IMPROVING WEB SERVICE CLUSTERING THROUGH POST FILTERING TO BOOTSTRAP THE SERVICE DISCOVERY

Banage T. G. S. Kumara, Incheon Paik, Koswatte R. C. Koswatte, Wuhui Chen
School of Computer Science and Engineering
University of Aizu, Aizu-Wakamatsu, Fukushima, Japan
btgsk2000@gmail.com, paikic@u-aizu.ac.jp, {ckoswatte, chenwuhui21}@gmail.com

Abstract
Web service clustering is one of a very efficient approach to discover Web services efficiently. Current approaches use similarity-distance measurement methods such as string-based, corpus-based, knowledge-based and hybrid methods. These approaches have problems that include discovering semantic characteristics, loss of semantic information, shortage of high-quality ontologies and encoding fine-grained information. Thus, the approaches couldn't identify the correct clusters for some services and placed them in wrong clusters. As a result of this, cluster performance is reduced. This paper proposes post-filtering approach to increase the performance of clusters by rearranging services incorrectly clustered. Our approach uses context aware similarity method that learns domain context by machine learning to produce models of context for terms retrieved from the Web in the filtering process to calculate the service similarity. We applied post filtering approach to hybrid term similarity based clustering approach that we proposed in our previous work. Experimental results show that our post-filtering approach works efficiently.

Keywords: Web service clustering, Web service filtering, Service similarity, Context aware Web service

1. INTRODUCTION
Web services, which share business logic, data and processes through a programmatic interface, represent an important way for businesses to communicate with each other and with clients. They are loosely coupled software components that are a popular implementation of service- oriented architecture. Web services can be published, located, and invoked across the Web through messages encoded according to XML-based standards, including Web Services Description Language (WSDL), Universal Description, Discovery, and Integration (UDDI), and simple object access protocol (SOAP) (Curbera, Duffler et al., 2002). Existing technologies for Web services have been extended to give value-added customized services to users through service composition (Paik, Chen et al., 2014). Developers and users can then solve complex problems by combining available basic services such as travel planners. Web service discovery, which aims to match the user request against multiple service advertisements and provides a set of substitutable and compatible services by maintaining the relationship among services, is a crucial part of service composition. Now most business organizations are moving towards the adoption of Web services, resulting in an increased number of services being published on the Internet in recent years. Thus, discovery of Web services is a challenging and time-consuming task because of unnecessary similarity calculations in the matchmaking process within repositories such as UDDIs and Web portals.

Clustering Web services into similar groups, which can greatly reduce the search space for service discovery, is an efficient approach to improving discovery performance. Clustering the Web services enables the user to identify appropriate and interesting services according to his or her requirements while excluding potential candidate services outside the relevant cluster and thereby limiting the search space to that cluster alone. Further, it enables efficient browsing for similar services within the same cluster. Web services can be clustered into functionally similar clusters by considering functional attributes such as input, output, precondition and effect (Dasgupta, Bhat et al., 2011).

A clustering approach requires similarity calculation method to compute the similarity of services. First, the method computes the similarity of service features. Then, the similarity of services is computed as an aggregate of the individual feature similarity values. Several methods have been used to compute the feature similarity in current functionally based clustering approaches, such as those using string-based methods like cosine similarity (Platzer, Rosenberg et al., 2009), the corpus-based methods like normalized Google distance (NGD) (Elgazzar, Hassan et al., 2010; Liu & Wong, 2009), knowledge-based methods like ontology methods (Xie, Chen et al., 2011; Wagner, Ishikawa et al., 2011) and hybrid term similarity (HTS) (kumara, paik et al., 2013) methods. These methods have their own drawbacks. String-based method like one-to-one matching may not accurately identify the semantic similarity among terms because of the heterogeneity and independence of service sources. These methods consider terms only at the syntactic level, whereas different service providers may use the same term to represent different concepts or may use different terms for the same concept. Further, method like cosine similarity does not perform the fine-grained measure for calculating service semantic similarity due to loss of the machine-interpretable semantics. On the other hand corpus-

http://hipore.com/ijsc
based methods such as NGD use up-to-date knowledge and information from the Internet, but they do not encode fine-grained information and that leads to reduce the precision of clustering results. Moreover, knowledge-based methods lack up-to-date information and the methods have the problem of shortage of high-quality ontologies. These drawbacks of the current similarity calculation approaches lead to reduce the performance of service clustering by reducing the recall and precision of the service clusters. Thus, it is very important to identify the invalid members in clusters and replace them into correct clusters to increase the performance of the clusters.

To identify the invalid member of clusters, we need to identify the real semantic that exist between services under a particular domain. Above similarity calculation methods compute the service feature similarity as global values without considering specific domains. Therefore, one common main issue in current functionally based clustering approaches is that they fail to identify semantic relationship that exists between services. For example, similarity value between services JaguarPrice and carPrice under the Vehicle domain is greater than the similarity value which is obtained by without considering the domain. Because, here term Jaguar in JaguarPrice service is considered not only as an animal but also as a car model. Although traditional search engine based methods reasoning as above, they do not consider about the domain specific context in measuring similarity. So, they fail to achieve more reasonable similarity value. It is very natural to match or retrieve information not only within a general context, but also within the context of a specific domain. Awareness of the domain-specific context in services will help to find tensors among the clusters, which characterize their elements. Context is any information that can be used to characterize the situation of an entity (Dey, 2001). In the literature, context awareness has been used to solve major problems in service computing such as recommendation (Abbar, Bouzeghoub et al., 2009) discovery (Suraci, Mignanti et al. 2007) and composition (Fujii & Suda, 2009). The various approaches have used different categories for the context. These categories include user properties such as location and time (Xiao, Zou et al., 2010; Bottaro, Bourcier et al., 2007), user preferences (Bottaro, Bourcier et al., 2007), and the environment affecting Web service delivery and execution such as service location (Li, Luo et al., 2013), service profiles and network bandwidth (Zhang, Liu et al., 2013; Chen, Yang et al., 2006). Contexts such as these are used as extra information in the matchmaking process to obtain the desired and most relevant services according to user requests. However, the contexts are not used to increase the semantics of functional similarity of the services or to cluster services, with these approaches considering nonfunctional properties as the context. In contrast, we propose to employ context to capture the hidden semantics of services for particular domains in identifying the invalid members in clusters.

