculations and the clustering procedure.

4. Proposed New Approach

The architecture of the proposed approach is shown in Fig. 1 and it contains five phases namely feature extraction, domain specificity weight and similarity weight calculation, ontology generation, similarity calculation and Web service clustering [22].

4.1 Feature Extraction

Real-world Web-service repositories and the OWL-S test collection were used as the services dataset for the WSDL documents related to five domains, namely *Vehicle*, *Medical*, *Film*, *Food* and *Book*. We extracted five features from WSDL documents, namely service name, operation name, port name, input message and output message. Each extracted term was used as an ontology node without splitting into words. For the specificity and similarity calculations,

Fig. 1 Architecture of the new approach

Fig. 2 Example of a service corpus

we split each complex term into individual words based on several assumptions.

The specificity and similarity weights depend on the set of all domain-related words contained in the service corpus. Here, service corpus included all extracted features from the WSDL documents. The accuracy of the ontology generation can be improved by adding more domain-related terms to the service corpus. We added more domain-specific terms from the Google by extracting frequently used terms in the particular domains related to five domains using the TF-IDF calculation procedure.

4.2 Domain-Specificity Weight and Similarity Weight Calculations for Ontology Generation

4.2.1 Domain-Specificity Weight for Ontology Generation

4.2.1.1 Self-Specificity Value

As described in Sect. 3, calculation of self-specificity by self-information is based on a set of compound terms. Figure 2 shows an example service corpus with frequencies of compound terms and their component words found in the service corpus.

Let $T = \{t_1, t_2, \dots, t_N\}$ be a set of terms (N is the number of terms) that were found in a service corpus, and $A = \{a_1, a_2, \dots, a_M\}$ be a set of atomic words (M is the number of atomic words) that compose the terms in T . Term frequency in the service corpus n_{t_i} is to be calculated to all the terms in T , and n_{a_j} is to be calculated to all the atomic words. And we set $N_T = \sum_{t_i \in T} n_{t_i}$ and $N_A = \sum_{a_j \in A} n_{a_j}$. In Fig. 2 we can count as $N_T = 8$ and $N_A = 21$.

If a term t_i is found in the service corpus, the information quantity of the event of t_i is observed by $I(c_i)$ and can be calculated via information-theoretic methods [21]. Based on this, the specificity of t_i is assigned as the following (1).

$$Spec(t_i) = I(c_i) \quad (1)$$

To calculate the specificity, the joint probability distribution $P\{c_i, d_j\}$ is given by $c_i \in C$ and $d_j \in D$. We can assume c_i for selecting a term from T and d_j for selecting a word from A , where events $\{c_1, c_2, \dots, c_N\}$ and $\{d_1, d_2, \dots, d_M\}$ are illustrated in the random variables C and D . D_{t_i} is a set of d_j that are associated with the words A_{t_i} . The mutual information between c_i and d_j compares the probability of observing c_i and d_j together and independently, as given by (2).

$$I(c_i, d_j) = \log \frac{P(c_i, d_j)}{P(c_i)P(d_j)} \quad (2)$$

The specificity of term t_i is represented by the value $I(c_i, D)$, which indicates the mutual information between c_i and D . $I(c_i, D)$ is estimated using the frequency of terms and words in a service corpus according to the following Eq. (3).

$$Spec(t_i) \approx I(c_i, D)$$

$$= \sum_{d_j \in D_{t_i}} P(c_i, d_j) \log \frac{P(c_i, d_j)}{P(c_i)P(d_j)}$$

Here, $P(c_i|d_j) = \frac{P(d_j|c_i)P(c_i)}{P(d_j)}$ by Bayes' theorem,

$$= \sum_{d_j \in D_{t_i}} P(c_i|d_j)P(d_j) \log \frac{P(c_i|d_j)}{P(c_i)}$$

$$\approx \sum_{a_j \in A_{t_i}} \frac{(n_{a_j} \cdot t_i)}{n_{t_i}} \frac{n_{t_i}}{N_T} \log \frac{(n_{a_j} \cdot t_i)}{n_{t_i}} \frac{N_A}{n_{a_j}}$$

$$SelfSpec(t_i) \approx \frac{1}{N_T} \sum_{a_j \in A_{t_i}} \left(\alpha \cdot \log \frac{N_A}{n_{t_i} \cdot n_{a_j}} \right) \quad (3)$$

Extracted terms from WSDL documents contain one or two words, according to the normal format of WSDL. Because of this consideration, $(n_{a_j} \cdot t_i)$, the number of the words in t_i , is assumed to be 1. The weighting scheme for the specificity of the modifier represented by α is based on linguistic knowledge [21]. Because of this consideration and from experimental results, $\alpha = 1$ is selected from the range $0 \leq \alpha \leq 1$.

