Journal of Web Engineering, Vol. 14, No. 3&4 (2015) 325–345

SEMANTIC-BASED CLUSTERING OF WEB SERVICES

LEONARDO DE JESUS SILVA
FORMAS/LASID/DCC/IM, Federal University of Bahia, Av. Adhemar de Barros, s/n, Ondina
Salvador, Bahia 40170-110, Brazil
leojssilva@hotmail.com

DANIELA BARREIRO CLARO
FORMAS/LASID/DCC/IM, Federal University of Bahia, Av. Adhemar de Barros, s/n, Ondina
Salvador, Bahia 40170-110, Brazil
dclaro@ufba.br

DENIVALDO CICERO PAVÃO LOPES
LESERC, Federal University of Maranhão Campus do Bacanga - CCET
São Luiz, Maranhão 40170-110, Brazil
dlopes@dee.ufma.br

Received April 7, 2014
Revised March 24, 2015

The interoperability needed for the exchange of information between organizations is currently obtained through Service Oriented Architecture (SOA). Through the implementation of Web services, components can be described in a consistent and standardized way. However, there is a high number of published Web services, it is important to have some organization in order to determine specific areas of these services. Thus, our approach proposes creating semantically similar domains of Web services by applying clustering algorithms. Two clustering algorithms were adapted and evaluated. Each cluster was validated in order to analyze whether the Web services were grouped correctly. Our results evaluated each group and the set of categorized services.

Keywords: Web Services, Clustering, Semantic-based Similarity
Communicated by: D. Schwabe & F. Vitali

1. Introduction

Several business processes have been developed by companies. Many of these are published as Web Service that are made available to users. Over the years, these services were characterized as islands of knowledge, without any integration among them. But with the advent of globalization, many companies have been merged and have given rise to a necessity to integrate these sectors which had previously been isolated even within the same organization. The integration of different industry sectors has been reflected with the use of technologies that already allow some interoperability, such as Web Services. This interoperability is guaranteed by SOA [4], because of the Web service standard format that facilitates the reuse and the integration of various systems. The World Wide Web Consortium (W3C)[4] defines a Web Service as *a software system designed to support interoperable machine to machine interaction over a network whose interface is described by machine-processable format.*

Companies frequently need to find new partners for the provision of their business processes. When using a Web Service format, a business process search allows new integration possibilities to be discovered. Authors in [15] list service portals, search engines and UDDI as mechanisms to find these Web services on the Internet. In [2], the authors investigated the sources of Web Services and they stated that the amount of services published through UDDI had diminished, while there was a growth in service portals and web crawling. In a UDDI repository only 53% of web services are active. On the other hand, using a search engine, the number of active web services obtained was 92%. However, researchers in [9] found that in a search for WSDL documents in Google, Yahoo and Bing only 12% of the results obtained were described using the WSDL language. Therefore, it is necessary to define a better way to organize Web Services published on the Internet so that a larger percentage of business processes can be continued and therefore the integration of information systems can take effect.

Regarding the discovery of services, search engines offer only a syntactic means to search, but making the discovery process difficult. In a syntactic search, business processes can be ambiguous, thus leading to undesirable recovering Web Services that are even not relevant for an organization. Given this scenario, Web services need to be described unambiguously in order to maximize the relevance for retrieving them. Some description languages of Semantic Web services have been proposed such as OWL-S[17], WSMO[19] and SAWSDL[14]. Other languages, also based on annotations, have been used such as microformats [12], WADL [7] and RDFa[1]. These languages allow Web Services to be described semantically relating their parameters to concepts in an ontology. This extends the possibility of disambiguation on discovering these services and consequently on recovering more relevant services for an organization.

However, the semantic Web description can facilitate the discovery process of Web Services, but it cannot organize them. In terms of increased competitiveness and the possibility of business process integration in an organization, it is necessary that similar services are grouped so that these organizations can minimize their discovery time and maximize the use of these services.

Thus, our approach aims to group semantically web services in order to make them available in clusters of domains. These clusters are automatically set according to each retrieved service. Experimentally, two clustering algorithms were adapted to semantic similarity concepts and some experiments were performed: Partitiong Around Medoids - PAM [11] and Agglomerative Hierarchical Clustering Algorithms [5]. The generated groups were evaluated by F-measure[10] and silhouette width[20]. The results showed the formation of semantically similar Web service groups thus, characterizing domain groups of similar services.

In an upper level, we can summarize two main contributions of this paper: (i) to better organize services based on their functional requirements leading to groups of domain and (ii) to facilitate the discovery of services in a given domain, allowing greater integration of business processes between companies. In a lower level, we can add two more contributions : (iii) we adapted Paolucci[18] and Sampler [21] filters to our semantic-based similarity measure to group our set of web services and (iv) we evaluate two algorithms based on their clustering results.

This paper is organized as follows: Section 2 presents related works. Section 3 discusses

clustering algorithms adapted in our work. Section 4 describes the similarity measure incorporated into both algorithms. Section 5 presents our methodology. Section 6 evaluates our proposed algorithms using precision, recall and F-measures. Silhouette value was also used to test our groups. Finally, section 7 describes the conclusions and some future works.

2. Related Work

Several works have been published whose purpose is to improve the efficiency of web services clustering. These works can be divided into two-folds: i) functional or non-functional requirements.

Some researchers have been only considered non-functional requirements to group semantic web services [24, 26]. Authors in [24] presented an approach to cluster web services based on their non-functional requirements. They stated that in order to better compose web services, they first need to have non-functional groups to diminish the search time for selecting a web service. Yi Xia et al. [26] also regard the need to attend non-functional requirements in service composition. The services that perform a common task are clustered conform to four QoS criteria (cost, execution time, reliability and success rate). Different from our proposal, none of them consider semantic-based criteria or functional requirements.

Elgazzar et al. [3] do web service clustering by tackling functional criteria. Their services are only described by WSDL (Web Service Description Language). Their main purpose is to extract some features from WSDL, such as content, types, messages, ports and web service name. In [25], the authors also carried out the same set of WSDL features. Additionally, each service is tagged by a user point of view. Each tag is included into a similarity measure that interferes into their separation group. Despite of both works tackle functional requirements, they do not have any semantic similarity approach to define each group.

