Weighted Fuzzy Clustering for Capability-driven Service Aggregation

Christoph Dorn, Schahram Dustdar
Distributed Systems Group
Vienna University of Technology, 1040 Vienna, Austria
lastname@infosys.tuwien.ac.at

***Abstract*—Workflow design, mashup configuration, and composite service formation are examples where the capabilities of multiple simple services combined achieve a complex functionality. In this paper we address the problem of limiting the number of required services that fulfill the required capabilities while exploiting the functional specialization of individual services. Our approach strikes a balance between finding one service that matches all required capabilities and having one service for each required capability. Specifically, we introduce a weighted fuzzy clustering algorithm that detects implicit service capability groups. The clustering algorithm considers capability importance and service fitness to support those capabilities. Evaluation based on a real world dataset successfully demonstrates the effectiveness and applicability for service aggregation.**

***Keywords*-fuzzy clustering, service capabilities, service aggregation**

I. Introduction

Service aggregation describes the process of selecting a set of services to exploit their combined functionality. Applications such as designing a service mashup, developing a composite service, or determining services for a workflow require the selection and aggregation of suitable services. Existing work applies advanced mechanisms such as ontologies [1], goal-models [2], context-based mediation [3], or QoS metrics [4] to determine the optimal set of services that provide the required functionality. These approaches can be roughly classified according to two extreme categories. Either each service is expected to provide exactly one of the operations or a single service provides all operations.

In this paper we present an approach to service aggregation that resides between those extremes. The problem is finding implicit functional groups that are prevalent amongst the available services. An implicit functional group describes a set of operations that tend to be provided together on a service instance. One major challenge is automatically extracting those groups. Simultaneously, we also need to address the importance of each function in the overall aggregation as well as the actual support of each operation by the set of available service instances.

We apply the concept of capabilities to abstract from the technical service specification. Capabilities are meta data about a service that go beyond the pure interface description (WSDL) but remain more specific than QoS attributes. A storage service, for example, provides an *upload* operation and comes with corresponding capabilities that specify the maximum allowed file size and file count.

We match required service capabilities (i.e., constraints) and available capabilities to obtain scores for each capabilities. These scores are then clustered to extract the implicit capability groups. After ranking the services in each cluster, we can select the top scoring services for the aggregation. The main focus, however, lies on the customization of the clustering algorithm. Our main contribution is a weighted fuzzy c-means (WFCM) clustering algorithm that is able to intelligently consider the importance and support of individual capability constraints when generating clusters.

We evaluated our clustering algorithm using a real world data set. The weighted fuzzy clustering technique provides consistently more sensible cluster results than the baseline, non-weighted fuzzy c-means (FCM) clustering technique.

The remainder of this paper is structured as follows. Section II presents a motivating scenario to outline the problem and challenges in more detail. Section III outlines the mechanisms for data transformation in preparation for clustering. Section IV details the weighted fuzzy clustering algorithm. Section V presents the results of the clustering algorithm when applied to a real world data set. Section VI discusses related work before Section VII concludes with an outlook on future work.

II. Motivating Scenario

The advantage of extracting implicit capability groups instead of searching for a single, optimal mapping of capabilities to services is *replacability*. A capability group represents a considerable number of similarly structured services as otherwise the clustering algorithm would not determine such a group. This allows for run-time replacement of services within a group, while leaving the structure of the remaining service aggregation untouched.

We expect that capabilities are not uniformly distributed across services. A few capabilities will be available on most services, while other capabilities are very specific and provided only by a subset. Fuzzy clustering allows us to assign those generic capabilities to every cluster with some extent, while hard clustering (e.g., K-Means Clustering) would require to assign such capabilities to exactly one cluster.

The clustering algorithm needs to become aware of capability importance (i.e., constraint weights) and capability support (i.e., how well are capabilities provided by the service instances in terms of measured utility). Capability importance enables to define vital and optional capabilities. The clustering technique should produce clusters of specific functionality, rather than creating two clusters: one containing all specific capabilities and the one containing all generic capabilities. Also, when encountering situations where some capabilities are hardly supported by available services, we want to avoid having two clusters emerge: one around the well supported capabilities, and one containing all unsupported capabilities. The following scenario outlines these effects based on example data.

Let us suppose we have a set of 13 capabilities. Figure 1a prints the utility values for each capability for 2 exemplary services for sake of clarity.[1] Capabilities 1 to 8 are well supported, while the remaining 5 capabilities are hardly supported.

Ideally, the clustering algorithm generates two clusters (Figure 1e), one for capabilities 1 to 4 and another one for capabilities 5 to 8. The remaining capabilities should be equally assigned to both clusters. Regular FCM clustering, however, produces the cluster configurations as provided in Figure 1b-d, all of them unsuitable for efficient service aggregation.

III. Data Preprocessing

The capability data model, specification of capability requirements, and the matching against capability profiles is described in previous work [5]. For the scope of this paper, it is sufficient to assume that there exists a matching function *eval* that maps each service $s \in \mathcal{S}$ and required capability $c \in C$ to a utility value $u_{i,n}$ in the interval $[0, 100]$, where 100 is the maximum score. The resulting utility matrix $\mathcal{U}$ contains for every combination of capability c_i and service s_n the corresponding utility value. Each row x_i in $\mathcal{U}$ describes the feature vector of capability c_i used for clustering.

