Online Service Similarities and Reputation-based Selection

Oana Dini[*], Pascal Lorenz[**], Abdelhafid Abouaissa[**], Hervé Guyennet[*]
[*)] Université de Besançon, France
[**)] Univeristé de Haute Alsace, France
oana.dini@univ-fcomte.fr, lorenz@ieee.org, a.abouaissa@uha.fr, guyennet@lifc.univ-fcomte.fr

Abstract – **Selection of the most appropriate service by correct invocation is a challenge. This is due to the difficulties of correctly exposing proper ways to invoke a service, to the variety of services, from on-line services, software pieces, to shopping, and to different invoker behavior. When considering invoker feedback, service ranking based on the user's perception, or based on the recommenders' statistics are relevant. The paper presents adapted approaches to select services based on distance and similarity, and introduces a similarity taxonomy to better tune various kinds of service invocation under specific constraints, such as relaxation, type of similarity, context, and service ranking. Selection is also based on the feedback from the user. The proposed model is used for building a selection algorithm that allows variations on service invocation.**

Keywords *–* ***service similarity; similarity class; temporal similarity; selection patterns****.*

I. INTRODUCTION

The large spectrum of user behaviors (and, in general, the variety of needed services) leads to the need of similarity-based matching, when a given service is required. Traditionally, the notions QoS (Quality of Service), and QoE (Quality of Experience) deals with these aspects. However, the perfect matching and the approximate matching depend on a large number of factors. For example, if we consider Web Services dedicated to weather forecast, location, month/day/year, parameters (rain, wind, temperature, and pressure) can be appropriate parameters when inquiring. Definitively, there are several forecast services, and the experience of a particular user might differ from one forecast service to another. Some provide information that is more accurate than others (i.e., data is more frequently updated), history is better preserved by particular services, via backward search, e.g., Weather Underground, etc. A similar problem is observed when choosing and downloading a particular piece of software, when inquiring for a specialized on-line book shop, or when looking for a service providing the most updated world-wide information. Finally, some services offer a friendlier interface for searching, ordering, and getting delivered a particular need (i.e., personalized interface, myAccount, etc.).

There are meta-services, providing the service at choice. Such examples are those for buying flight tickets, where the cheapest, the quicker, or other selection criteria are used for service selection. Other meta-services are for selecting the most appropriate software to download, or for booking a hotel. In most of the cases mentioned above, one criterion is usually considered to select from an existing service pool.

Two phases involve service features, (i) service discovery (locating) and (ii) service selection (in the case of a set of services, relatively satisfying the needs with similar degrees of satisfaction). Both phases require special mechanisms to assess service similarity. Meta-services have a restrained number of known services, that are well localized and whose parameters are also limited. Because of this, the selection appears to be less complex. With a well known service and limited criteria (usually one search/ selection criterion), similarity is relatively easy to be determined.

The above considerations are no longer valid for a large spectrum of properties a service might expose in order to satisfy a given service request. To satisfy a request, service similarity plays an important role for timely identification and delivery, and for an optimal (maximal) customer (invoker) satisfaction. Customer satisfaction is expressed by QoE, on-line feedback, service ranking, and manifested by variations of QoS to keep service costs and satisfaction in synchrony.

The paper deals with service similarity and proposes a adaptive similarity taxonomy and mechanisms to handle service discovery and service selection considering service specification, end-user (requester) perception, and service reputation. In Section II, existing approaches for service similarities are presented. Section III introduces a context-based similarity model, including distance and similarity metrics, a similarity taxonomy, and other facilities to consider service ranking and feature relaxations. Section IV presents an algorithm to compute a minimum set of existing services satisfying a given query, following the newly introduced model. Section V concludes on the approach and presents further developments.

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-105-2

II. RELATED WORK

Finding similar services (approximate but satisfactory matching) is somehow similar to (i) text matching, (ii) schema matching, or (iii) software-component marching. For some text matching solutions (information retrieval) mechanisms based on term frequency are used [7][8]. In schema matching, special techniques are using semantics of the schemas to suggest schema matching [9]. Mainly, linguistic and structural analyses, as well as domain knowledge, are methods to handle schema matching. When expanding to software component matching [10] (considerably used in software reuse) component signature and program behavior (usually formally defined) are considered; in this case, data types and post-conditions should be considered for matching. However, these techniques are not suitable for Web Services [6], as data types and post-conditions are not available. Usually, such a service has a name and text description in UDDI **(**Universal Description, Discovery, and Integration) registry, operation decryptions, and input/output descriptions; the last two are usually specified in WSDL (Web Service Description Language).