The closest work to ours is concerned by [23]. The authors proposed to group OWL-S semantic web services. Although they analyze a degree of matching, they do not use Paolucci[18] approach, which is one of the most usage to matchmaking web services functional descriptions. On the other hand, they only present the PAM [11] algorithm without any evaluation. Different from our approach, we consider Paolucci [18] and Sampler [21] filters. For a better analysis over the clustering approach, we deeply evaluate two algorithms and compare each results group.

3. Clustering algorithms

Clustering algorithms group a collection of objects, distinguishing these objects according to similarity: dissimilar objects belong to an intergroup while similar objects belong to an intragroup.

Clustering algorithms are usually classified into two categories: *Hierarchical Clustering* and *Partitional Clustering* [27]; [6]. Hierarchical clustering is divided into two types: Divisive Hierarchical and Agglomerative Hierarchical. In the divisive hierarchical approach, the hierarchical groups are formed in a top-down manner, that is, the algorithm begins with a single group of all objects and after each iteration the groups are split. In agglomerative hierarchical, groups are formed in a bottom-up manner, i.e. each object belongs to a group and the groups are joined on each iteration. Unlike hierarchical clustering, partitional clustering objects are directly divided into k groups defined by a user without a hierarchical structure,

i.e. the number of groups can be specified manually.

In the next two subsections, both algorithms used for grouping Semantic Web Services are presented. The first is an agglomerative hierarchical clustering algorithm and the second is a partitional clustering algorithm. We decided to use one algorithm of each category to better evaluate their evolution of web services grouping semantic.

3.1. *Agglomerative Hierarchical Clustering Algorithm*

The category of agglomerative hierarchical clustering algorithms forms a hierarchical structure of groups, starting with groups with only one object. In this study we have determined the threshold level of the hierarchy, i.e. a predefined number of groups. Thus, the algorithm starts with **M** groups, where each group contains a single object and stops when it reaches the **k** groups specified in the input (Algorithm 1).

Algorithm 1 Agglomerative Hierarchical Clustering Algorithm

Input:

- k: number of groups to be formed.
- D: data set of **N** objects.

Output: k groups.

1: Start with M groups (each group with an object). Calculate the proximity matrix for all M groups.
2: **repeat**
3: Find the minimal distance $D(C_i, C_j) = \min_{m \neq l}^{1 \leq m, l \leq M} D(C_m, C_l)$ where D(*,*) is the distance function and combines the groups C_i and C_j to form a new group.
4: Update the proximity matrix to calculate the distances among the new group.
5: **until** form k groups

Regarding the definitions to calculate the distance between two groups, there are several approaches[5]:

- *single connection* - the distance between two groups is determined by the two closest objects in different groups.
- *complete connection* - the distance between two groups is determined by the two most distant objects in different groups.
- *group average (UPGMA - unweighted pair-group method using the average)* - the average distance of all possible pairs of objects that are composed of one object from each group.
- *centroid (UPGMC - unweighted pair-group method using the centroid)* - is the Euclidean distance squared between vectors average (centroid).
- *median (WPGMC - weighted pair-group method using the centroid)* - it is similar to the centroid method, except by the centroids of the constituencies that are equally weighted to produce a new centroid of the merged group.
- *connection weighted average (WPGMA - weighted pair-group method using the average)* – it is similar to the group average measure, but the distance weighted between groups is inverse to the number of objects in each group.

- *ward* - it is defined as the merger of two groups based on the standard error (sum of Euclidean distances). The objective is to minimize error squared for each group on each iteration.

In order to better determine which distance approach might be used by our algorithm, we performed a set of tests with each measure to evaluate each distance in semantic web services scenario. All these tests are described in Section 6. The best method was selected for use by our algorithm.

3.2. *Partitioning Around Medoid - PAM*

Partition Around Medoids (PAM) (Algorithm 2) was first described by [11] based on searching representative objects in a data set. This algorithm searches for k representative objects in a data set. In PAM algorithm, representative objects are called group medoids. After finding a set of k representative objects, the k clusters are constructed by assigning each object to the nearest representative object.

The algorithm has two phases. First a **phaseBuild** selects k central objects to be used as initial medoids. Second, **phaseSWAP** seeks to reduce an objective function by exchanging a representative object with an unselected object. If this reduction is possible, then the exchange is performed. When the objective function can no longer be reduced, the iteration stops.

Algorithm 2 PAM

Input:

- k: number of groups,
- C: data set of n objects.

Output: Set of k groups.

```
1: phaseBUILD()
2: phaseSWAP()
```

The procedure **phaseBUILD()** (Algorithm 3) selects the first object where the sum of the dissimilarity is the lowest one. Subsequently, another object is selected. This object is the one that reduces the objective function as much as possible. The variable cost represents the loss of maintaining a previous medoid j instead of exchanging it for i; D is a vector where the position l indicates the shortest distance between the unselected object I and the set of selected objects; $candidateMedoid$ indicates the possible medoid until a loop breakpoint and $d(i, j)$ denotes the distance between the objects i and j.

Algorithm 3 Build phase

```
1: procedure phaseBUILD( )
2:     o = Object with lowest dissimilarity measure among the other objects
3:     addMedoid(o)
4:     repeat
5:         for (object i non-selected from C) do
6:             cost = 0
7:             costActual = −∞
8:             for (object j non selected from C, j ≠ i) do
9:                 cost+ = max(D_j − d(i, j), 0)
10:            end for
11:            if (cost > costActual) then
12:                costActual = cost
13:                candidateMedoid = i
14:            end if
15:        end for
16:        addMedoid(candidateMedoid)
17:    until kmedoids
18: end procedure
```

The procedure **phaseSWAP()** (Algorithm 4) presents the vector E. This vector keeps the position l that indicates the second smallest distance from an unselected object l among the **k** selected objects. The *exchange* variable represents the gain on exchanging j for i when its value is negative.

Although the most popular partitional algorithm is the k-means [8, 27], this algorithm uses a numerical mean to define its group center. In PAM algorithm, it is no longer necessary to convert a Web Service value because PAM defines the most representative object to be the group center. Moreover, PAM avoids the effect of outliers[8, 27].