4.2.1.2 Context-Specificity Value

Context-specificity is based on the entropy of the probabilistic distribution of modifiers for a term [21].

$$H_{mod(t_i)} = - \sum_{1 \leq m \leq F} P(mod_m, t_i) \log P(mod_m, t_i) \quad (4)$$

Here, F is the number of modifiers of t_i . The probability that mod_m modifies t_i is given by (mod_m, t_i) . The relative frequency of (mod_m, t_i) in all (mod_f, t_i) pairs in a service corpus is estimated for $1 \leq f \leq F$. The entropy value is given by the average information in all (mod_m, t_i) pairs. Because domain-specific terms have simple modifier distributions, specific terms have low entropy. Therefore, the result of (4) is converted as an inverse entropy and assigned to $I(c_i)$, as given by (5), giving a large quantity of information [21].

$$ConSpec(t_i) \approx I(c_i) \approx \max_{1 \leq j \leq K} H_{mod(t_j)} - H_{mod(t_i)} \quad (5)$$

Here, K includes number of all modifiers with the same head and $H_{mod(t_j)}$ is used for each modifier.

4.2.1.3 Hybrid Specificity Value

The two methods just described are powerful tools for calculating specificity values. Self-specificity helps to cover the component words' characteristics, and context-information addresses areas that cannot be handled by self-information.

We therefore combine the results from (3) and (5) to form a hybrid specificity as given by (6) that takes advantage of both methods:

$$HySpec(t_i) = \frac{1}{\beta \left(\frac{1}{SelfSpec(t_i)} \right) + (1-\beta) \frac{1}{ConSpec(t_i)}} \quad (6)$$

Here, β is in the range $0 \leq \beta \leq 1$ and was given the value 0.7 by experimentation. Because of the normalization, all results for $SelfSpec(t_i)$, $ConSpec(t_i)$ and $HySpec(t_i)$ were between 0 and 1.

4.2.1.4 Domain-Specificity Weight Value for Ontology Generation

The optimal ontology structure is based on the domain-specificity weight, which is calculated using sibling terms. Here we used the theoretical substructure, with sibling terms of similar specificity being assigned a higher score. The domain-specificity weight of a substructure is calculated using (7), as follows:

$$W_{Spec}(G_i) = \begin{cases} 1 - \frac{\sum_{t_s \in F_{t_s}} |Spec(t_s) - Spec(t_{new})|}{|F_{t_s}|} \\ 0.5 \end{cases} \quad (7)$$

Here, F_{t_s} is the number of sibling terms of the new term t_{new} and t_s is a sibling in G_i . $Spec(t_s)$ describes the specificity of each sibling term and $Spec(t_{new})$ describes the specificity of new term. If F_{t_s} is empty, $W_{Spec}(G_i)$ is directly assigned as 0.5 from experimentation.

4.2.2 Term Similarity Weight for Ontology Generation

4.2.2.1 Term Similarity Value for Ontology Generation

Here, we use the basic similarity calculation procedure by comparing the words common to two terms.

$$Sim(t_x, t_{new}) = \frac{2 F_{(t_x, t_{new})}}{|F_{t_x}| + |F_{t_{new}}|} \quad (8)$$

Here, $F_{(t_x, t_{new})}$ is the number of common words of t_x and t_{new} . F_{t_x} and $F_{t_{new}}$ describe the number of compositional words in each term. This result is used to calculate the similarity weight values, as given by (9) below.

4.2.2.2 Term Similarity Weight Value for Ontology Generation

Here, we assume that a substructure with more similar terms clustered around the new term will have a higher score. We use all the terms connected to the new term, which are parent, child and sibling terms, as more-similar terms. The similarity weight value is calculated by using the results from (8) to produce (9). Here, K is the total number of parent, sibling and child terms. t_x is a term, with $t_x \in K$.

$$W_{Sim}(G_i) = \frac{\sum_{t_x \in K} |Sim(t_x, t_{new})|}{|K| - 1} \quad (9)$$

Fig. 3 Target area of the existing ontology

Fig. 4 Screenshot of part of a generated ontology

4.3 Ontology Generation

Here, the ontology-generating procedure considers each term's relations in the ontology and depends on the calculated specificity weight (7) and similarity weight (9). This is a top-down approach that builds the hierarchy by starting from the top root node and adding other nodes one by one to the current hierarchy.