The overall impact ω_i of capability c_i during the clustering process is given by the initial capability requirement importance τ_i (with $\sum \tau_i = 1$), and the support by the available services as measured by the fulfillment metric f_i.

$$\omega_i = \frac{f_i * \tau_i}{\sum f_i * \tau_i} \qquad where \qquad f_i = \frac{\sum_n u_{i,n}}{\sum f_i} \tag{1}$$

Subsequently we ensure that services determine the clustering result proportional to their utility. To this end, we transform the utility matrix ($\mathcal{U}$) before clustering to reflect the preliminary service rank $r_n = \sum_i u_{i,n} * \tau_i \;/\; \sum r_n$. We multiply each $u_{i,n}$ with the service rank $r_n \in \mathcal{R}$ and renormalize the matrix such that services with average utility $u_i = \bar{u}$ maintain their utility value ($\mathcal{U}_\omega = \mathcal{U} \times \mathcal{R} \times |\mathcal{R}|$). We thereby exploit the FCM's sensitivity towards outliers. After the transformation, the weighted utility matrix $\mathcal{U}_\omega$ contains higher utility values for better ranked services compared to lower ranked services. Hence, above average service will have more impact during the clustering process as described in the following section.

[1] For visualizing n services, we would need an n-dimensional figure.

IV. Capability Clustering

A. Fuzzy Clustering Principles

The underlying principle in fuzzy clustering is assigning data elements to multiple clusters with varying degree of membership. Algorithm 1 details the steps to obtain the best clustering for a particular number of clusters z. We refer to the listing when explaining the individual steps.

For our problem, the basic Fuzzy C-Means (FCM) [6] associates each capability feature vector x_i with every cluster $k_j \in \mathcal{K}$. The membership table $\mathcal{M}_{ij}$ describes the degree of capability x_i belonging to a particular cluster k_j, such that $\sum_j \mu_{ij} = 1$. Elements close to the cluster center (i.e., the centroid) have higher membership values for that particular cluster than elements farther away.

The clustering algorithm's objective is minimizing the overall distance of data elements to the cluster centers (line 11). This *within-class least squared-error function* is defined as:

$$J_m = \sum_{i=1}^{|C|} \sum_{j=1}^{|\mathcal{K}|} \mu_{ij}^m * \|x_i, k_j\|^2 \tag{2}$$

where $m > 1$ is the fuzzy factor (discussed later) and $\| \bullet \|$ is a distance measurement between data element x and the cluster center k. FCM iteratively recalculates cluster centers (line 7) and membership degree (line 8) until the objective function converges (line 12) $|J_m^t - J_m^{t-1}| < \varepsilon$ (where ε denotes the convergence limit) or until the maximum number of iterations *maxIt* is reached (lines 6 to 17). For our purpose, the distance function is the euclidian distance, defined as:

$$distance(x, k) = \left(\sum_{d=1}^{n} |x(d) - k(d)|^2 \right)^{1/2} \tag{3}$$

with n the dimension of the capability vectors x and centroid k equal to the number of service candidates.

B. Weighted Fuzzy Clustering

The basic FCM algorithm considers all data elements of equal importance. Thus when calculating the cluster centroid, the impact of a capability feature vector x is determined by its distance and membership degree. Data elements further away—thus having lower membership degree—yield lower impact on the center than closer elements. We adapt the FCM algorithm considering also the capability importance. To this end, we drop the condition that $\sum \mu_i = 1$. We weight the membership according to the importance vector $\mathcal{W}$ (line 4). After multiplying the membership table with the importance vector ($\mathcal{M}_\omega = \mathcal{M} \times \mathcal{W}$), less significant capabilities yield little impact when calculating the cluster center. The weighted centroid k_j with importance vector $\mathcal{W}$ and fuzzy factor m is defined as:

$$k_j = \frac{\sum_i \mu_{i,j,omega}^m * x_i}{\sum_i \mu_{i,j,\omega}^m} \tag{4}$$

The membership of a capability x belonging to a particular cluster k depends on the ratio of distance between x and k

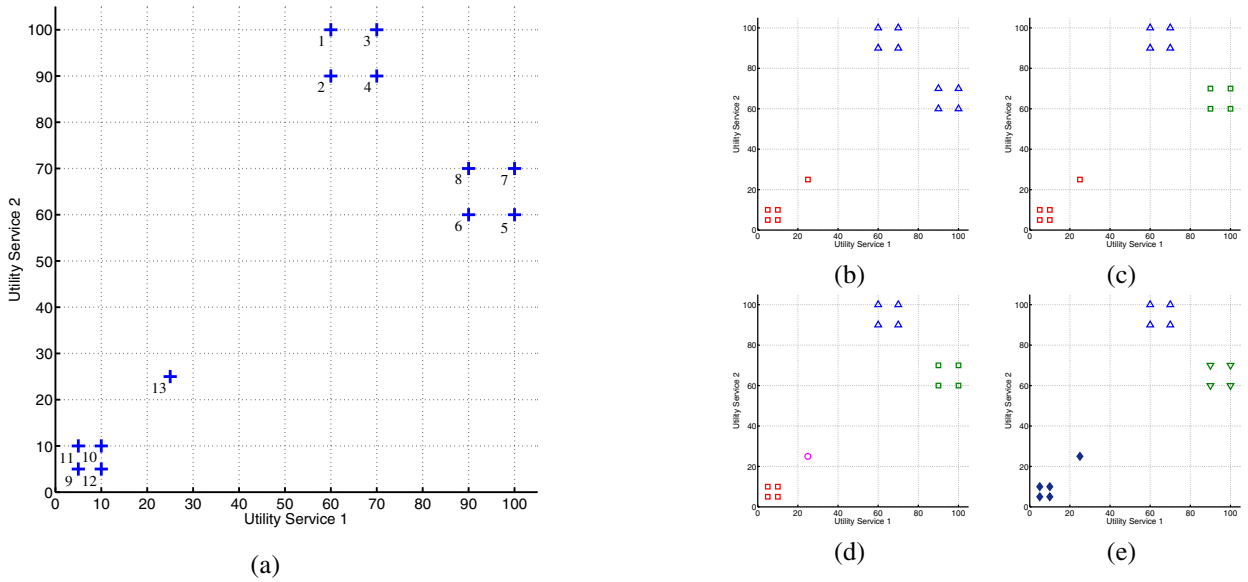


Fig. 1. Fuzzy clustering results on data set (a) for two (b), three (c), and four (d) clusters. Same colors/icons represent mutual cluster membership. Full icons depict near equal fuzzy membership in the main two clusters that form the optimal solution (e).