In this paper, we proposed post-filtering approach based on functional aspect which uses Context Aware Similarity (CAS) method to calculate the service similarity in filtering process. Context is created using snippets that are extracted from real Web using search engines. Support vector machines (SVMs) are trained to produce models for computing the similarity of services for different domains. CAS method can improve semantic similarity of terms with up-to-date knowledge and encoded fine-grained information.

The rest of this paper is organized as follows. In section 2, we discuss the related work. Section 3 discusses the motivation for CAS based post-filtering method. Section 4 describes CAS method. Section 5 describes the CAS based post-filtering approach. Section 6 presents the experiments and evaluation. Finally, section 7 concludes the study.

2. RELATED WORK

In this section, we discuss work related to Web service clustering, context aware Web services and role of SVM in Web service categorizations.

2.1 Web Service Clustering

Clustering approach (Platzer, Rosenberg et. al., 2009) used string-based methods such as cosine similarity to compute the service similarity. Cosine similarity usually focuses on plain text, whereas Web services can contain much more complex structures, often with very little textual description. (Elgazzar, Hassan et al., 2010; Liu & Wong, 2009) combined string-based similarity methods such as structure matching with a corpus-based method based on NGD to measure the similarity of Web service features and to cluster them appropriately. However, structure matching may not accurately identify the semantic similarity among terms because of the heterogeneity and independence of service sources. These methods consider terms only at the syntactic level. Further, there can be a loss of the machine-interpretable semantics found in service descriptions when converting data provided in service descriptions into vectors in string based Information Retrieval (IR) techniques. Moreover, as we discussed in introduction section, NGD does not take into account the context in which the terms co-occur, and, although the method uses up-to-date knowledge and information from the Internet, it does not encode fine-grained information, leading to low precision in the clustering results.

(Nayak & Lee, 2007) proposed a Web service discovery approach with additional semantics and clustering. They took advantage of the OWL-S ontology and WordNet lexicon to enhance the description with semantics. Each of the extracted terms from the service documents was expanded to enhance its semantics by using WordNet to identify synonyms. They used the Jaccard coefficient in computing the service similarity.

(Wen, Sheng et al., 2011) presented a Web service discovery method based on semantics and clustering.

http://hipore.com/ijsc
Similarities among services were computed by using knowledge-based methods based on WordNet and ontologies. (Wagner, Ishikawa et al., 2011) arranged Web services in a functionality graph based on the Exact and Plug-in relationships. Logic-based filters were used to calculate the service similarity. Connected components in the graph were considered as clusters. (Xie, Chen et al., 2011) measured the similarity based on two aspects (functional and process), with the aim of clustering services. A weighted-domain ontology method was used to calculate the functional similarity using input and output parameters and the domain ontology was developed using a semantic dictionary and existing ontologies from the Internet. However, by using fixed ontologies and fixed knowledge bases, these approaches failed to capture reasonable similarity values for different domains. Further, the ontologies considered shared model of domain. Thus, the approaches fail to identify the real semantic relation exist between services under a particular domain. Furthermore, knowledge-based methods lack up-to-date information and have the problem of a shortage of high-quality ontologies. As we discuss, each method has its own drawbacks. Thus, we need one common solution to fix the errors that occurred due to those drawbacks. We focus on filtering the clusters to identify the invalid members.

2.2 Context-Aware Web Services

Instead of a formal definition of context (Dey, 2001), we can find enumeration definitions for context in the literature. In an enumeration definition, researchers define the context from three aspects, namely where a service is, what other services are present, and what resources are nearby. They argue that the context should include items such as location, lighting, noise level, network connectivity, and communication costs (Rong & Liu, 2010). (Zhang, Liu et al., 2013) proposed a novel approach to modeling the dynamic behavior of interacting context-aware Web services, aiming to process and take advantage of context effectively in realizing the behavior adaptation of Web services. Moreover, for service recommendation, some researchers have used both personal-usage history-oriented contexts and group-usage history-oriented contexts, such as collaborative filters that assess users’ ratings of Web services (Resnick, Iakovou et al., 1994; Manikrao & Prabhakar, 2005).

However, although these context-aware services help to identify those services that match a user’s preference, user properties, usage history, or environment properties in the match-making process, they do not bootstrap to increase the performance of clustering. Further as we mentioned, these approaches consider only nonfunctional aspects. In our approach, we define a context that can capture the hidden semantics of terms within particular domains.

2.3 SVM in Service Categorization and Term Similarity Calculation

(Bollegala, Matsuo et al. 2007) used SVM to measure the similarity of two terms. They used fixed patterns extracted from WordNet to identify the semantic relationships between terms. Those patterns and search-engine-based similarity values were used to generate feature vectors. (Zhang, Wang et al., 2012) proposed an approach to service categorization involving novel feature-vector elements for service characteristics and an extension of the SVM-based text classification technique to enhance the accuracy of the service categorization. This approach incrementally established domain knowledge and leveraged that knowledge to automatically verify and enhance the service categorization. Web search engine based term similarity measure and results integration by SVM were used to discover matched service on the trip domain by (Paik & Fujikawa 2012). In our post filtering approach, we use SVM to compute semantic similarity to identify the invalid cluster members.

3. Preliminaries

In this section, we present motivating scenarios for post filtering, context-awareness and the architecture of our proposed approach. In addition, we describe the SVM used in machine learning to generate models for post filtering cluster.

3.1 Web Service Clustering and Motivation for Post Filtering

Service clustering, which can greatly reduce the search space of service discovery, is an efficient approach to increasing the discovery performance. The idea is to organize semantically similar services into one group. Various similarity-computing matrices, such as string- and knowledge-based methods, have been used in current clustering approaches. However, as we discussed in previous sections these methods have their own drawbacks which reduce the performance of service clustering. Further, these methods do not use domain context in measuring the similarity, which hinders the accurate calculation of service similarity. In a real situation, there will be semantic relationships between services within a particular domain that should be utilized. In this research, we focus on post filtering approach which addresses the drawbacks of current clustering approaches. We calculate term similarity according to context by learning in a domain in the filtering process to identify the hidden semantic of services under particular domain.