4. Our semantic-based similarity measurement

Our semantic-based similarity measure aims to determine the distance among web services in a collection. In this work, each Web service ($ws_i, i = 1, .., n$, where n is number of web services in a collection) is described using OWL-S language[17]. Only OWL input ($ws_i.\overrightarrow{I}$) and OWL output ($ws_i.\overrightarrow{O}$) parameters are used.

- $ws_i.\overrightarrow{I}$ - Input parameter vector of a Web Service i.
- $ws_i.\overrightarrow{O}$ - Output parameter vector of a Web Service i.

Each position in parameter vector represents a concept in an OWL ontology. In order to establish a similarity between these concepts, the degrees of similarity proposed by Paolucci[18] were used. The four degrees defined by Paolucci[18] are described below including the sibling degree, introduced by [21]:

- *Exact* - two outputs are equal or an output is a direct subclass of a request output.
- *Plugin* - the output of a service is a subclass of a request output.

Algorithm 4 Swap phase

```
1: procedure PHASESWAP( )
2:     for (object i selected from C) do
3:         for (object h non selected from C) do
4:             change = 0
5:             changeActual = +∞
6:             for (object j non selected from C, j ≠ h) do
7:                 if (d(j,i) > D_j) then
8:                     if (d(j,h) < D_j) then
9:                         change+ = d(j,h) − D_j
10:                    end if
11:                else
12:                    if (d(j,h) < E_j) then
13:                        change+ = d(j,h) − d(j,i)
14:                    else
15:                        change+ = E_j − D_j
16:                    end if
17:                end if
18:            end for
19:            if (change < 0) then
20:                exchange(i,h)
21:                restarts the loop for the selection of i
22:            else
23:                break
24:            end if
25:        end for
26:    end for
27: end procedure
```

- *Subsumes* - the request output is a subclass of the service output.
- *Sibling* - two outputs have a common direct superclass.
- *Fail* - no semantic relationship is found between these outputs.

The average dissimilarity (Eq. 1) is calculated between two web services input parameters. For each filter, a weight was assigned, such as: 1.0 - Exact, 0.75 - Plugin, 0.5 - Subsumes, 0.25 - Sibling, 0 - Fail. These weights are analogous to output parameters.

$$avgD(ws_i.\overrightarrow{I}, ws_j.\overrightarrow{I}) = \frac{\sum_{l=1}^{r}(1 - maxSimilarity(ws_i.I_l, ws_j.\overrightarrow{I}))}{r} \quad (1)$$

where r and s are vector sizes of $ws_i.\overrightarrow{I}$ and $ws_j.\overrightarrow{I}$, respectively, and $r < s$. The function $maxSimilarity(ws_i.I_l, ws_2.\overrightarrow{I})$ indicates the maximum similarity between a service parameter $ws_i.I_l$ and a second service parameters in the vector $ws_j.\overrightarrow{I}$.

In Eq. (2), the dissimilarity measure is presented between two web services ws_i and ws_j.

$$\begin{aligned} D(ws_i, ws_j) = avgD(ws_i.\overrightarrow{I}, ws_j.\overrightarrow{I}) * 0.4 \\ + avgD(ws_i.\overrightarrow{O}, ws_j.\overrightarrow{O}) * 0.6 \end{aligned} \quad (2)$$

In [18], services are ordered according to the correspondence degree of the request output parameter with the service output parameters. The priority order starts with the exact degree, followed by plugin, subsumes and sibling. The input parameters are ordering as a second criterion while the output parameters are ordering as the first one. The authors stated that if the input parameter of a request and a service do not match, one can replace this web service with another that has the desired output parameters. Consequently, a higher weight was assigned to output parameters (Eq. 2).

When the algorithms were executed, the services were grouped into similar domains to be further published over the Internet. An analysis of the types of semantic clusters was performed in order to validate our proposal.

5. Methodology

Our methodology can be divided into two parts. First, we evaluate our algorithms. We decided to use one algorithm from each category. From the partitional category, the PAM was the most common and popular algorithm found in the literature [8, 16]. From Hierarchical category, there are a set of distance measures that differentiate each algorithm. Thus, we decided to evaluate each one in order to select the best measure used into a hierarchical approach.

In order to evaluate each algorithm, including PAM, we calculate the silhouette for each one of them with two different values to k numbers of clusters as detailed below. Thus, the two best silhouette values were chosen for our algorithms: PAM and Ward.

The authors in [20] describe silhouette as a graphical validity where "each cluster is represented by a so-called silhouette, which is based on the comparison of its tightness and separation. The silhouette shows which objects fit well within each cluster, and which ones are merely somewhere in between clusters". The silhouette measure might determine a 'natural' number of clusters and as a consequence it can provide a clustering validity.

Our second part of experiments corresponds to an evaluation of our generated groups. In order to decide the number of groups, we carried out two tests. First, we analyzed the Test Collection of Web services dataset and we extracted its predefined number of groups. In the Test Collection dataset, the number of predefined groups is 7. Secondly, we tried to manually analyze the Test Collection dataset. We manually categorize each web service into groups. Thus, we achieved 15 groups manually for the Test Collection dataset. Considering both number of groups, we performed our evaluations taking $k = 7$ and $k = 15$.

Table 1 shows the relationship between 7 and 15 groups in function of web services. The *economy* domain was mainly split among 6 domains of the 15 groups of classifications.

Table 1. Classification 7 and 15 groups

	communication	weapon	travel	medical	food	education	economy	total
location	0	0	16	0	0	0	0	16
various	1	0	6	10	1	4	11	33
employee	0	0	0	0	0	92	0	92
entertainment	55	0	0	0	0	0	1	56
devices	0	0	0	0	0	0	31	31
vehicles	0	0	0	1	0	6	92	99
weather	0	0	41	0	0	0	0	41
government	0	4	0	0	0	5	0	9
weapon	0	17	0	0	0	0	0	17
research	0	0	0	0	0	91	2	93
food	0	0	1	0	29	5	49	84
medical	0	0	0	62	0	1	0	63
travel	0	0	100	0	0	1	2	103
book	1	0	0	0	0	55	111	167
drink	0	0	0	0	3	6	46	55
total	57	21	164	73	33	266	345	959

The groups generated by PAM and Ward are classified according to each F-measure (Table 1). A group is labeled based on its highest F-measure value. Considering the groups, *Group 1* and *Group 2* are generated automatically by our algorithms and *Medical* and *Financial* labels are defined manually (Table 2). *Medical* can label *Group 2* and *Financial* can label *Group 1*. However, we can have other situations where two or more labels can be associated with the same group. Although the authors in [16] labeled their groups based on precision values, our work considers a balanced value between precision and recall, called F-measure. This can increase the correction of labeling a group.