Algorithm 1 Weighted Fuzzy C-Means Clustering Algorithm $WFCM(\mathcal{U}_\omega, z, m, \varepsilon, maxIt, \mathcal{W}, \beta)$.

1: **function** PERFORMCLUSTERING($\mathcal{U}_\omega, z, m, \varepsilon, maxIt, \mathcal{W}, \beta$)
2: $\mathcal{M} \leftarrow$ `initRandomMembership`$(\mathcal{U}_\omega, z)$
3: /* Weight membership according to importance. */
4: $\mathcal{M}_b \leftarrow \mathcal{M} * \mathcal{W}$
5: $lastJ \leftarrow 0$
6: **for** $round = 1 \ldots maxIt$ **do**
7: $K \leftarrow$ `calculateClusterCenters`(z, m)
8: $\mathcal{M} \leftarrow$ `updateClusterMembership`$(\mathcal{U}_\omega, \mathcal{K}, m)$
9: /* Recalculating the cluster membership resets $\sum \mu_{ij} = 1$, therefore update membership again according to importance. */
10: $\mathcal{M}_\omega \leftarrow \mathcal{M} * \mathcal{W}$
11: $J =$ `calculateObjectiveFct`$(\mathcal{U}_\omega, \mathcal{K}, m)$
12: **if** $|J - lastJ| < \varepsilon$ **then**
13: *break*
14: **else**
15: $lastJ \leftarrow J$
16: **end if**
17: **end for**
18: $maxDist \leftarrow$ `calculateTotalDistance`$(\mathcal{U}_\omega, \mathcal{W})$
19: `calculateQuality`$(\mathcal{U}_\omega, \mathcal{W}, K, \beta, \mathcal{M}_\omega, maxDist, m)$
20: $\mathcal{M} \leftarrow$ `normalizeMembership`$(\mathcal{M}_\omega)$
21: **return** $\mathcal{M}$
22: **end function**

and the distance from x to all centroids $\mathcal{K}$:

$$\mu_{ij} = \left(\sum_{l=1}^{|K|} \left(\frac{||x_i, k_j||}{||x_i, k_l||} \right)^{2/(m-1)} \right)^{-1} \quad (5)$$

The reevaluation of membership degree in each iteration resets $\sum \mu_i = 1$, hence we need to recalculate the membership table in every iteration (line 10).

C. Cluster Quality

The clustering procedure itself does not give any indication on the optimal number of clusters (the 'c' in fuzzy c-means). Instead we require some quality measurements to determine which number of clusters provides the most sensible results. A rule of thumb [7] recommends selecting $max_z \approx i^{1/2}$ with i the number of data elements. A computationally more intensive approach calculates the clustering quality for increasing number of clusters until reaching maximum quality. [8] propose a combination of cluster compactness and cluster separation for crisp clustering as a viable overall quality measure. Compactness describes how well the clusters explain the variance in the data. Cluster separation describes the heterogeneity between clusters. Clusters further apart exhibit more distinct elements than clusters close together.

First we update the definition of variance to account for weighted data elements. The weighted variance v_ω of a set of capability constraints and importance vector $\mathcal{W}$ is defined as:

$$v_\omega(\mathcal{U}_\omega) = \sqrt{\sum_{i=1}^{n} \left(||x_i, \bar{x_\omega}||^2 * \omega_i^2 \right) * \left(\sum_i \omega_i^2 \right)^{-1}} \quad (6)$$

$$\bar{x_\omega} = \sum_i \frac{x_i * \omega_i^{\varpi}}{\sum_i \omega_i^{\varpi}} \quad (7)$$

where $||x_i, \bar{x_\omega}||$ computes the distance of x_i to the weighted mean $\bar{x_\omega}$ of all elements in X. The less dispersed the elements, the smaller the variance.

Next, we compare the variance found in each cluster to the overall variance. We alter the definition of compactness cmp

to consider fuzzyness:

$$cmp = \frac{1}{z}\sum_{j=1}^{z}\sqrt{\frac{\sum_i \mu_{ij} * \|x_i, k_j\|^2}{\sum_i \mu_{ij}}} * v_\omega(\mathcal{U}_\omega)^{-1} \tag{8}$$

where $\sqrt{\frac{\bullet}{\bullet}}$ calculates the variance of elements weighted by their degree of membership in cluster k_j. Compactness is 1 for one cluster. With increasing clusters, the compactness eventually decreases to 0 at which point each data element resides in a separate cluster. We prefer lower compactness (i.e., clusters describe the variance increasingly well), but we need to avoid introducing too many clusters. To this end, we reuse the cluster separation metric by [8].