3.1.1 Motivating Example for post filtering approach

To better motivate and illustrate the importance of post filtering, we will refer to a simple example. This example contains 13 services from three different domains (Medical, Electronic and Food). Assume, we want to cluster the services according to the domains. If we use string-based methods such as one-to-one matching or cosine similarity,
then we will obtain the clustering results as shown in Figure 1. We can see that AppleComputer is placed with Food domain services incorrectly, because the service uses the same terms as some services in the Food domain and the methods do not consider Apple as a computer manufacturer. In addition, terms used in S3 and S13 not appear in the service names of other services in the Medical and Electronic domains respectively. Thus, we cannot guarantee that AmbulanceService and BlackBerryZ10Price services will be placed in Medical and Electronic clusters. On the other hand, corpus-based methods like NGD may also obtain low precision in filtering as we discussed in the Introduction section due to lack of encoded fine-grained information. Further, we cannot guarantee that there exist concepts for services like AppleComputer and BlackBerryZ10Price in fixed ontologies to calculate similarity using knowledge-based methods. Sometime, current approaches may identify the BlackBerryZ10Price as a member of Food cluster due to lack of domain knowledge. Thus, it is very important to consider the domain knowledge to identify the invalid members of the clusters.

Figure 1. Example Scenario

3.1.2 Motivating Example for Context Awareness

Semantic similarity between two terms depends on the domain. For example, the similarity value between the terms “jaguar” and “car” is higher in the Vehicle domain than in the Animal domain, or in any other domain. Figure 2 illustrates the distance between two terms in different domains. To identify this semantic, we can use domain specific context. Some services share characteristics with the more than one domain because of the terms used in service feature and does not identify easily with one cluster among domains. They may be placed between cluster boundaries. Awareness of the domain-specific context in services will help to find tensors among the clusters, which characterize their elements. Thus, with the domain context we can identify the more appropriate cluster to the service.

http://hipore.com/ijsc

3.2 Proposed Post Filtering Approach

Figure 3 shows the architecture of the proposed CAS based post-filtering approach. First, we cluster the services by calculating service similarity using an existing approach. In this research, we use HTS approach which we proposed in our previous work (kumara, paik et al., 2013) and edge-count based (ECB) approach. Structure of WSDL document is used to cluster services and we translate material retrieved in other formats into WSDL. Then, we identify the cluster centers using cluster center identification algorithm, which we proposed in our previous work (kumara, paik et al., 2013). Next, the post filtering is applied using CAS module. Here, first we detect the domain of the cluster using cluster center and CAS module. Then, we calculate the similarity of cluster center with other services in the cluster in order to identify the invalid members. We consider term similarity according to context by learning in a domain using the CAS module, which uses search query results and trained SVM. After identifying the invalid members of the cluster, we calculate the similarity of invalid members with other cluster centers to identify the correct clusters. Finally, we swap the invalid cluster members appropriately.

3.3 SVM in Term Similarity Calculation
CAS module in the above architecture calculates the services similarity in the filtering process using the trained SVM. Service descriptions are free-text documents that include a variety of terms. The existing approaches to calculate similarity of terms in services are based on general concept. Here, we want to devise a new approach to calculate term similarity according to domain context using model from SVM. The SVM is a state-of-the-art classification method that uses a learning algorithm based on structural risk minimization (Vapnik & Cortes, 1995). It outperforms other categorization methods. The classifier can be used in many disciplines because of its high accuracy, ability to deal with high dimensions, and flexibility in modeling diverse sources of data (Cristianini & Shawe-Taylor, 2000). Our approach adopts the SVM as a machine learner in the post filtering process to enhance service clustering.

In classification, the SVM first transforms the input space to a higher-dimensional feature space through a nonlinear mapping function, and then constructs a separating hyperplane that has the maximum distance from the closest points of the training set. The optimal hyperplane is constructed by maximizing the distance between samples in different classes and the hyperplane. For a binary classification problem, given a set of linearly separable samples:

\[ P = \{(x_i, y_i)\}_{i=1}^N \quad \text{and} \quad y_i = [+1, -1] \]

where \( x_i \) is an \( n \)-dimensional vector. Here, we use the domain context to construct the vector (we discuss the formation of the vector in detail in Section 4). The label of the class that the vector belongs to is \( y_i \). The objective of the SVM is to find an optimal hyperplane with following condition:

- for \( y_i = +1 \), \( W^T x_i + b \geq 1 \)
- for \( y_i = -1 \), \( W^T x_i + b \leq -1 \)

Here, \( W^T \) is a weight vector and \( b \) is a bias. By introducing Lagrange multipliers, we can arrive at the following optimization problem:

minimize: \[ W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j) \]

subject to: \[ \sum_{i=1}^N \alpha_i y_i = 0, \forall i : 0 \leq \alpha_i \] (1)

Here, \( K(x_i, x_j) = x_i \cdot x_j \) is a kernel function. The hyperplane decision function can then be expressed as:

\[ q(x) = \text{sign} \left( \sum_{i=1}^N \alpha_i y_i K(x_i, x) + b \right) \] (2)

The training process determines \( \alpha_i \) and \( b \), and the criteria for optimization are to maximize the margin as well as and minimize the error.

\( q(x) \) indicates the distance between testing data and the optimal hyperplane. However, the output cannot be used directly to measure the semantic similarity of services. Constructing an SVM to produce the posterior probability \( P(\text{class}|\text{input}) \) that the feature vector belongs to a positive class is a fine-grained solution to this problem. In the literature, several approaches have been proposed to convert the uncalibrated output \( q(x) \) of the SVM into a probability (Wahba, 1992; Platt, 2000).