Step 1. The calculated hybrid specificity values are arranged in ascending order and the first three values are selected as the first three nodes for starting the ontology generation procedure.

Step 2. The terms are then added to the ontology hierarchy in ascending order. Target substructures for a t_{new} to be combined as a new term, as shown in Fig. 3, are selected based on the hybrid specificity values of the new node and the existing nodes of the ontology.

Step 3. With this set of target nodes, we can identify a set of candidate substructures.

Step 4. The optimal substructure is found by calculating the specificity weight (7) and similarity weight (9) for each candidate substructure associated with the t_{new} and finding the maximum $W_{final}(G_i)$ by combining them.

$$W_{final}(G_i) = \gamma \cdot W_{Spec}(G_i) + (1 - \gamma) \cdot W_{Sim}(G_i) \quad (10)$$

Here, γ was assigned as 0.4 through experimentation. This procedure iterates until the ontology hierarchy is complete, with all terms added in the prepared order. Figure 4 shows a screenshot of part of a generated ontology that contains terms from the *vehicle* and *book* domains. They are separated automatically into hierarchies according to the

domains. Algorithm 1 describes the ontology generation procedure.

4.4 Service-Similarity Calculation in an Ontology

To measure the similarity of two Web services, we first take the relevant extracted features from the two WSDL files. Then match these two terms via the generated ontology without splitting the words. There are two possibilities: they can be identical or they can differ. If two terms match exactly, we assign the term to the extract filter (see below) and give it the highest similarity value, i.e., 1. If there is no exact match in the ontology, we define a procedure for similarity calculation involving six different filters, weighted toward different relationships. If we find that the two extracted terms in our generated ontology satisfy one of our defined filters, then a similarity calculation is performed using the appropriate equation.

We now describe the machine filters we used in the similarity calculations. They are used to compute the degree

Algorithm 1: Ontology generation

```

Input  $I_1$  =: Array of WSDL extracted terms
Input  $I_2$  =: Array of Google search engine extracted terms
Output  $O_s$  : Ontology
1: For  $I_1$  and  $I_2$  do
2:    $I$  = Array of ( $I_1 + I_2$ ) ascending order; //create an array
   using all terms according to the hybrid specificity
3: end-for
4:  $I_s$  = Select 1st three nodes from  $I$ ;
5: Start ontology  $O_s$  with  $I_s$ ;
6: For ontology  $O_s$  do
7:   For each remaining term  $t_{new}$  in  $I$  do
8:     if  $(Spec(t_{new}) - 0.3 < Spec(\text{existing nodes of ontology}$ 
        $(T_n)) < Spec(t_{new}) + 0.3)$ 
9:       select target nodes as  $T_n$ ; //target area nodes will be
       selected
10:    end
11:    For each  $T_n$  in  $O_s$  do
12:      add  $t_{new} \rightarrow G_i$  = candidate substructures; // set of
       candidate substructures will be generated
13:       $W_{Spec}(G_i)$  = Calculate domain-specificity weight;
14:       $W_{Sim}(G_i)$  = Calculate similarity weight;
15:       $W_{final}(G_i)$  = Calculate final maximum weight;
16:       $O_s$  = Substructure of maximum weight
       (from  $W_{final}(G_i)$ ); //select optimal substructure
17:    end-for
18:  end-for
19: end-for

```

of semantic similarity for a pair of services. Figure 5 shows an example ontology that contains nine terms describing nine WSDL files for one feature among five features. As a Table 1, to investigate the strength of the defined filters, we conducted experiments and assigned weight values for each filter in the following order, based on the strength in logic-based matching: Exact > Siblings > Parent–Child > Near-Descendants > Shared-Ancessor > Far-Descendants > Fail. We use Eqs. (11) and (12) below with the relevant weight values to calculate the similarity:

$$S_{Onto}(t_1, t_2) = -\log \frac{d(t_1, t_2)}{2D} \quad (11)$$

This calculation is an edge-count-based method. Here, $d(t_1, t_2)$ describes the shortest distance between the two terms t_1 and t_2 , and D describes the maximum depth of the generated ontology.

$$Sim_F(t_1, t_2) = W_1 + W_2 S_{Onto}(t_1, t_2) \quad (12)$$

Here, $S_{Onto}(t_1, t_2)$ is assigned using the results from (11) and W_1 and W_2 are assigned according to the relevant filter values in Table 1. For normalization purposes in the similarity values, we select values that satisfy $W_1 + W_2 = 1$. Further, the final similarity value is between 0 and 1.