Separation is the coefficient of total pairwise distance between cluster centers and maximum possible distance. Separation reaches its maximum ($sep = 1$) when each cluster contains exactly one element. When one cluster comprises all elements, separation is zero. We update the function for calculating the total distance between constraints accordingly. Distance between important constraints gains significance, while distance between less important or mixed important elements has little effect on the overall distance.

$$dist_{\mathcal{U}} = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \|x_i, x_j\|^2 * \frac{\omega_i + \omega_j}{2} \tag{9}$$

For cluster separation, we have to adapt the distance measurement between clusters. For each cluster we compute the importance of the contained elements and apply the same weighted distance function as introduced above.

$$sep_\omega = \sum_{j=1}^{z-1}\sum_{p=j+1}^{z}\left(\|k_i, k_j\|^2 * \frac{\sum_i(\omega_i * \mu_{ij}) + \sum_i(\omega_i * \mu_{ip})}{2}\right) * dist_{\mathcal{U}}^{-1} \tag{10}$$

where $\sum_i \omega_i * \mu_{ij}$ defines the importance of cluster j. The sum of pairwise distance between all elements yields computational complexity $O(|C|^2)$. However, the distance remains unchanged for all iterations of cluster counts $z = 1 \ldots |C|$ and thus needs computation only once (line 18).

The combined metrics identify the maximum clustering quality. For one cluster, compactness equals 1 and separation equals 0. For all elements in individual clusters, compactness yields 0 and separation 1. The quality function $q(\beta)$ identifies the number of clusters that best describe the underlying distribution (line 19):

$$q(\beta) = 1 - (\beta * cmp + (1 - \beta) * sep) \tag{11}$$

where β defines a preference on compactness or separation. A β value below 0.5 assigns more weight on distinct clusters (sep) than on (lower) intra-cluster variance (cmp), and vice versa. The maximum quality value identifies the best number of clusters.

D. Effect of Weights on Cluster Formation

One phenomena when applying the importance vector $\mathcal{W}$ is having the most significant capabilities rapidly split up into separate, roughly equal clusters. Clusters of less important capabilities form rather late (i.e., for high numbers of z, close to $|C|$) if m is low and do not form at all for $m > 3$.

Figure 2 compares cluster entropy H_k for unweighted (a) and weighted (b) clustering, with $m = 2$. Cluster entropy measures for each element the membership degree distribution across all available clusters. Low entropy values (dark colors) indicate focus on one or a few clusters. Bright colors highlight elements that (equally) belong to many clusters. Each column comprises the entropy values for a single capability. The top row contains the entropy values for $z = 1$ clusters, respectively the bottom row for $z = 13$ clusters.

In the weighted case, we notice how capabilities $1 \ldots 8$ break into smaller clusters before populating individual clusters ($z = 8$). The remaining capabilities belong equally to an increasing number of cluster until after row eight capability 13 separates into a distinct cluster. Capabilities $9 \ldots 12$ never form a cluster themselves. The regular FCM algorithm has capabilities $9 \ldots 12$ rapidly form a cluster, while capabilities $1 \ldots 8$ display comparatively less crisp cluster membership.

E. Effect of Weights on Compactness and Separation

Figure 3 displays compactness and separation for $m = [1.5; 2; 3]$ with unweighted and weighted clustering side by side.

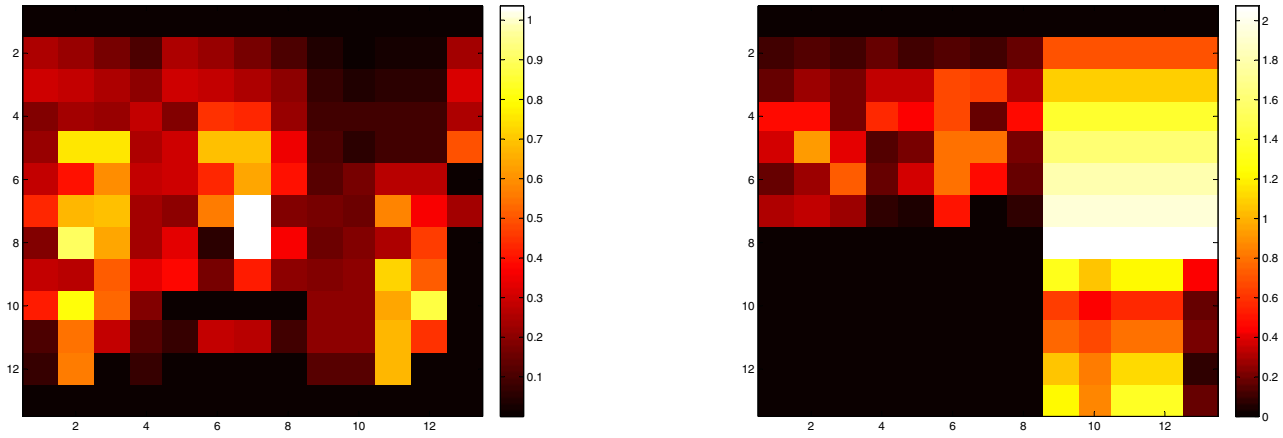
We notice an early, sharp decline in compactness opposed to a late, steep increase in separation. Compactness is minimal when elements populate individual clusters. As observed above, the most significant elements quickly scatter into separate groups. If insignificant elements eventually occupy their own cluster, they barely reduce compactness.

The same effect determines the late, steep incline of cluster separation. Separation is maximal when each cluster contains a single element. As long as clusters of significant elements split into increasingly smaller clusters, the centroids remain close together, adding little to separation. The distance between centroids grows once less significant elements form individual clusters clearly separated from the existing cluster centers.

Comparing unweighted and weighted clustering, we notice weighted compactness reaching its minimum once all important elements reside in separate clusters. Unweighted compactness drops similarly fast at the beginning, but then phases out, reaching its minimum at $z = |C|$. Weighted separation remains low until all important elements populate individual clusters, then rising sharply. In contrast, the unweighted separation metric displays near-linear growth.