4. THE CAS METHOD

We used CAS method to calculate the service similarity in post filtering. In this section, we describe the steps of CAS method, namely extracting the domain context, generating feature vectors, training the SVM and calculating term similarity by converting the SVM output into posterior probabilities.

4.1 Outline

We first analyzed snippets obtained from search engines for term pairs to identify significant information that help to capture the semantics of terms within a domain. For example, consider the following snippets (see Fig. 4) from the Google search engine for the queries “apple computer,” “apple banana,” and “software hardware.” There are domain-related terms that are frequently used in that domain (e.g., terms such as hardware and software are frequently used in the Computer domain, and terms such as fruit and vegetable are frequently used in the Food domain). According to snippets sn1 and sn2, we can see that the terms Apple and Computer are associated with frequently used terms in the Computer domain, such as laptop and desktop. According to snippets sn3 and sn4, the terms Apple and Banana are associated with frequently used terms in the Food domain, such as fruit and drink. The terms software and hardware are also associated with frequently used terms in the Computer domain, according to snippets sn5 and sn6. Using these snippets, we can determine that the terms Apple and Computer are semantically related to each other in the Computer domain, and Apple and Banana are semantically related to each other in the Food domain. If two terms are associated with more frequently used terms, then there will be a strong semantic relationship between those two terms. The term combination “Hardware–Software” is associated with more frequently used terms in the Computer domain than other two combinations. Therefore, there exists a strong semantic relationship between the terms combinations for the Computer domain. Variation of
semantic relationships between term pairs can be shown as the following ordering under the Computer domain:

- hardware–software > apple–computer > apple–banana

If we consider the Food domain, the sequence will be:

- apple–banana > apple–computer > hardware–software

We extended the definition of “context” beyond that used in ubiquitous computing. Here, we define context as $C = \{T_1, T_2, \ldots, T_n\}$, where $T_i$ is a term that is frequently used in a domain $d_i$. Context is based on the domain and it varies from domain to domain. We extract the context using Web search engines. To measure the semantic similarity of two terms within a particular domain, we implement a domain filter by training an SVM for that domain. For example, to measure the similarity between car and vehicle under the Vehicle domain, we implement a Vehicle domain filter.

Figure 4. Snippets for the Queries from the Google Search Engine.

4.2 Generating Context Vectors for Domains

We generated context vectors for each required domain to implement the domain filters (see Algorithm 1). The context vector for domain $d_i$ contains all the terms in the domain-specific context $C$ of $d_i$. To extract the domain-specific context, we used the Google and Wikipedia search engines. We extracted frequently used terms in domain $d_i$ as the context. Figure 5 shows this process. First, we gave the domain name as the search query to the two search engines and retrieved the top 100 snippets from each search engine (SnippetG($d_i$) and SnippetW($d_i$)). 200 snippets are extracted for each domain (Lines 1–3 of Algorithm 1). We then performed stop-word filtering and computed the term frequency–inverse document frequency (TF–IDF) value of all the terms in the 200 snippets for each domain (Lines 4–13). The advantage of using TF–IDF is that it reflects the importance of the term $T_i$ for domain $d_i$ from among the collection of domains $d_1, d_2, d_n, \ldots, d_m$. Even if term $T_i$ is frequently used in domain $d_n$, it may not be a domain-specific term, but may also be a common term for other domains. If so, we obtain no advantage by selecting that term as a context term. We therefore use the TF–IDF values to identify the preferred important domain-specific terms.

After computing the TF–IDF values, we selected the 200 terms that had the highest TF–IDF values from domain $d_i$ as the context for that domain. We then generated the context vector for each domain $d_i$, where each element was a frequently used term in that domain (Lines 14).

**Algorithm 1 Context vector generation for domains**

```
Input $D$: Array of domains
Output $C$: Context vectors for each domain

1: For each domain $d_i$ in $D$ do
2:   $S_{d_i} = \text{getSnippets}(d_i)$; //get total 200 snippets from Google and Wikipedia search engines
3:   end-for

4: For each domain $d_i$ in $D$ do
5:   For each snippets $s_i$ in $S_{d_i}$ do
6:     $\text{StopWord\_filtering}(s_i)$;
7:   end-for
8:   $T_{d_i} = \text{Calculate\_term\_frequency}();$ // 2D array with term and term frequency
9:   end-for
10: For each domain $d_i$ in $D$ do
11:   For each term $T_i$ in $T_{d_i}$ do
12:     $y_{c_i} = \text{calculate\_TF\_IDF}(T_i)$;
13:   end-for
14:   $C_{d_i} = \text{generateContextVector}();$ //select 200 terms with highest TF-IDF value
15: end-for
```

4.3 Training the SVMs

After generating the context vectors for each domain, next step is to train a separate SVM for each domain to implement the domain filters. For example, SVM has to be trained within a medical context to implement the Medical domain filter. Each SVM was trained to classify same-domain term pairs that belonged to the domain under consideration and term pairs that did not belong to that domain. We extracted terms for five domains: Book, Medical, Vehicle, Food, and Film from dictionaries, thesauruses, and service documents. We prepared term pairs from those selected as belonging to the domain under consideration to be the positive-training dataset. Negative-training term pairs were prepared by taking terms from other domains. For example, in implementing the filter for the Vehicle domain, we prepared positive term pairs by taking terms from the Vehicle domain such as car–automobile or...
Consider training SVM to implement the domain filter for domain \( d_i \). We need to prepare a training dataset \( P = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \} \), where \( x_i \) is the feature vector and \( y_i \) is the expected class label for the \( i \)th instance. For each training pair of terms \( (a, b) \), we created a feature vector \( x_i \), as shown in Algorithm 2. Figure 6 shows the feature vector generation process. We first searched Google for \( (a, b) \) to find snippets (Line 1). After that frequency of each term \( T_i \) in the snippets, where \( T_i \) is a member of the context vector for domain \( d_i \) (Lines 3–9) was counted. Then count of each term \( T_i \) was normalized by dividing it by the total number of frequently used terms (Line 11). Finally, we had a vector of frequently used terms in which each element is the normalized frequency of the corresponding frequently used term. Feature vectors were produced for all positive and negative term pairs in this manner. The SVM was then trained with the labeled feature vectors to implement the filter for domain \( d_i \). In this way, we trained SVMs for each domain that required a filter.