If the two extracted terms do not satisfy any of the defined filters, then we use an IR-based method to calculate the similarity [8], [9].

4.5 Web-Service Clustering

We used an agglomerative clustering algorithm based on the cluster center method using TF–IDF values for Web service clustering that introduced before [8], [9]. Finally, the services are grouped into five different clusters, namely *Food*, *Book*, *Medical*, *Film* and *Vehicle*.

Fig. 5 Example of ontology hierarchy

Table 1 Assigned matching filters and weights with examples

Filter	Description	Example	Weight	
			W_1	W_2
1-Exact	If term t_i and term t_j are the same and represent the same feature, then the services exactly match.	$t_2 = a_2a_1, t_2 = a_2a_1$	Similarity=1	
2-Siblings	Term t_i and term t_j plug into term t_p with $t_i \in DirectChildren(t_p)$ and $t_j \in DirectChildren(t_p)$.	$t_2 = a_2a_1, t_3 = a_3a_1$	0.9	0.1
3-Parent–Child	Term t_i plugs into term t_j with $t_i \in DirectChildren(t_j)$.	$t_2 = a_2a_1, t_1 = a_1$	0.8	0.2
4-Near-Descendants	Term t_c plugs into term t_i and term t_j plugs into term t_c with $t_c \in DirectChildren(t_i)$ and $t_j \in DirectChildren(t_c)$.	$t_2 = a_2a_1, t_6 = a_5a_4a_2a_1$	0.78	0.22
5-Shared-Ancessor	Term t_i and term t_j plug into a child term of term t_q with $t_q \in Ancestor(t_i)$ and $t_q \in Ancestor(t_j)$.	$t_4 = a_4a_2a_1, t_5 = a_4a_3a_1$	0.65	0.35
6-Far-Descendants	Term t_i and term t_j have a far-descendants relationship with $t_j \in FarAncestor(t_i)$.	$t_1 = a_1, t_8 = a_5a_4a_3a_1$	0.62	0.38
7-Fail	If none of the other filters generates a match, then there is a fail.	Used IR-based methods		

5. Experiments and Evaluation

The experimental platform used Microsoft Windows 10 on a PC with an Intel Core i7-6500 at 2.59 GHz and 8.00 GB of RAM. Java was used for programming the ontology generation and the service clustering procedure. The generated ontology was displayed visually by using JTree. In each experiment, we used a set of extracted features from 400 WSDL files. Performance evaluation of the clustering results involved with the standard measures of retrieval information such as precision, recall, F-measure, purity and entropy in a comparison to previous approaches (Note that the “entropy” measure here is totally different from that defined in Eq. (4)).

5.1 Evaluation of Ontology Generation

When adding a new term to a partially generated ontology, we first need to select a suitable target area of nodes. Expanding the target area will provide opportunities for more nodes to be tested and will indicate the most suitable nodes for adding the new term. Figure 6(a) shows the different results for precision, recall and F-measure for two different target areas in an existing ontology. Target area-1 ranged from $Spec(t_{new}) - 0.1$ to $Spec(t_{new}) + 0.1$ and Target area-2 ranged from $Spec(t_{new}) - 0.3$ to $Spec(t_{new}) + 0.3$. Expanding the target area will also increase the computation time for the program. We chose to use a target area in the existing ontology that ranged from $Spec(t_{new}) - 0.3$ to $Spec(t_{new}) + 0.3$ because this would help to improve the performance with respect to clustering results. In addition, we found that we could improve the ontology performance by adding more domain-specific terms to it. Adding frequently used terms extracted from the Google search engine resulted in better performance as Fig. 6(b). Next, we compared the number of nodes in our new ontology with a previous HTS-based

ontology [8]. As shown in Fig. 6(c), the previous method contains more ontology nodes, which means that complex terms were more likely to be divided into individual terms in generating their ontology. Because we generate our ontology directly using the original terms, the new method will contain fewer nodes.