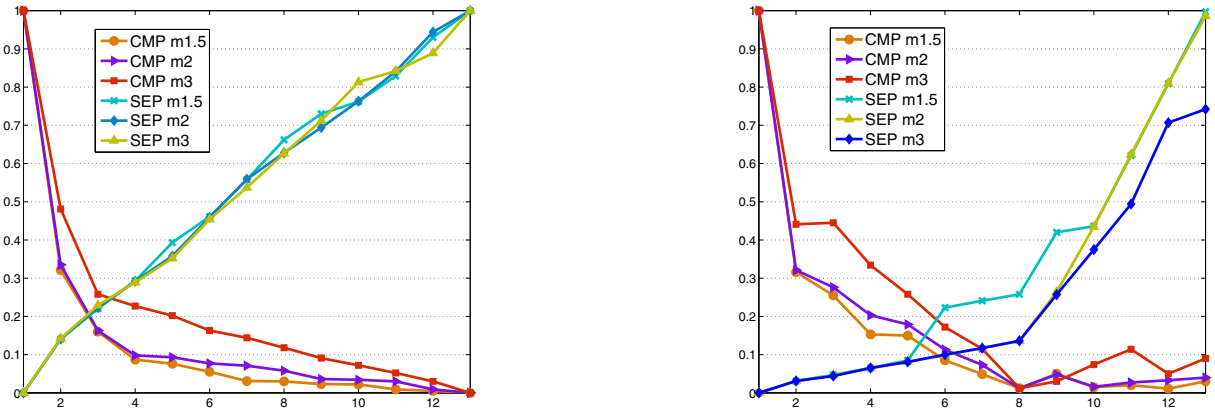
V. Evaluation

To the best of our knowledge, there exists no data set describing the utility, respectively the score, of Web service operations. Most collections consist of services with a single capability or provide only Quality of Service data; both types are unsuitable for evaluating our clustering approach. We therefore have to adapt another real world data set that can be applied to the domain of service aggregation.



(a) (b)

Fig. 2. Cluster entropy H_k for unweighted (a) and weighted (b) clustering. Capabilities 1 to 13 along the x-axis, cluster size z along the y-axis.



(a) (b)

Fig. 3. Compactness and separation for unweighted (a) and weighted (b) clustering.

Slashdot[2] is a user-driven news portal focusing on various aspects of information technology. News fall into multiple categories (i.e., subdomains). Slashdot exhibits similar characteristics as large-scale complex service systems. Some entities remain consistently active throughout all subdomains. Other entities join in an ad-hoc manner, participate for a limited period, and then vanish again. In Slashdot, users are interested in providing their knowledge to improve the quality and information content of a story (i.e., they fulfil a task). They rarely engage in direct communication with other users [9], [10]. Hence, we treat Slashdot users as Human provided Services (HPS [11]).

The general question we can answer applying our weighted clustering approach to the Slashdot data set is: Which clusters do arise, and how much do we benefit from selecting the best users (i.e., services) from clusters compared to a regular ranking process that matches either all capabilities to a single user, or each capability to a distinct user.

[2]http://www.slashdot.org

A. Experiment Setup

Slashdot postings are subject to a moderation system. Postings receive scores between −1 (low quality) and +5 (high quality). Predicates enable the classification of postings according to *insightful*, *interesting*, *informative*, *funny*, etc. content. Total scores (i.e., *Score*) and total posting counts (i.e., *Count*) serve as required service capabilities. Specifically, we derive for a user the total posting count and the total score summarized across his/her postings for each subdomain and predicate combination. For each capability constraint, users with the highest total score (respectively total posting count) receive a utility value of 100, with the worst users having utility 0.

We select a set of subdomains SD and identify all users who have submitted at least 5 postings in these subdomains that scored 2 or higher. Given a set of predicates P we arrive at the capability constraints set C of size $|P| * |SD| * 2$. For each user, we derive the utility values for all constraints. We select three subdomains—*Ask*, *Entertainment*, and *Mobile*—and focus in particular on the scores of *funny*, *interesting*, and

insightful postings arriving at 18 constraints (i.e., $Ask-Fun-Count$, $Ask-Fun-Score$, $Ask-Ins-Count$, etc.). Across the three subdomains, we select users having 5 or more postings rated 2+ from February 1st to July 1st 2008. We treat users below this initial threshold as services exhibiting capabilities we are not interested in. As a side effect, reducing the initial set of candidates (here 255 users) reduces the duration of the clustering process.

B. Unweighted Clustering Results

First, we analyze unweighted clustering, where we ignore constraint importance and work with the unweighted utility matrix $\mathcal{U}$. The cluster quality metric identifies 12 clusters to optimally describe the constraints. Specifically, the resulting cluster membership places *Count* and *Score* constraints of every subdomain and predicate in the same cluster except for the statistics describing Ask-Insightful, Entertainment-Funny and Entertainment-Interesting which populate individual clusters. Cluster membership μ is larger than 0.9 for all constraints.

We calculate the pairwise Jaccard similarity between any two clusters for the top 50 users. Figure 4a visualizes the resulting similarity matrix. Row 13 and column 13 contains the unclustered ranking set. We notice that some clusters yield extremely high similarity. Specifically cluster 3 and 5, 4 and 10, as well as cluster 6 and 11 have many users in common. Although all clusters yield significant differences to the unclustered ranking, the clustering process has created three pairs of clusters that should be merged. Incidentally, these pairs comprise of the above mentioned constraints, where Score and Count of the same subdomain and predicate end up in different clusters (Ask-Insightful, Entertainment-Funny and Entertainment-Interesting). Merging these cluster pairs would not significantly reduce the overall clustering quality.