Dong *et al.* [6] proposed criteria for associating similar terms. They introduced the cohesion/correlation score, as a measure of how tight two terms are. However, they do not consider particular characteristics of a term. They applied the score only to Web Services. We start from the idea that services similarity has a meaning only between services than can be context-oriented and belong to a cluster (e.g., invoking a service gives a list of similar operations with similar results). Other approaches consider both diversity and similarity at the same time, having the distance as a metric [11]. We adopt these metric (see Section III) and adapt them to the service similarity computation.

In fact, specific to each service, there are particular service parameters that are agreed upon between a provider and a subscriber, commonly settled by the SLA (Service Level Agreement). On the provider side, the SLA parameters are used for technical audit and litigations (leading to penalties or bonuses towards a given user or class of users). Specific on-line and off-line measuring mechanisms for SLA metrics and specialized audit techniques have been proposed. On the consumer side, the subscribers' satisfaction is gathered and mapped to the audit results to validate a given service, to detect flaws in delivering a service, and to ultimately build a view on service reputation. In general, a record is handled per service or per products, with respect to a given subscriber or a class of subscribers. Feedback can be used to enforce service similarity.

In this paper, we expand the cluster-based similarity to service similarity and introduce similarity taxonomy, where the service consumer has a weight in deciding service similarity. The idea is to establish service ranking (and reputation) inside a given cluster, and define similarity considering service-provider and service-user feedback.

III. A CONTEXT-BASED SIMILARITY MODEL

Then main idea of our approach is (i) having well defined service clusters, (ii) compute the distance between service feature, (iii) evaluate service similarity, based on service features, (iv) consider user-, service,- and producer-based similarity reflected by the appropriate reputations, and (v) evaluate how interchangeable two services are. When a service query is issued, the algorithm we propose selects the most appropriate service, considering both distance and similarity between services.

A. Identifying clusters of similar services

Expanding what was mentioned in [6], service cohesion of a service cluster must be strong (best potential to be similar), while correlation between two service clusters should be weak (service independence). We say that service s_1 is similar with s_2, and note $s_1 \sim s_2$, if the similarity confidence is greater than a given threshold δ. In a cluster S with ||S||, where ||x|| is the cardinality of x, we redefine cohesion and correlation as follows:

$$Cohe_S = \{(s_i, s_j) \mid s_i \sim s_j \ (\sim_{thres} > \delta)\} / (||S|| \times (||S|| - 1)) \quad (1)$$

and

$$Correl_{S,S'} = (A(S, S') + A(S', S)) / 2 \times ||S|| \times ||S'||, \quad (2)$$

where

$$A(S, S') = \| \{s_i, s_j \mid s_i \in S, s_j \in S' \text{ si } \sim sj \mid \sim_{thres} > \delta\}\| \quad (3)$$

with $\sim_{score} = Cohe_S / Correl_{SS'}$ (4)

defining the similarity score.

We notice that $\sim_{score}$ defines similarity classes based on the preexisting service clusters. To enhance the similarity score, clusters aggregation and clusters split operations are possible. Conditions and assessments for doing these are presented in [6].

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-105-2

B. Distance metrics for service similarity

Let us assume that a service *s* has *n* features (usually called data-points, as they are expressed by concrete values in an *n* dimensional space). The following distance methods are adapted for comparing services:

(a) Service Euclidian distance between two services in the *n* dimensional space

$d(s_1,s_2) = 1/n \Sigma (a_{1i} - b_{2i})^{**2}$, for all i = 1…n
(5)
where a_i, b_i are service features.