Table 2. Example of classification

	Medical	Financial
Group 1	0.6	0.9
Group 2	0.7	0.8

Our experiments are presented in the next section so as to validate our proposal.

6. Validation of our semantic-based similar clusters

In order to validate our approach, we did a set of experiments which are described in this section.

6.1. *Environment Configuration*

In order to validate our semantic-based clusters proposal, some experiments were performed using a computer with the following configuration: Intel I5 2.2 Ghz 64 bit, 4 GB of RAM and Windows 8 64-bit OS. A Web server was also used to publish our web services [22]. The web services used were extracted from the OWL-S TC version 2.2 [13]. As this collection describes the web services using both versions 1.0 and 1.1 of OWL-S and none of them was the latest version of OWL-S, we converted all of them into OWL-S 1.2 format. Our new dataset has 959 semantically described web services format in OWL-S 1.2 version. All algorithms were developed using Java language JVM 1.6_0_37.

6.2. *Experiments*

6.2.1. *The average Silhouette value of clustering*

The Silhouette value was applied to both category of algorithms used in this work: the PAM algorithm and all variants of hierarchical algorithm described in section 3.1. As illustrated in Figure 1, PAM and Ward algorithms achieved the highest Silhouette values, for $k = 7$ (0.218 and 0.141 respectively) and $k = 15$ (0.265 and 0.3, respectively). The complete approach slightly diminished from 0.082 to 0.081, while Ward approach nearly doubles its average Silhouette value when augmenting the number of clusters.

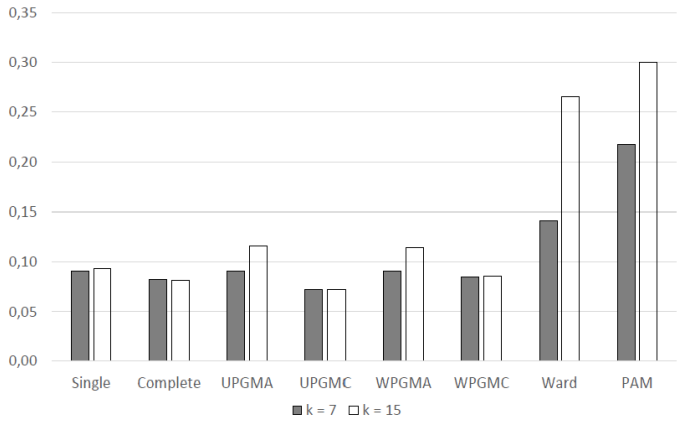

Fig. 1. Silhouette Global Average

Considering both Silhouette values for PAM and Ward varying from $k = 7$ to $k = 26$, we can observe that the Ward approach had an increase growth. For $k = 19$, there is a slowdown in the improvement of Ward approach and a stabilization of PAM approach, varying their Silhouette value between 0.35 and 0.40. Taking these Silhouette values and the number of manually defined groups, the potential 'natural' groups could be between $k = 19$ and $k = 26$ (Figure 2). However, for a better proof of concepts, we evaluate our groups taking $k = 7$ and

$k = 15$.

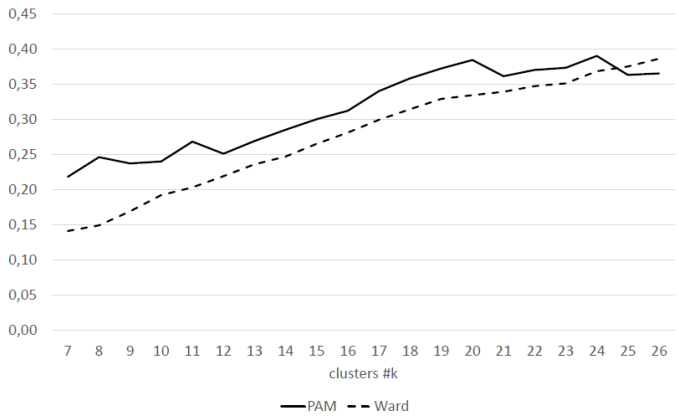

Fig. 2. Silhouette values for $k = 7$ to $k = 26$

6.2.2. *The F-Measure of clusters and classification for $k = 7$*

Since PAM and Ward algorithms had higher values for the Silhouette measure, a deeper analysis was carried out. First, each group was named based on F-measure values. The group received a label in accordance to its highest F-measure value. Figure 3 and Figure 4 provide such example.

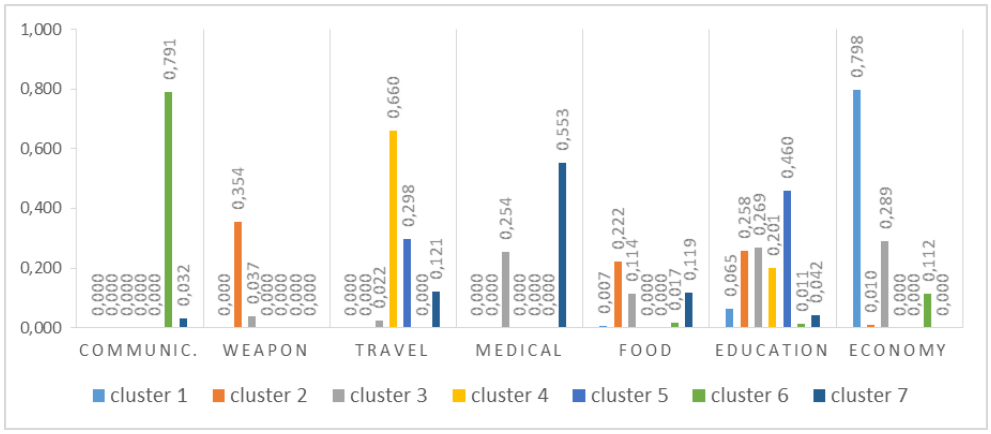

Fig. 3. F-measure PAM ($k = 7$)