Assume we need to compute the similarity of two terms within the Vehicle domain. SVM that was trained for the Vehicle domain was needed to use in this situation. First, we generated the feature vector for the selected term pair using the same method that we used to generate feature vectors for training term pairs. As mentioned above, in current research disciplines, SVM can be used to classify a given data set. We adopt it here to calculate semantic similarities. We define the semantic similarity between terms as the posterior probability that the feature vector belongs to the same-domain term-pair (positive) class as that for the domain under consideration. However, the output of SVM \( q(x) \) is both uncalibrated and not in the form of a probability. To convert the SVM output to a calibrated posterior probability, we follow a previous approach (Platt, 2000) that uses a sigmoid function.

5. POST-FILTERING METHOD

Post filtering is used as a post step of the clustering. Our objective is identifying the invalid members of the clusters. Thus, first we need to cluster the Web services using existing approach. In evaluation, first we used HTS approach (Kumara, Paik et al., 2013) and ECB approach as the clustering approaches. In section 6.1, we will give the brief description of HTS approach. In this section, we describe the steps of post filtering method, namely detect domain of the clusters, identifying invalid members of the clusters and identifying correct clusters for invalid members. Algorithm 3 describes the post filtering process.

5.1 Detecting Domain of the Clusters

We detected domain names of the clusters using cluster centers and CAS method (Line 1–3). First, we prepared term set to represent the each cluster center. For that, we extracted service features of the center service from the WSDL document. Here, we used service name, operation name, Input and Output as the features. If the service feature value is a complex term, then we tokenized it into individual terms based on several assumptions. For an example, we splitted the service name AuthorizePhysician into two parts (Authorize, Physician) based on the assumption that a capital character indicates the start of a new word. After tokenizing the name, stop-word filtering was performed to remove any stop words. Then, non-sense numbers were removed. Remaining words of the four features were considered as represent term set of cluster center service. In that way, we prepared cluster center represent term set for each centers.

Then, we prepared term set for represent the each domain that we clustered the services. We considered the top ten terms of the context vector of the domain \( d_i \) as represent term set for domain \( d_i \). Next, we computed similarity between represent term set of the cluster center with represent term set of each domain using CAS method. When we were computing the similarity, we used SVM that

---

Algorithm 2: Feature vector generation for term pairs

Input \( \tau \): Array of 200 terms // frequently used terms in domain \( d_i \)
(Output \( x_i \): Feature vector of term pair \( (a, b) \))

1. \( S = \text{getSnippets}(a, b) \);
2. \( \text{total_count} = 0 \);
3. \( \text{for each term } T_i \text{ in } \tau \text{ do} \)
4. \( \text{count}_{T_i} = 0 \);
5. \( \text{for each snippet } s \in S \text{ do} \)
6. \( \text{count}_{T_i} = \text{count}_{T_i} + \text{count}(T_i, s) \);
7. \( \text{end-for} \)
8. \( \text{total_count} = \text{total_count} + \text{count}_{T_i} \);
9. \( \text{end-for} \)
10. \( \text{for each term } T_i \text{ in } \tau \text{ do} \)
11. \( N_{T_i} = \text{normalized(count}_{T_i}, \text{total_count}) \);
12. \( \text{end-for} \)
13. \( \text{generateFeatureVector}(N_{T_i}) \);

---

Figure 6. Process of Feature Vector Generation

4.4 Calculating Term Similarity from the Model

http://hipore.com/ijsc
was trained for relevant domain. For an example, if we compute the similarity between center of the cluster $c_i$ and $Book$ domain represent term set, then we need to use SVM that was trained for $Book$ domain.

Final similarity value between two represent term sets (cluster center and domain) was calculated using (3):

$$Sim_i(c_i, d) = W_{DF} \sum_{x=1}^{m} \sum_{y=1}^{n} CAS_{sim} \left( s_x, r_y \right) \frac{m+n}{T} \quad (3)$$

where $s_i$ and $r_j$ denote the individual terms in represent term set of cluster center of the cluster $c_i$ and domain $d$, respectively. The values $m$ and $n$ are the numbers of individual terms in two represent term sets. It is important to determine the strength of the cluster center to be in the considered domain. We therefore introduced a weight $W_{DF}$ as a domain factor and defined by (4):

$$W_{DF} = \frac{1}{1 + \log_{10} \left( \frac{T_p}{D_p} \right)} \quad (4)$$

where $T_p$ is the number of term pairs between two represent term sets and $D_p$ is number of term pairs that belong to the considered domain.

After computing the similarity value, we selected the domain for the cluster based on the highest similarity value. If the center of the cluster $c_i$ is obtained highest similarity value with represent term set of domain $d$, then services in the cluster $c_i$ are belonging to the domain $d$. In the same manner, we identified the domain for remaining cluster centers. When we were measuring the similarities for remaining centers, we did not used domain represent term set that were already detected as a domain for a particular cluster.

5.2 Identifying Invalid Members of the Clusters

To identify the invalid members, we computed similarity value between center service and all other services in the cluster under the detected domain (Line 4-9). To calculate the similarity, we extracted service name and operation name as the features. First, we calculated the individual feature similarity ($Sim_F(c_{i,c}, S_j, F)$) using (3) and (4). In this case, $s_i$ and $r_j$ denote the individual terms in feature $F$ of cluster center $c_i$ and service $S_j$, respectively. The values $m$ and $n$ are the numbers of individual terms in two features.

Then, we calculated the final service similarity value by integrating the similarity values of service name and operation name using (5):

$$Sim_i(c_{i,c}, S_j) = \alpha \cdot Sim_N(c_{i,c}, N_j) + \beta \cdot Sim_O(c_{i,c}, O_j) \quad (5)$$

where $Sim_N(c_{i,c}, N_j)$ and $Sim_O(c_{i,c}, O_j)$ are the similarity values of service name and operation name, respectively. Parameters $\alpha$ and $\beta$ are weights for each feature. Further, $\alpha + \beta = 1$.