When generating our ontology we used fine-grained information of Web service data. If we apply for the general text we can get the general ontology, but it will be complex and still, we did not apply for that. And also our generated domain ontology can be applied for other general purposes rather than Web service clustering.

5.2 Evaluation of Domain-Specificity Weight and Similarity Weight Calculations

Figure 6(d) shows the variation in the total number of words (N_A) and the total number of terms (N_T) as the number of services varies from 100 to 400. Equation (3) uses these values in the calculation of self-specificity. The total numbers of words and terms both increased as the number of services increased. Note that the rate of increase for words is greater than the rate of increase for terms as the number of Web services increases. Then to show the impact of purity and entropy for the self-specificity calculation in Eq. (3), we varied the value of α from 0.8 to 1.2. Figure 6(e) shows that optimal purity and entropy values occur when $\alpha = 1$. Next, we measured the purity and entropy values while changing the β parameter in the hybrid specificity calculation (see Eq. (6)). It gives optimal results for purity and entropy when $\beta = 0.7$ within the range 0.5 to 0.9, as shown in Fig. 6(f). We can note that self-specificity makes a higher contribution to the final hybrid specificity value than does context-specificity.

The domain-specificity weight of a substructure, as calculated in Eq. (7), is dependent on the number of sibling terms of the new term t_{new} . If t_{new} has no sibling terms,

Fig. 6 Evaluation of ontology generation, domain-specificity weight and similarity weight calculations

Table 2 Performance measures for clusters using precision, recall and F-measure

Cluster	Edge-Count-Based (WordNet) (%)			HTS Approach (%)			CAS Approach (%)			New Approach (%)		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Vehicle	56.00	80.90	66.20	81.60	94.80	87.70	89.47	89.47	89.47	90.80	91.86	91.33
Medical	100.00	70.00	82.40	100.00	83.10	90.80	88.10	100.00	93.67	94.64	84.13	89.08
Food	55.00	60.00	57.40	96.00	91.30	93.60	85.71	93.10	89.26	92.86	96.30	94.55
Book	67.10	61.30	64.10	86.70	100.00	92.90	82.14	92.00	86.79	90.59	96.25	93.33
Film	77.70	50.00	60.80	91.00	88.00	89.50	87.50	84.85	86.15	94.20	95.59	94.89
Average	71.16	64.44	66.18	91.06	91.44	90.90	86.59	91.89	89.07	92.62	92.82	92.63

then we assign a fixed value of 0.5 for the domain specificity weight. It was found by experimentation with values from 0.4 to 0.7 that 0.5 gives optimal results for purity and entropy as Fig. 6 (g). Then Fig. 6 (h) shows variations in the parameter γ for the final calculation (see Eq. (10)) of the domain-specificity weight and similarity weight combination that we used to select the optimal ontology structure with the highest weight value. We assigned $\gamma = 0.4$ from the range 0.3 to 0.7, given the high contribution from similarity weight.

5.3 Evaluation of Web-Service Clustering

Five different domains were considered, namely *Vehicle*, *Medical*, *Film*, *Food* and *Book*. We compared with previous approaches such as an edge-count-based method, an HTS approach [8], a CAS approach [10] and our new approach.

When comparing clustering results of previous approaches, we can see that our approach group services more correctly by taking the advantage of term specificity. Following services are examples of wrong cluster groups in the previous HTS method.

Film: *BookTaxedprice*, *RawFoodPrice*, *AuthorBook*, *Science-fiction-novelAuthorBook-type*

Book: *PreparedFoodPrice*

Medical: *VehiclePrice*, *FourwheeledCarYearprice*

Here explain three cluster groups of *Vehicle*, *Medical*, *Film*, *Food*, and *Book* five clusters. It contains some example Web services that are put on the wrong cluster. However, when compared with the new method, it successfully groups them in the correct cluster. Because, new method took the advantage of the specificity calculation method and successfully identify *Book*, *Food*, *Vehicle* and *Car* like key terms in the Web services.

Table 2 gives the experimental results, comparing precision, recall and F-measure values. From these clustering results, the best cluster performance was achieved by our new approach, which placed services correctly for more of the clusters than did the other methods. Our new approach offered improvements in the average precision values of 21.46%, 1.56% and 6.03%, the average recall values of 28.38%, 1.38% and 0.94 %, and the average F-measure values of 26.45%, 1.73% and 3.57%, over those for the edge-count-based, HTS and CAS methods, respectively. In fact, all results for the new approach exceed 84%. Based on these clustering results, we can note that extracting features from WSDL documents alone is insufficient to identify the correct cluster for terms.