In Figure 3, the clusters 1, 4, 5, 6 and 7 received respectively the labels *economy*, *travel*,

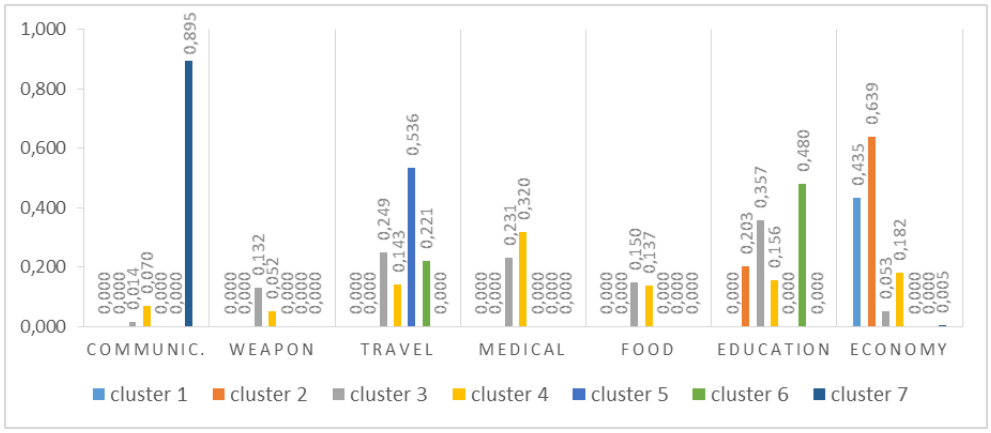

Fig. 4. F-measure Ward ($k = 7$)

education, *communication* and *medical*. The domains *weapon* and *food* received its highest F-measure for cluster 2. For the cluster 3, no label was defined.

In Figure 4, the clusters 7, 5, 4, 6 and 2 received respectively the labels *communication*, *travel*, *medical*, *education* and *economy*. Once more, the domains *weapon* and *food* stayed within the same cluster 3. This time, cluster 1 was not labeled. Doing a manual analysis, we stated that almost all web services were related to economical domain (approximately 20%).

It can be observed in Figure 3 and Figure 4 that some domains tend to be more disperse among groups independently of the algorithm that is used, e.g. *education*.

Analyzing each group and each algorithm, we can highlight some strength of our algorithms. In relation to precision (Figure 5), PAM algorithm has a higher performance in 3 groups (*weapon/food*, *medical* and *economy*), although *Ward* has a higher performance in others (*communication*, *travel* and *education*).

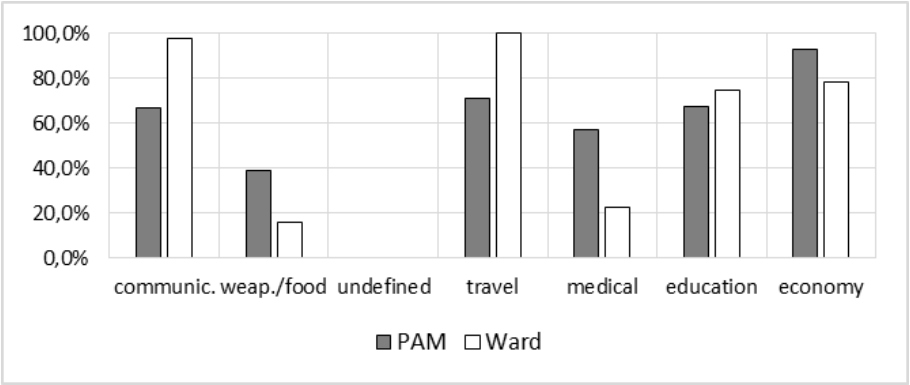

Fig. 5. Precision (k=7)

Considering the recall measure, for *medical and education* domains, there is a balance between both algorithms, as shown in Figure 6.

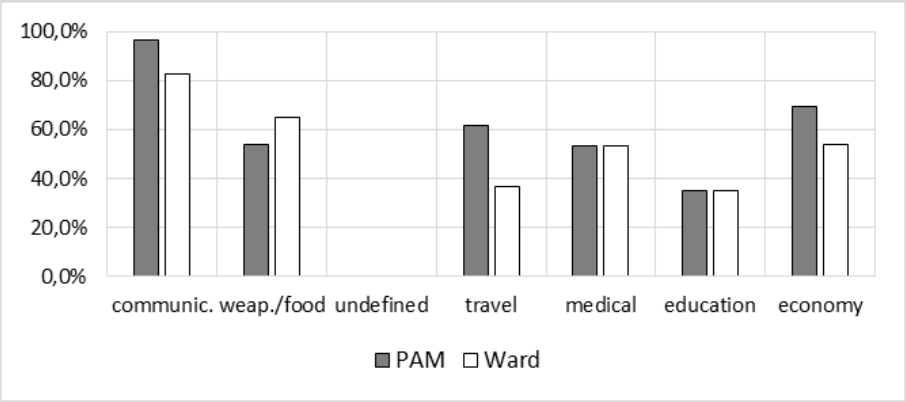

Fig. 6. Recall (k=7)

Comparing precision values (Figure 5) with the Silhouette values (Figure 7), it can be observed that both groups *communication* and *travel* have both high values. However, there is no relationship between those values.