Next, we defined following rule 1 to filter the services.

Rule 1: If the similarity value $Sim_i(c_{i,c}, S_j) < T$, then service $S_j$ is considered as an invalid member of cluster $c_i$. Otherwise service $S_j$ is a valid member of cluster $c_i$.

According to the rule 1, if the similarity value between center service of cluster $c_i$ and service $S_j$ in cluster $c_i$ is lower than given threshold value $T$, then we temporary consider service $S_j$ as an invalid member of cluster $c_i$ and keep it for future processing to identify the correct cluster. However, if the similarity value is greater than $T$, then we identify the service $S_j$ as a valid member of cluster $c_i$.

5.3 Identifying Correct Clusters for Invalid Members

After identifying the invalid members of the cluster $c_i$ we need to replace them into the correct clusters (Line 10-21). First, we computed the similarity between invalid member $S_j$ and centers of all other clusters $c_x$ where $x \neq j$. To compute the similarity value, we used CAS method as discuss in above sub-section (using (3), (4) and (5)). Here also we need to use SVM that was trained for relevant domain. If we compute similarity between $S_j$ and center of
cluster \( c_o \), then we need to use SVM that is trained for the domain which services on the cluster \( c_i \) are belonging to.

After calculating the similarity value of \( S_j \) with other all cluster centers, we determined the correct cluster for that service by applying following rule 2.

**Rule 2:** If the similarity value between services \( S_j \) and center service of cluster \( c_i \) (\( \text{Sim}_d(c_i, S_j) \)) is the highest similarity value comparing to other similarity values which are obtained by calculating the similarity of \( S_j \) and other cluster centers, then service \( S_j \) moves into cluster \( c_i \).

In the same manner, we identified the correct clusters for all the invalid members in all the clusters and placed them into correct clusters.

However, if all similarity values are lower than the similarity value which is obtained previously by calculating the similarity with center of the cluster \( c_i \) (center of the cluster where already service \( S_j \) belongs to), then we keep the service in the current cluster \( c_i \) without applying rule 2 and we consider service \( S_j \) as a valid member for cluster \( c_i \). Figure 7 shows the procedure of applying rule 1 and rule 2.

![Figure 7. Procedure of Applying Rule 1 and Rule 2](image)

Next, to investigate the effect of domain context in measuring the term similarity, we calculated the similarity of term pairs by five domain filters and analyzed the effect of domain context in terms of changes in similarity values. We selected the term pairs from the WordSimSimilarity-353 test collection published by (Finkelstein, Gabrilovich et al., 2002) as the test dataset.

Then, we evaluated the CAS method in comparison to existing methods, computing term similarities via the NGD and edge-count-based methods, as used in WordNet. NGD is a corpus-based method and edge count is a knowledge-based method. Here, we used Pearson correlation (6) to check the performance of each of the term similarity methods:

\[
r(\text{HR}, \text{IRT}) = \frac{\sum (ts_1, ts_2) - \sum ts_1 \sum ts_2}{\sqrt{\prod_{i=1}^{2} \left( \sum ts_i^2 - \frac{\left( \sum ts_i \right)^2}{N} \right)}}
\]

Here, \( \text{HR} \) is the human rating and \( \text{IRT} \) is the IR-based term-similarity method. Parameters \( ts_1 \) and \( ts_2 \) are the human-rating similarity and the term similarity for terms, respectively. Parameter \( N \) is the number of term pairs.

Finally, we evaluated the effect of post filtering algorithm. Here, first we clustered services using HTS method. To cluster the Web services, WSDL documents related to the Book, Medical, Food, Film, and Vehicle domains were gathered from real-world Web service repositories, and the OWL-S (http://projects.semwebcentral.org/projects/owl-s-tcl. n.d) test collection to act as the services dataset. We computed precision, recall, F-measure and purity of the clusters to evaluate the proposed approach. As another clustering approach, we used ECB approach for further evaluation and applied the post filtering.

### 6. Experiments and Evaluations

The experiments were conducted on a computer running Microsoft Windows 7, with an Intel core i7-3770, a 3.40 GHz CPU and 4 GB RAM. Java was used for CAS, the service affinity calculations, and the SVM implementation. We implemented five domain filters, namely Book, Medical, Food, Film, and Vehicle, by training an SVM for each domain.

First, we evaluated the performance of different SVM kernels. We consider linear kernels, polynomial kernels, and a radial bias function and measured the accuracy of each kernel type to identify the best kernel for our implementation.

http://hipore.com/ijsc
An agglomerative algorithm was used as the clustering algorithm. To identify the cluster centers we used cluster center identification algorithm which we proposed in our previous work (Kumara, Paik et al., 2013). The algorithm was implemented by combining the service-similarity values with the TF–IDF value of the service name, which reflects the importance of the service to its cluster. Experimental results of previous work showed that cluster identification algorithm works efficiently.

6.2 SVM kernel performance

As described above, we experimented with different kernel types to select the best kernel for our implementations. Accuracy values were 94.80% for linear, 93.53% for radial bias function, 71.30% for polynomial degree = 3, and 43.00% for polynomial degree = 2. According to the results, the best performance was obtained using a linear kernel and the lowest performance was obtained by using higher-degree kernels (polynomial degree = 2 or 3). We therefore used a linear kernel in our implementation.

6.3 Term Similarity Methods Evaluation

We selected term pairs from WordSimilarity-353 test collection as the test dataset. First, we calculated the term similarities using the CAS method. Here, we calculated similarities for the five domains using the five trained SVMs and we considered the highest value as the best value. The result in the Table I shows that the similarity values of a term pair under different domains. Due to the space limitation here we included limited number of term pairs in the table. The results in Table I show that the similarity value of a term pair differs from domain to domain. For example, the similarity value between emergency and victim was 0.99 for the Medical domain, 0.20 for the Vehicle domain, and 0.00 for all other domains. When we analyzed these similarity values, we can see that the domain context affects the similarity values greatly. We can therefore determine that domain context can play a significant role in the values obtained for term similarity in a particular domain.