Fig. 7 Cluster performance of new and existing approaches

Figure 7 (a) and Fig. 7 (b) show the variation in purity and entropy values, respectively, as the number of Web services increases from 100 to 400. As the number of Web services increases, our approach gives increasingly better results than previous approaches in terms of purity decrease and entropy increase. From these results, the new approach gives better accuracy for a high number of inputs. In addition, our new approach gave lower entropy and higher purity values throughout, with the rate of purity-value decrease and the rate of entropy-value increase both being smaller. This comparative study of alternative approaches supports the validity of the proposed approach.

6. Conclusions and Future Work

In this paper, we have proposed a new domain-specificity-based ontology-generation method, with Web-service clustering being achieved via similarity calculations based on the generated ontology relationships. New machine filters are proposed for the similarity calculations that compare ontology relationships. This new approach is expected to help improve the clustering or recommendation performance of Web services.

The new approach takes advantage of the information in specific terms instead of relying on more-general terms. Specific terms are more significant than general terms when classifying domain-related information. Previous approaches have focused on general terms and have not taken advantage of specific terms. Our measurements of specificity values showed that we could achieve higher accuracy. Furthermore, according to our experimental results, our new information-theory-based approach gave improved validity and accuracy when compared with previous methods such as the edge-count-based, HTS and CAS approaches. We achieved superior results in terms of precision, recall, F-measure, entropy and purity.

Our future research will focus on improving the clus-

tering performance of Web services by considering different features and using new ontology generation methods. We also hope to be able to investigate other significant aspects of service discovery and recommendation.