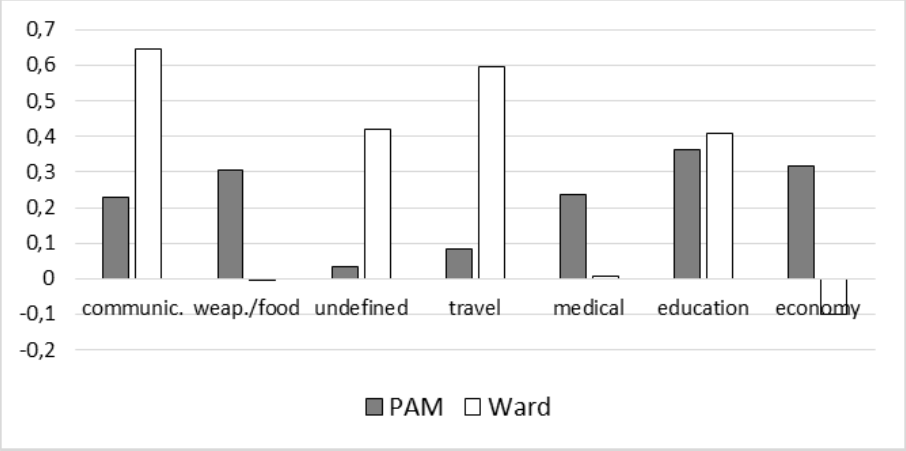

Fig. 7. Groups Silhouette values (k=7)

The *economy* group received a negative value for the Silhouette measure. This was due to a misclassification within the Ward algorithm. From 237 services, 166 of them would be better classified in an *undefined* group. Only 66 services were correctly classified.

Figure 8 presents a comparison for each group and we can observe (a) the quantity of services identified only by PAM (b) the quantity of services identified by both algorithms and (c) the quantity of services identified by Ward. Within this experiment, it is possible to observe how similar is a group using both algorithms, i.e. *Communications* group formed by PAM in relation to Ward.

The *communications* and *travel* groups formed by Ward have all the services contained in

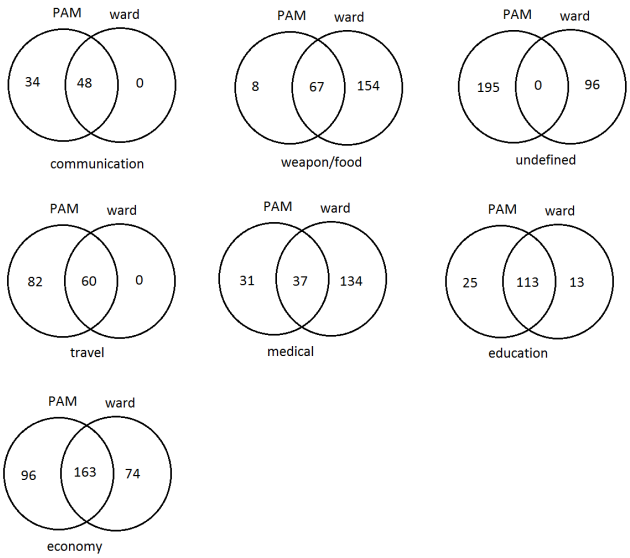

Fig. 8. Comparison between **PAM** and **Ward** groups (k=7)

their respective groups formed by PAM. In Figure 5, we note that these groups have higher precisions.

In groups *arms/feeding* and *medical*, the number of services found by Ward was much higher than those found using PAM algorithm. However, we observed the opposite with Ward in that it obtained a very low precision and a low similarity between groups.

In the group that was not given a label there was no similarity between our two algorithms proposed.

The group *education* had 151 services grouped respectively 138 and 126 web services by PAM and Ward algorithms. Comparing both groups, 113 web services were presented in both groups, that is, the most similar between the two groups of algorithms.

The group *economy* was saving a considerable amount of 163 of the same services grouped by our algorithms. But the sum of the different services grouped by the algorithm reached 170.

Our proposed similarity measure and the PAM algorithm was feasible for clustering semantic web services. The *weapon* and *food* domains have only 21 and 33 services respectively, which may have difficulty for creating a specific group. Using the Ward hierarchical algorithm, we can state that in general this algorithm performs worse than the PAM algorithm. We could observe it by analyzing the F-measure and Silhouette values.

Thus our experiments allowed us to verify the effectiveness of our proposed similarity measure and the algorithms that were used.

6.2.3. *The F-measure of clusters and classification for* $k = 15$

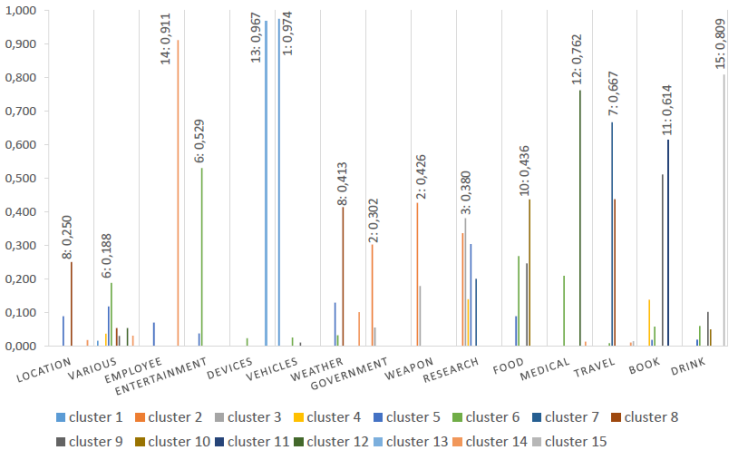

Fig. 9. F-measure PAM (k=15)

In Figure 9, the clusters 1, 3, 7, 10, 11, 12, 13, 14 and 15 received respectively the labels *vehicles*, *research*, *travel*, *food*, *book*, *medical*, *devices*, *employee* and *drink*. The domains *government* and *weapon* received their highest F-measure value for cluster 2, *location* and *weather* their highest values for cluster 8, *various* and *entertainment* for cluster 6 and clusters 4, 5, 9 had not any defined label. Groups 4 and 9 had 59% and 67% of their web services of label *book*, while group 5 had 42% of its web services labeled as *research*.