Next, we compared the CAS method with existing similarity calculation methods. Table II shows the correlation of each method with a human rating. Here, we computed term similarities with the Web-PMI and edge count-based methods, which was used WordPMI. WebPMI is a corpus-based method and edge count is a knowledge-based method. Table II shows similarity value results and we calculated the Pearson correlation of each method with the corresponding human ratings by (6). The results show that the CAS method gave higher correlation values than the other two methods. When we analyzed the results, we saw that the CAS method computed the similarity by considering the semantic relations of terms in the domain.

For an example, the method gave the highest similarity value for the term pair jaguar and car compared with the other two methods. We therefore determined that domain-specific contexts help to improve the semantic similarity value of term pairs.

### Table I. Term Similarity with Context Aware Term Similarity in Different Domains.

<table>
<thead>
<tr>
<th>Word 1</th>
<th>Word 2</th>
<th>Medical</th>
<th>Book</th>
<th>Film</th>
<th>Vehicle</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>paper</td>
<td>0.00</td>
<td>0.89</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>automobile</td>
<td>car</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>0.94</td>
<td>0.00</td>
</tr>
<tr>
<td>psychology</td>
<td>doctor</td>
<td>0.90</td>
<td>0.03</td>
<td>0.05</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>jaguar</td>
<td>car</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.99</td>
<td>0.00</td>
</tr>
<tr>
<td>emergency</td>
<td>victim</td>
<td>0.99</td>
<td>0.00</td>
<td>0.00</td>
<td>0.20</td>
<td>0.00</td>
</tr>
<tr>
<td>liquid</td>
<td>water</td>
<td>0.10</td>
<td>0.10</td>
<td>0.19</td>
<td>0.29</td>
<td>0.70</td>
</tr>
<tr>
<td>video</td>
<td>archive</td>
<td>0.00</td>
<td>0.04</td>
<td>0.95</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>doctor</td>
<td>nurse</td>
<td>0.80</td>
<td>0.00</td>
<td>0.24</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Table II. Term Similarity with CAS and Existing Approaches.

<table>
<thead>
<tr>
<th>Word 1</th>
<th>Word 2</th>
<th>Human Rating</th>
<th>Edge Count</th>
<th>WebPMI</th>
<th>CAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>paper</td>
<td>7.46</td>
<td>0.70</td>
<td>0.97</td>
<td>0.89 (Book)</td>
</tr>
<tr>
<td>book</td>
<td>library</td>
<td>7.46</td>
<td>0.79</td>
<td>1.00</td>
<td>0.90 (Book)</td>
</tr>
<tr>
<td>food</td>
<td>fruit</td>
<td>7.52</td>
<td>0.77</td>
<td>0.90</td>
<td>0.77 (Food)</td>
</tr>
<tr>
<td>cup</td>
<td>food</td>
<td>5.00</td>
<td>0.63</td>
<td>0.84</td>
<td>0.6 (Food)</td>
</tr>
<tr>
<td>psychology</td>
<td>clinic</td>
<td>6.58</td>
<td>0.62</td>
<td>0.81</td>
<td>0.65 (Medical)</td>
</tr>
<tr>
<td>doctor</td>
<td>liability</td>
<td>5.19</td>
<td>0.40</td>
<td>0.60</td>
<td>0.53 (Medical)</td>
</tr>
<tr>
<td>jaguar</td>
<td>car</td>
<td>7.27</td>
<td>0.47</td>
<td>0.80</td>
<td>0.99 (Vehicle)</td>
</tr>
<tr>
<td>automobile</td>
<td>car</td>
<td>8.94</td>
<td>0.83</td>
<td>0.82</td>
<td>0.94 (Vehicle)</td>
</tr>
<tr>
<td>movie</td>
<td>star</td>
<td>7.38</td>
<td>0.70</td>
<td>0.95</td>
<td>0.89 (Film)</td>
</tr>
<tr>
<td>movie</td>
<td>theater</td>
<td>7.92</td>
<td>0.58</td>
<td>0.93</td>
<td>0.99 (Film)</td>
</tr>
</tbody>
</table>

Correlation: 1.00 0.61 0.53 0.86

6.4 Post-Filtering Evaluation

First, we used precision, recall and F-measure to evaluate the clusters before applying the post-filtering and
after applying the post-filtering. We set threshold $T$ in rule 1 as 0.7. Precision is the fraction of a cluster that comprises services of a specified class. Recall is the fraction of a cluster that comprises all services of a specified class. The F-measure measures the extent to which a cluster contains only services of a particular class and all services of that class.

Following (7), (8) and (9) were used to calculate these three criteria.

$$\text{Precision}(i, j) = \frac{\text{NM}_{ij}}{\text{NM}_j}$$  \hspace{1cm} (7)

$$\text{Recall}(i, j) = \frac{\text{NM}_{ij}}{\text{NM}_i}$$  \hspace{1cm} (8)

Here, $\text{NM}_{ij}$ is the number of members of class $i$ in cluster $j$, $\text{NM}_j$ is the number of members of cluster $j$ and $\text{NM}_i$ is the number of members of class $i$.

The F-measure of cluster $i$ with respect to class $j$ is,

$$F(i, j) = \frac{2 \times \text{Precision}(i, j) \times \text{Recall}(i, j)}{\text{Precision}(i, j) + \text{Recall}(i, j)}$$  \hspace{1cm} (9)

First, we used HTS approach to cluster the services. The experimental results in the Table III shows result of three criterias in HTS method and after applying post-filtering. Here we used WSDL from the above selected data set.

According to the experimental results, there are no false positives for the Medical cluster in either approach, the precision values for both approaches being 100%. Further recall value of the Book cluster is 100% in both approaches. However, we observed that post-filtering approach increased the precision, recall and F-measure values of other every other cluster. For an example, precision of the Vehicle cluster increased by 18.4% and recall of the Film cluster increased by 10.7% as the highest precision and recall increments. When we analyzed the WSDL documents, we observed that some extracted features failed to identify their ontology. For example, the CheckRoomAvailability and CheckEquipmentAvailability services that were belonging to the Medical domain were not successfully placed in the Medical cluster in HTS method. In this case, the service failed to join with other services, such as MedicalOrganization, HospitalClinic and many others in the Medical domain, in generating the ontology. This was result to reduce the precision and recall values in HTS method. But, post-filtering approach measure the similarity based on the domain specific context and able to capture the hidden semantic between two services exists under a certain domain.