C. Weighted, Equal Importance Clustering Results

Second, we cluster with equal constraint importance but weighted utility matrix $\mathcal{U}_\omega$. Constraints that exhibit low service support yield less impact during clustering than well supported constraints. Table I provides the constraint membership in the resulting six clusters. Most constraints yield crisp cluster membership ($\mu > 0.9$) with exception to Entertainment Funny, Entertainment Interesting, and Mobile Funny which do not strongly belong to any cluster. The weight vector $\mathcal{W}$ is a good indicator on which constraints are likely to yield crisp clusters. A low weight value by itself, however, is not sufficient. The constraints Ask-Funny-Score ($\omega = 0.804$) exhibits lower weight than Entertainment-Funny-Count ($\omega = 0.836$) but ends up clearly assigned to cluster 1. Here, the close correlation of count and score values is decisive.

We ensure that clusters provide more fitting elements than the unclustered ranking result by pairwise comparing the top-k elements with Pearson's correlation coefficient (ρ) and Jaccard similarity (J). Table II lists the ranking differences of the top 50 users. The Jaccard similarity measures the set overlap of users regardless of their rank. Pearson's coefficient requires both sets to contain the same elements. We therefore take the union of elements from both rankings (given in brackets in Table II) and then compute ρ.

Constr.	ω	Cl_1	Cl_2	Cl_3	Cl_4	Cl_5	Cl_6
A-Fun-C	0.864	**.959**	.005	.011	.012	.008	.006
A-Fun-S	0.805	**.968**	.004	.009	.009	.006	.005
A-Ins-C	1.394	.019	.010	.023	.025	**.914**	.010
A-Ins-S	1.735	.005	.003	.008	.007	**.975**	.003
A-Int-C	1.185	.009	.004	.006	**.969**	.008	.005
A-Int-S	1.226	.011	.004	.007	**.964**	.009	.005
E-Fun-C	0.836	**.240**	.120	.176	.158	.104	.202
E-Fun-S	0.702	**.253**	.116	.170	.146	.099	.215
E-Ins-C	1.035	.010	.004	**.965**	.007	.010	.004
E-Ins-S	1.008	.007	.003	**.976**	.004	.007	.003
E-Int-C	0.513	**.338**	.090	.179	.156	.105	.132
E-Int-S	0.748	**.291**	.085	.215	.158	.143	.108
M-Fun-C	0.563	.197	.136	.123	.112	.087	**.346**
M-Fun-S	0.563	.205	.130	.127	.110	.087	**.341**
M-Ins-C	1.459	.003	**.980**	.003	.003	.003	.007
M-Ins-S	1.503	.003	**.983**	.003	.003	.003	.007
M-Int-C	0.814	.006	.011	.005	.006	.004	**.969**
M-Int-S	1.046	.008	.016	.006	.008	.005	**.957**

TABLE I
CLUSTER MEMBERSHIP AND WEIGHT VECTOR $\mathcal{W}$ FOR CONSTRAINTS FROM *COUNT*, *SCORE*, WITH SUBDOMAINS *ASK*, *ENTERTAINMENT*, AND *MOBILE* AND PREDICATES *FUNNY*, *INSIGHTFUL*, AND *INTERESTING*.

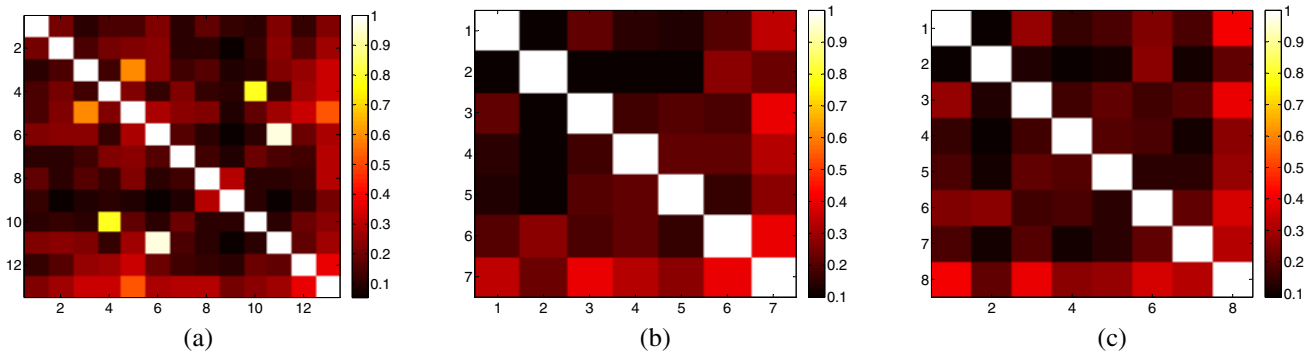
Average Jaccard similarity remains low. On aversage, only 34% of the top-50 nonclustered users are also listed in individual clusters. The average Pearson's coefficient stresses the ranking differences even more. We observe no correlation in ranks for the top 50 users.

D. Weighted, Variable Importance Clustering Results

Next, we introduce constraint importance to analyze their effect on the clustering result. In this third experiment, we increase the importance of following four constraints: Entertainment-Interesting-[Count|Score] ($\tau = 0.08$) and Mobile-Funny-[Count|Score] ($\tau = 0.07$). The remaining constraints exhibit identical weights ($\tau = 0.05$) so the total weight remains 1.

Clustering this configuration produces one more cluster. Also cluster membership has changed for some constraints. Both, Entertainment-Interesting and Mobile-Funny populate now their own cluster exhibiting high crispness. On the other hand, Ask-Funny looses its clear membership in a single cluster, now yielding fuzzy membership across all clusters. Entertainment-Funny maintains its fuzzyness but shares its largest membership with Mobile-Funny instead of Entertainment-Interesting.