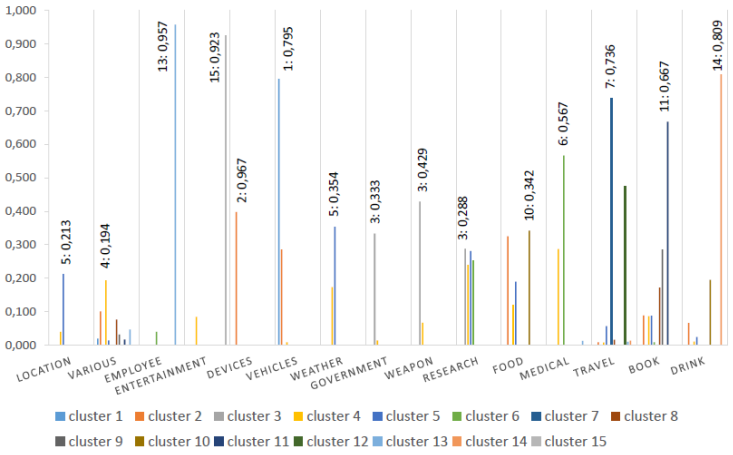

Fig. 10. F-measure Ward (k=15)

In Figure 10, the clusters 1, 2, 4, 6, 7, 10, 11, 13, 14 and 15 received respectively the labels *vehicles*, *devices*, *various*, *medical*, *travel*, *food*, *book*, *employee*, *drink* and *entertainment*. The domains *government*, *weapon* and *research* received their highest F-measure for cluster 3, *location* and *weather* their highest values for cluster 5, and clusters 8, 9, 12 had not any defined label. Groups 8 and 9 had 84% and 97% of their web services fit on *book* label group while group 12 had its entire web services belonging to label *travel*.

The group *various/entertainment* of PAM were labeled only as *entertainment* and the group *government/weapon/research* of Ward algorithm was labeled only as *government/weapon*.

Figure 11 depicts that PAM had achieved better precision values than Ward algorithm, especially when dealing with *location/weather*, *devices*, *food* and *medical* domains. While Ward algorithm had obtained its better performances when dealing with *travel*, *entertainment* and *employee* domains.

Considering recall values (Figura 12), there is a major balance between both algorithms. Almost all groups had more than 50% of recall values. The group *employee* was one of the best defined groups for both algorithms, considering precision and recall values.

In Figure 13, we can observe a relation between the Silhouette values and the Precision

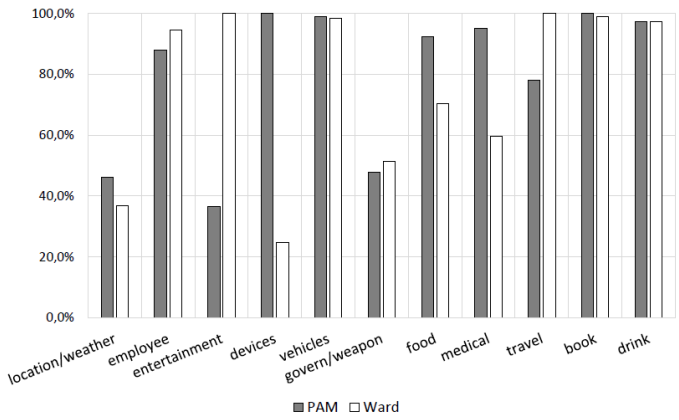

Fig. 11. Precision (k=15)

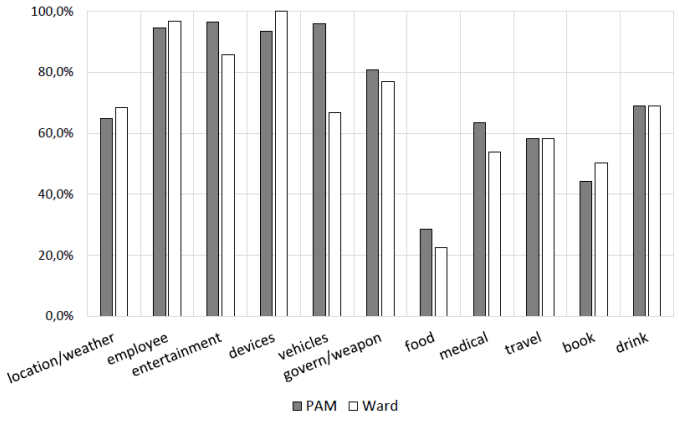

Fig. 12. Recall (k=15)

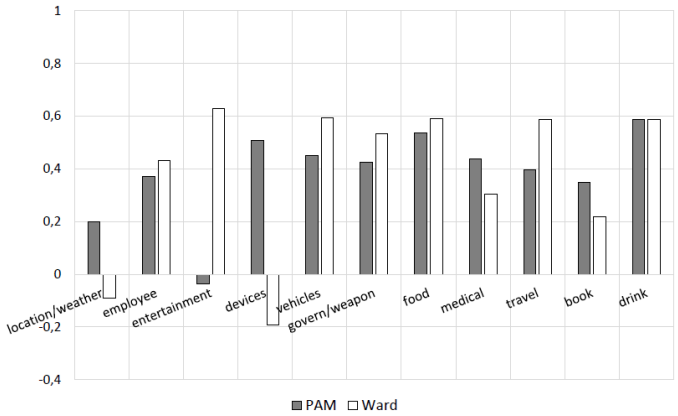

Fig. 13. Groups by the Silhouette values (k=15)

values. For instance, the PAM algorithm had its worst Silhouette value for the *entertainment* group as well as it had its worst value for the precision. On the other hand, the Ward algorithm obtained its worst values of Silhouette as well as its worst precision values for the groups *location/weather* and *devices*. Although the *book* group had a good precision value for both algorithms, it was not the same for Silhouette values.

Figure 14 depicts that *employee*, *government/weapon* and *drink* groups for both algorithms fits almostly the same services. The opposite occurs for the *Food* group, where the web services were completely different for both algorithms.

For *entertainment*, *vehicles* and *travel* groups, the web services found with Ward algorithm were the same using PAM algorithm. Moreover, PAM algorithm found more services with 100% of precision fitting the *vehicles* group. However, its performance was worse than Ward algorithm, considering the other groups.

Ward algorithm had found more services for *devices* and *medical* groups than the PAM algorithm, but analyzing the precision value, PAM algorithm had a better performance.

The number of web services in a group which was not labeled is presented in Table 3. This can confirm that the Ward algorithm had minor number of web services unlabeled than PAM algorithm. All web services of Group 9 and 5 web services of Group 8 using Ward algorithm were included into the Group 9 of PAM algorithm.