However, post-filtering approach also obtained the lowest recall value in Medical cluster as the HTS approach. We saw that some services like PatientTransport and selectMedicalFlight were placed to the Vehicle cluster in post-filtering approach. Because, the services obtained slightly higher similarity value with Vehicle domain represent term set than Medical domain represent term set. However, when we analyzed these services, we saw that this kind of service shows the multi-domain nature. They used terms related with more than one domain.

Then, we calculated purity of class as another evaluation criterion. The purity of a cluster solution is defined as,

$$\text{Purity} = \frac{1}{n} \sum_{j=1}^{k} \max \{ n^j_i \}$$  \hspace{1cm} (10)

where $n$ is the total number of services and $n^j_i$ is the number of services in cluster $i$ belonging to domain class $j$.

Figure 8 shows the purity values for the two approaches with respect to the number of services. Purity decreased when increasing the number of services in both approaches. However, post-filtering approach obtained higher purity values throughout. In addition, the rate of purity-value decrease in post-filtering approach was smaller than for the HTS approach where without post-filtering.

Table III. Performance Measures of Clusters with HTS approach.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>HTS approach without post-filtering</th>
<th>After applying post-filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision %</td>
<td>Recall %</td>
</tr>
<tr>
<td>Book</td>
<td>86.7</td>
<td>100</td>
</tr>
<tr>
<td>Medical</td>
<td>100</td>
<td>82.0</td>
</tr>
<tr>
<td>Food</td>
<td>96.0</td>
<td>91.3</td>
</tr>
<tr>
<td>Film</td>
<td>91.0</td>
<td>88.0</td>
</tr>
<tr>
<td>Vehicle</td>
<td>80.3</td>
<td>94.8</td>
</tr>
</tbody>
</table>

Figure 8. Cluster Performance with Post-Filtering
As the next step of evaluation, we applied the post filtering approach ECB clustering approach. Here, first we cluster services using ECB approach. ECB approach used WordNet to calculate the similarity between services. We used 200 WSDL files as the dataset and computed the precision of each clusters. Table IV shows the increment of precision values after applying the post filtering approach to the ECB approach. According to the experimental results, there are no false positives for the Medical cluster and Vehicle cluster in both cases, the precision values for both approaches being 100%. But, post-filtering approach increased the precision values of Food, Film and Book clusters. For an example, precision value of Book cluster is increased by 16.2% by the post filtering.

Table IV. Performance Measures of Clusters (ECB Approach).

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Precision %</th>
<th>ECB approach without post-filtering</th>
<th>ECB approach with post-filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book</td>
<td>70.0</td>
<td>86.2</td>
<td></td>
</tr>
<tr>
<td>Medical</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Food</td>
<td>88.0</td>
<td>95.7</td>
<td></td>
</tr>
<tr>
<td>Film</td>
<td>75.0</td>
<td>89.2</td>
<td></td>
</tr>
<tr>
<td>Vehicle</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

According to the above evaluations results, we observed that post filtering increase the performance of clusters by replacing the invalid members of clusters into correct clusters. The post filtering approach able swap the services into correct clusters by considering the semantic of services that exits under a particular domain. Thus, we can use post filtering as a post step of the clustering to increase the clustering performance.

7. CONCLUSIONS

In this paper, we have proposed a CAS based post-filtering method to increase the Web service clustering performance. The CAS method uses models learned from real Web and domain datasets for context retrieved from Web search engines that captures the hidden semantics of services within a particular domain. According to the term similarity calculation results, we show that context plays a significant role in measuring the semantic similarity of terms. In this work, we implemented five domain filters by training five learners to selected domains. In the evaluation, first we clustered services using HTS and ECB approaches. Then, post filtering was applied for both approaches. Our post-filtering approach was able to capture the hidden semantics between Web services which exists under a particular domain via CAS method. Invalid members in clusters were identified through this hidden semantic and placed them in correct clusters. Experimental results showed that the post-filtering approach performed better than without post-filtering method in both clustering approaches. In HTS approach, average precision value and average recall value are increased by 6.48% and 5.88% respectively by the post filtering. Further, average precision value is increased by 7.6% after applying the post filtering to the ECB clustering approach.

As future work, we plan to apply our CAS based post-filtering approach to the big problems in service computing, issues such as discovery, recommendation and composition.

8. REFERENCES


### Authors

**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. Now he is a Ph.D student in School of Computer Science and Engineering, University of Aizu, Japan. His research interests include Semantic Web, Web Services and Their Composition, Web Data Mining, Web Service Discovery and Composition.

**Incheon Paik** received the M.E. and Ph.D degrees in Electronics Engineering from Korea University in 1987 and 1992, respectively. Now he is an associate professor in the University of Aizu, Japan. Research interests include Semantic Web, Web Services and Their Composition, Web Data Mining, Awareness Computing, Security for e-Business, and Agents on Semantic Web. He served several conferences as a program chair and program committee member for numerous international conferences. He is a member of IEEE, ACM, IEICE and IPSJ.

**Koswatte R. C. Koswatte** received the bachelor’s degree in 2006 from Sri Lanka Institute of Information Technology, Sri Lanka. She completed master degree in Information Management in 2010 from Sri Lanka Institute of Information Technology, Sri Lanka. Now she is a master student in School of Computer Science and Engineering, University of Aizu, Japan. Her research interests include Semantic Web, Ontology Learning and ICT Education.

**Wuhui Chen** received the bachelor’s degree in 2008 from Software College Northeast University, China. He received the master’s degree in 2011 and Ph.D degree in 2014 from School of Computer Science and Engineering, University of Aizu, Japan. His research interests include Service Computing, Big Data, and Social Computing.

http://hipore.com/ijsc

13