Again, we pairwise compare the non-clustered ranking and each cluster for ranking differences. Compared to the previous experiment, Jaccard similarity is similarly low (average $\sim$ 32%) and Pearson's coefficient either shows no correlation with the non-clustered set of top 50 users. Cluster 7 emerges not only due to the changed importance. We evaluate the pairwise cluster similarity to ensure that the underlying data justifies this additional cluster. Figure 4 provides the similarity matrix including the non-clustered set in the last row and



(a) (b) (c)

Fig. 4. Cluster Jaccard similarity for Top 50 users for unweighted (a); weighted, equal importance (b); and weighted, variable importance (c) capability constraints.

column. Cluster 7 remains distinctively different from the other clusters for the top 50 users. Hence, the additional cluster allows for further specialization compared to the previous experiment.

Cluster	Top 50 (ω) ρ	J	Top 50 ($\omega + \tau$) ρ	J
Cl1	0.016 (74)	0.351	0.218 (71)	0.408
Cl2	-0.152 (81)	0.235	-0.199 (83)	0.205
Cl3	0.193 (71)	0.408	0.097 (72)	0.389
Cl4	0.014 (75)	0.333	-0.043 (79)	0.266
Cl5	0.031 (78)	0.282	-0.062 (78)	0.282
Cl6	-0.061 (71)	0.408	-0.164 (73)	0.37
Cl7			-0.148 (76)	0.316
Avg	0.007 (75)	0.336	-0.043 (76)	0.319

TABLE II
RANKING DIFFERENCES OF TOP 50 USERS BETWEEN EACH CLUSTER AND THE UNCLUSTERED RANKING ORDER MEASURED WITH PEARSON'S CORRELATION COEFFICIENT (ρ) AND JACCARD SIMILARITY (J). WEIGHTED, VARIABLE IMPORTANCE ($\omega + \tau$) AND WEIGHTED, EQUAL IMPORTANCE (ω) CAPABILITY CONSTRAINTS.

E. Discussion of Clustering Results

The three experiments have shown how constraint weights influence the clustering result for the same underlying utility data. Weighted, equal importance clustering highlights the constraints that are fulfilled by most users (i.e., services). Subsequently weighted, variable importance clustering successfully shifts the focus onto the capability constraints considered more important. New, or changed, clusters emerge only when the preferred constraints indeed meet a distinct difference in the underlying capability data set compared to the remaining constraints.

Regular clustering — as tested in the unweighted experiment — produced too many clusters. However, both weighted experiments exhibited very distinct clusters. Inter-cluster Jaccard similarity remained in the range of [0 ↔ 0.25], [0.02 ↔ 0.28], and [0.13 ↔ 0.42] for the top-10, top-50, and top-100 users (out of 255), respectively.[3] The corresponding similarity between the unranked result and each cluster resulted in slightly higher values.

[3]Due to space constraints, Table 4 provides only the detailed similarity results for top 50 users.

Clustering also promotes users to the top elements in a cluster which are badly ranked in the non-clustered set. The Pearson's correlation coefficient emphasized the element positioning difference between non-clustered and clustered ranking order for both weighted experiments. We observed a negative correlation for the top-10 users, no correlation for the top-50 users, and hardly any correlation for the top-100 users.

The service utility for a given capability and cluster is determined by the original utility value, capability importance, and capability cluster membership:

$$u_{ijk} = u_{ij} * \tau_i * \mu_{ik} \tag{12}$$

Table III compares the utility values of the single best unclustered service to an aggregation of services from the various clusters. The average utility benefit for selecting the top-10 users set from every cluster amounts to a 38% (equal importance) to 51% (variable importance) utility increase compared to the unclustered ranking result.

	Non	Cl_1	Cl_2	Cl_3	Cl_4	Cl_5	Cl_6	Cl_7	Impr
	w_C	.196	.150	.167	.159	.143	.185		
1	60.7	76.2	73.3	76.3	78.2	84.1	61.9		23%
2	39.3	48.8	71.7	71.9	70.1	79.0	49.2		62%
3	35.7	47.5	67.4	60.3	47.1	61.0	42.2		50%
10	30.5	32.8	50.3	39.3	38.8	50.6	31.1		30%
	w_C	.177	.119	.135	.115	.119	.126	.209	
1	58.2	84.4	82.6	84.5	85.0	90.9	77.5	85.8	45%
2	43.6	83.7	78.1	76.7	71.2	81.5	47.9	49.8	58%
3	34.3	55.7	75.7	64.4	48.0	65.7	44.7	48.7	66%
10	29.5	39.0	55.7	41.5	37.8	53.0	36.6	33.3	40%

TABLE III
UTILITY COMPARISON OF TOP 10 USERS OF EACH CLUSTER AND THE UNCLUSTERED RANKING ORDER. CLUSTER WEIGHT w_C PROVIDES THE IMPORTANCE OF ALL CAPABILITIES IN THAT CLUSTER ACCORDING TO CAPABILITY MEMBERSHIP.

Compared to existing techniques that map each required capability to a dedicated service, our approach greatly reduces the number of required services. In the weighted examples only 6, respectively 7, services are required compared to the 18 capability constraints.

VI. Related Work

Numerous papers improve the Fuzzy C-Means algorithm to achieve robustness (e.g., [12], [13], [14]). These techniques apply data distribution intrinsic metrics to identify and mitigate the effect of outliers and noise. We focus on achieving optimum clusters where significant capability constraints and capability support by actual service instances should influence the result more than insignificant constraints or services.