Table 3. Size of unlabeled groups

	PAM		WARD
undefined 4	22	undefined 8	19
undefined 5	52	undefined 9	29
undefined 9	103	undefined 12	32
	177		80

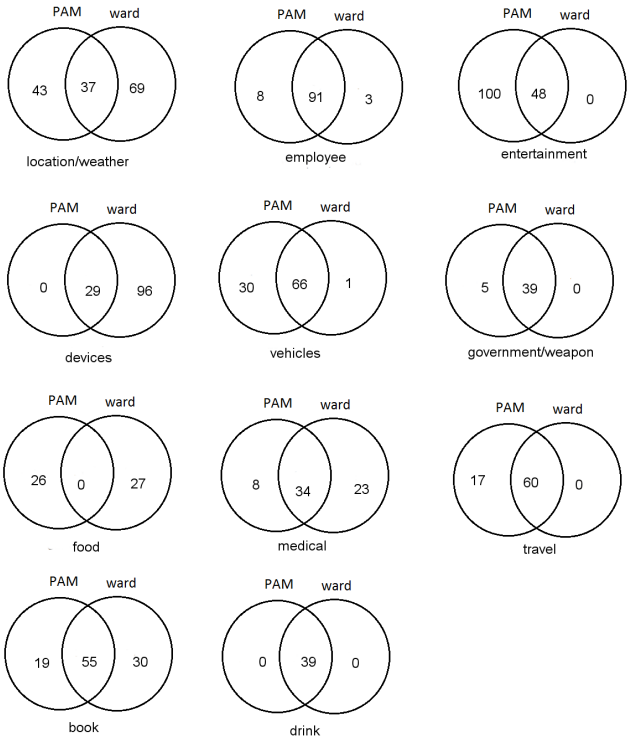

Fig. 14. Comparison between **PAM** and **Ward** groups (k=15)

We can conclude that the PAM algorithm had better performance when dealing with labeled groups. However, it had a higher value of unlabeled group. Thus, considering that there are few outlier web services, the Ward algorithm had a better performance leading with labeled and unlabeled groups.

7. Conclusion and Future Work

In this work we have presented a way to group semantic OWL-S web services. This grouping is shown as complex services from different domains which may have the same or similar concepts making an artificial reduction of the distance between two groups. Experiments were carried out using Silhouette and F-measure values. With the Silhouette measure it was possible to identify the quality of the formed groups and F-measure provides a breakdown of the groups. Through this work, we deeply analyzed the PAM algorithm and the Hierarchical Ward algorithm with the possibility of creating groups of web services fitting similar semantic.

As a future work, we plan to better analyze the merging among groups, using a fuzzy clustering approach.

Acknowledgements

This research work was supported by CNPq through the Grant $560231/2010-5$.

References

1. B. Adida, M. Birbeck, S. McCarron, and S. Pemberton. RDFa in XHTML: Syntax and processing, October 2008.
2. E. Al-Masri and Q. H. Mahmoud. Investigating web services on the world wide web. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 795–804, New York, NY, USA, 2008. ACM.
3. K. Elgazzar, A. E. Hassan, and P. Martin. Clustering wsdl documents to bootstrap the discovery of web services. In *ICWS*, pages 147–154. IEEE Computer Society, 2010.
4. T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.
5. B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis.* Wiley Publishing, 4th edition, 2009.
6. G. Gan, C. Ma, and J. Wu. *Data clustering - theory, algorithms, and applications.* SIAM, 2007.
7. M. J. Hadley. Web application description language (wadl). Technical report, Mountain View, CA, USA, 2006.
8. J. Han. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
9. O. Hatzi, G. Batistatos, M. Nikolaidou, and D. Anagnostopoulos. A specialized search engine for web service discovery. In C. A. Goble, P. P. Chen, and J. Zhang, editors, *ICWS*, pages 448–455. IEEE, 2012.
10. N. Jardine and C. J. van Rijsbergen. The use of hierarchical clustering in information retrieval. *Information Storage and Retrieval*, 7:217–240, 1971.
11. L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis.* John Wiley and Sons, New York, 1990.
12. R. Khare and T. Çelik. Microformats: a pragmatic path to the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 865–866, New York, NY, USA, 2006. ACM.
13. M. Klusch and P. Kapahnke. Owl-s service retrieval test collection, February 2005.

14. J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6):60–67, Nov. 2007.
15. J. Ma, Y. Zhang, and J. He. Efficiently finding web services using a clustering semantic approach. In *Proceedings of the 2008 International Workshop on Context Enabled Source and Service Selection, Integration and Adaptation: Organized with the 17th International World Wide Web Conference (WWW 2008)*, CSSSIA '08, pages 5:1–5:8, New York, NY, USA, 2008. ACM.
16. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
17. D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. Owl-s: Semantic markup for web services. Internet [http://www.w3.org/Submission/2004/SUBM- OWL-S-20041122/], 2004.
18. M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 333–347, London, UK, UK, 2002. Springer-Verlag.
19. D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. *Appl. Ontol.*, 1(1):77–106, Jan. 2005.
20. P. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, Nov. 1987.
21. J. J. Samper, F. J. Adell, L. van den Berg, and J. J. Martinez. Improving semantic web service discovery. In *Journal of Networks*, 2008.
22. K. Seidler and K. Vogelgesang. Apache friends xampp, February 2013.
23. M. K. W. Abramowicz, K. Haniewicz and D. Zyskowski. Design of web services filtering and clustering system. *International Journal On Advances in Internet Technology*, 1(1), 2008.
24. X. Wang, Z. Wang, and X. Xu. Semi-empirical service composition: A clustering based approach. In *ICWS*, pages 219–226. IEEE Computer Society, 2011.
25. J. Wu, L. Chen, Z. Zheng, M. R. Lyu, and Z. Wu. Clustering web services to facilitate service discovery. *Knowl. Inf. Syst.*, 38(1):207–229, 2014.
26. Y. Xia, P. Chen, L. Bao, M. Wang, and J. Y. 0001. A qos-aware web service selection algorithm based on clustering. In *ICWS*, pages 428–435. IEEE Computer Society, 2011.
27. R. Xu and D. Wunsch, II. Survey of clustering algorithms. *Trans. Neur. Netw.*, 16(3):645–678, May 2005.