References

- [1] K. Elgazzar, A.E. Hassan, and P. Martin, "Clustering WSDL documents to bootstrap the discovery of Web services," IEEE International Conference on Web Services (ICWS 2010), Miami, Florida, USA, July 5–10, 2010, pp.147–154, 2010. doi:10.1109/ICWS.2010.31
- [2] L. Chen, L. Hu, Z. Zheng, J. Wu, J. Yin, Y. Li, and S. Deng, "WTCluster: Utilizing tags for Web services clustering," Service-Oriented Computing, pp.204–218, Springer, 2011. doi: 10.1007/978-3-642-25535-9-14
- [3] S. Dasgupta, S. Bhat, and Y. Lee, "Taxonomic clustering and query matching for efficient service discovery," 9th IEEE International Conference on Web Services, Washington, DC, pp.363–370, July 4–9, 2011. doi:10.1109/ICWS.2011.112
- [4] Y. Xia, P. Chen, L. Bao, M. Wang, and J. Yang, "A QoS-aware Web service selection algorithm based on clustering," 9th IEEE International Conference on Web Services, Washington, DC, pp.428–435, July 4–9, 2011. doi:10.1109/ICWS.2011.36
- [5] W. Chen, I. Paik, and P.C.K. Hung, "Constructing a global social service network for better quality of web service discovery," IEEE Transactions on Services Computing, vol.8, no.2, pp.284–298, March–April, 2015. doi:10.1109/TSC.2013.20
- [6] Web Services Description Language (WSDL) 1.1, <https://www.w3.org/TR/wsdl> (accessed 03.03.2017).
- [7] J. Ma, Y. Zhang, and J. He, "Efficiently finding Web services using a clustering semantic approach," Proc. International Workshop on Context-enabled Source and Service Selection, Integration and Adaptation, Organized with the 17th International World Wide Web Conference, Beijing, China, pp.1–8, April 22, 2008. doi:10.1145/1361482.1361487
- [8] B.T.G.S. Kumara, I. Paik, W. Chen, and K.H. Ryu, "Web Service Clustering using a Hybrid Term-Similarity Measure with Ontology Learning," International Journal of Web Services Research (IJWSR), vol.11, no.2, pp.24–45, 2014. doi: 10.4018/ijwsr.2014040102
- [9] R.A.H.M. Rupasingha, I. Paik, and B.T.G.S. Kumara, "Calculating Web service similarity using ontology learning with machine learning," 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), IEEE, pp.1–8, 2015. doi: 10.1109/ICCIC.2015.7435686
- [10] B.T.G.S. Kumara, I. Paik, H. Ohashi, Y. Yaguchi, and W. Chen, "Context-Aware Web Service Clustering and Visualization," International Journal of Web Services Research, 2014.
- [11] C.D. Maio, G. Fenza, M. Gallo, V. Loia, and S. Senatore, "Formal and relational concept analysis for fuzzy-based automatic semantic annotation," Applied intelligence, vol.40, no.1, pp.154–177, 2014.
- [12] D. Fang, X. Liu, I. Romdhani, P. Jamshidi, and C. Pahl, "An agglutination-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation," Future Generation Comp. Syst., vol.56, pp.11–26, 2016. doi: 10.1016/j.future.2015.09.025
- [13] L.-L. Xie, F.-Z. Chen, and J.-S. Kou, "Ontology-based semantic Web services clustering," Proc. 18th IEEE International Conference on Industrial Engineering and Engineering Management, pp.2075–2079, 2011. doi: 10.1109/ICIEEM.2011.6035578
- [14] R. Mizoguchi and K. Kozaki, "Ontology engineering environments," Handbook on Ontologies, pp.315–336, Springer, Berlin, Heidelberg, 2009.
- [15] A. Ojamaa, H.-M. Haav, and J. Penjam, "Semi-automated generation of DSL meta models from formal domain ontologies," Model and Data Engineering, pp.3–15, Springer, Cham, 2015.
- [16] C. Fautsch and J. Savoy, "Adapting the tf idf vector-space model to domain specific information retrieval," Proc. 2010 ACM Symposium on Applied Computing, pp.1708–1712, ACM, 2010.
- [17] M. Shi, J. Liu, D. Zhou, M. Tang, F. Xie, and T. Zhang, "A Probabilistic Topic Model for Mashup Tag Recommendation," 2016 IEEE International Conference on Web Services (ICWS), IEEE, pp.444–451, June 2016. doi: 10.1109/ICWS.2016.64
- [18] Y. Lei, Z. Jiantao, Z. Junxing, W. Fengqi, and W. Juan, "Time-aware semantic Web service recommendation," IEEE International Conference on Services Computing, New York, pp.664–671, 2015. doi: 10.1109/SCC.2015.95
- [19] N. Zhou, W.M. Gifford, J. Yan, and H. Li, "End-to-End Solution with Clustering Method for Attrition Analysis," 2016 IEEE International Conference on Services Computing (SCC), pp.363–370, 2016. doi:10.1109/SCC.2016.54
- [20] N.B. Mabrouk, N. Georgantas, and V. Issarny, "Set-based bi-level optimization for qos-aware service composition in ubiquitous environments," 2015 IEEE International Conference on Web Services (ICWS), pp.25–32, June 2015. doi: 10.1109/ICWS.2015.14
- [21] P. Buitelaar, "An information-theoretic approach to taxonomy extraction for ontology learning," Ontology Learning from Text: Methods, Evaluation and Applications, Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdam, vol.123, p.15, July 2005.
- [22] R.A.H.M. Rupasingha, I. Paik, and B.T.G.S. Kumara, "Improving Web Service Clustering through a Novel Ontology Generation Method by Domain Specificity," 2017 IEEE International Conference on Web Services (ICWS), pp.744–751, IEEE, 2017.
- [23] WordNet-A lexical database for English, <https://wordnet.princeton.edu/> (accessed 06.02.17).

Rupasingha A. H. M. Rupasingha received the bachelor's degree in 2013 from Sabaragamuwa University of Sri Lanka. She received the master's degree in 2016 from School of Computer Science and Engineering, University of Aizu, Japan and now she is a Ph.D student in School of Computer Science and Engineering, University of Aizu, Japan. Her research interests include Web Service Clustering, Ontology Learning and recommendation.

Incheon Paik received his master's and doctoral degrees in Electronics Engineering from Korea University in 1987 and 1992, respectively. Now he is a full professor in the University of Aizu, Japan. His research interests include Semantic Web, Web Services and Their Composition, Web Data Mining, Awareness Computing, and Agents on Semantic Web. He served several conferences as a program chair. He also serves as an editor of journals of JIPS and IEICE.

Banage T. G. S. Kumara received the bachelor's degree in 2006 from Sabaragamuwa University of Sri Lanka. He received the master's degree in 2010 from University of Peradeniya, Sri Lanka and he received the Ph.D from School of Computer Science and Engineering, University of Aizu, Japan in 2015. His research interests include Semantic Web, Web Data Mining, Web Service Discovery and Composition.