In the SOA domain, clustering techniques have been applied to compute the similarity of Web services based on key words extracted from the service interface descriptions (WSDL) [15], [16], [17]. These approaches aim to find similar services given a particular query. In contrast, we extract a set of complementary service clusters, from which an aggregation of individual services provides the required capabilities. Also these approaches observe only keywords, but not consider a service's fitness to provide a certain capability.

More advanced adaptation techniques exist in the broader domain of service composition. Work focusing on QoS-centric service selection, recommendation, and composition is orthogonal to our approach. QoS metrics describe parameters such as throughput and latency [18], or trust and reputation [4], [19] that apply equally to any service. In contrast, our aggregation mechanism builds on capabilities which reside between QoS and service interface descriptions. Other complementary techniques to service composition are goal-driven approaches such as [20], [2], [21]. These approaches, however, focus on optimal replacement of individual services to maintain desirable, global composition quality metrics.

VII. Conclusion

We have presented a weighted fuzzy clustering approach to exploit implicit service groups. Our technique produces service aggregations that significantly reduce the number of actual involved services (compared to the number of capability requirements) while taking advantage of service specialization. As a side effect, discovered service clusters enables the rapid replacement of individual services without having to analyze the overall composition. Analysis of capability support amongst the services, and capability requirement weights allow for fine-grained cluster formation control. We demonstrated the effectiveness of our approach based on a real world data set.

The current approach, however, does not consider any dependencies between service capabilities (e.g., capabilities that need to be available on the same service instance) and required QoS levels. We aim to include such restrictions in the clustering algorithm in the future. Additionally, we plan to investigate how the clustering result can respect SLAs to address compliance concerns.

Acknowledgments

This work has been partially supported by the EU STREP projects Commius (FP7-213876) and COMPAS (FP7-215175).

References

[1] 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," *Applied Ontology*, vol. 1, no. 1, pp. 77–106, Jan. 2005. [Online]. Available: http://iospress.metapress.com/content/dccb867p347xxebx

[2] D. Greenwood and G. Rimassa, "Autonomic goal-oriented business process management," in *ICAS '07: Proceedings of the Third International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 43.

[3] M. Mrissa, C. Ghedira, D. Benslimane, Z. Maamar, F. Rosenberg, and S. Dustdar, "A context-based mediation approach to compose semantic web services," *ACM Trans. Internet Technol.*, vol. 8, no. 1, p. 4, 2007.

[4] L.-H. Vu, M. Hauswirth, and K. Aberer, "Qos-based service selection and ranking with trust and reputation management," in *OTM Conferences (1)*, 2005, pp. 466–483.

[5] C. Dorn, D. Schall, and S. Dustdar, "Context-aware adaptive service mashups," December 2009, iEEE Asia-Pacific Services Computing Conference (APSCC) - (short paper).

[6] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[7] A. McBratney and J. De Gruijter, "A continuum approach to soil classification by modified fuzzy k-means with extragrades," *Journal of Soil Science*, vol. 43, pp. 159–175, 1992.

[8] J. He, A.-H. Tan, C.-L. Tan, and S.-Y. Sung, *On Quantitative Evaluation of Clustering Systems*. Kluwer Academic Publishers, 2003. [Online]. Available: citeseer.ist.psu.edu/he02quantitative.html

[9] V. Gómez, A. Kaltenbrunner, and V. López, "Statistical analysis of the social network and discussion threads in slashdot," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. New York, NY, USA: ACM, 2008, pp. 645–654.

[10] F. Skopik, H.-L. Truong, and S. Dustdar, "Trust and reputation mining in professional virtual communities," in *9th International Conference on Web Engineering (ICWE)*. Springer, June 2009.

[11] D. Schall, H.-L. Truong, and S. Dustdar, "Unifying human and software services in web-scale collaborations," *IEEE Internet Computing*, vol. 12, no. 3, pp. 62–68, May/June 2008.

[12] K. Chintalapudi and M. Kam, "A noise-resistant fuzzy c means algorithm for clustering," *Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, vol. 2, pp. 1458–1463, May 1998.

[13] J.-S. Zhang and Y.-W. Leung, "Improved possibilistic c-means clustering algorithms," *Fuzzy Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 209–217, April 2004.

[14] J. Leski, "Towards a robust fuzzy clustering," *Fuzzy Sets Syst.*, vol. 137, no. 2, pp. 215–233, 2003.

[15] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*. VLDB Endowment, 2004, pp. 372–383.

[16] R. Nayak and B. Lee, "Web service discovery with additional semantics and clustering," in *WI '07: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 555–558.

[17] C. Platzer, F. Rosenberg, and S. Dustdar, "Web service clustering using multidimensional angles as proximity measures," *ACM Trans. Internet Technol.*, vol. 9, no. 3, pp. 1–26, 2009.

[18] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic, and S. Dustdar, "Towards composition as a service - a quality of service driven approach," 29 2009-April 2 2009, pp. 1733–1740.

[19] E. Maximilien and M. Singh, "Self-adjusting trust and selection for web services," June 2005, pp. 385–386.

[20] T. Yu and K.-J. Lin, "Adaptive algorithms for finding replacement services in autonomic distributed business processes," in *Autonomous Decentralized Systems, 2005. ISADS 2005. Proceedings*, April 2005, pp. 427–434.

[21] S. H. Ryu, F. Casati, H. Skogsrud, B. Benatallah, and R. Saint-Paul, "Supporting the dynamic evolution of web service protocols in service-oriented architectures," *ACM Trans. Web*, vol. 2, no. 2, pp. 1–46, 2008.