(b) Service city-block distance
$d(s_1,s_2) = 1/n \Sigma |a_{1i} - b_{2i}|$, for all i = 1…n
(6)

(c) Service Pearson correlation coefficient

$r(s_1,s_2) = 1/n \Sigma ((a_{1i} - \underline{a})/\sigma_a) \times ((b_{2i} - \underline{b})/\sigma_b)$, (7)
where $\underline{a}$ and $\underline{b}$ are the sample mean of a_i and b_i respectively, and σ_a and σ_b are the sample standard deviation of a_i and b_i.

The service Pearson distance is defined as
$d(s_1,s_2) = 1 - r(s_1,s_2)$
(8)

(d) Service Cosine similarity
$d(s_1,s_2) = \cos(\theta) = (s_1 \bullet s_2) / (|s_1| |s_2|)$
(9)
where ● is the vector product of s_1 ans s_2.

By selecting a service distance metric, a clustering algorithm computes the distance matrix between two services. Mostly, (a) and (b) of the above are satisfying the triangle inequality, as true metrics.

C. Classes of similarities

In order to select the most appropriate service, we introduce producer-based similarity ($\sim_{prod}$), recommender-based similarity ($\sim_{recc}$), and user-based similarity ($\sim_{user}$). Producer similarity is based on the expectation, recommender's similarity is statistics-based, and user similarity is based on user feedback. In this taxonomy, $s_1 \sim_{prod} \{s_2, s_3, \ldots\}$ define a cluster of similar services, as defined by the producer.

To refine service similarity, we introduce the notions of *primary service features* and *secondary service features*, as shown in Figure 1,

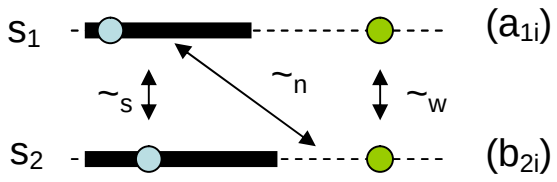

Figure 1. Similarity classes.

where the bold items represent primary service features (A_1 set), and the dashed items represent secondary service features (A_2 set) (similar for s_2)

We introduce strong, weak, and normal similarity, represented by $\sim_s$, $\sim_w$, and $\sim_n$, respectively.

Therefore, $(s_1 \sim s_2) =$

$= \sim_s$, iff all $a_{1i} \in A_1$ and $b_{2i} \in B_1$

$= \sim_w$, iff all $a_{1i} \in A_2$ and $b_{2i} \in B_2$

$= \sim_n$, iff there are $a_{1i} \in A_1$ and $b_{2i} \in B_2$ or
there are $a_{1i} \in A_2$ and $b_{2i} \in B_1$ (10)

Similarity composition allows to capture all possible combinations, e.g,, $\sim_{prod/s}$ represents a strong similarity defined by the producer, based on the primary service features.

A refinement of feature-based similarity can be expressed when service features do not show a direct semantic matching, but feature composition might lead to such a match. Considering a subset of a service feature for a given service equivalent with a feature for a service for which the similarity is computed, we introduce feature composition-based similarity, as shown in Figure 2.

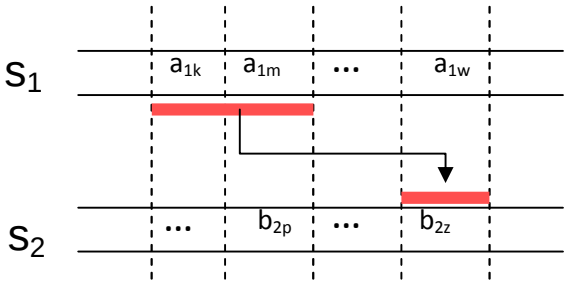

Figure 2. Feature composition-based similarity.

$s_1 \sim_{a1k,a1m\,/\,b2z} s_2$ (11)

with the semantic that the values of a_{1k} and a_{1m} composed are similar to the values of b_{2z}. Composition might be any arithmetic or Boolean operator, according to the nature of

the features, e.g., if sets, then 'U' (union), if values, then '+' (addition), etc. If type, and a_{1k}:T1 and a_{1m}:T2, and b_{2z}:T3, then, then T3 is a subtype of either T1 or T2.

Combination between $\sim_s$, $\sim_w$, and $\sim_n$, and feature composition-based similarity can be applied following (10).

D. Updating similarity

When evaluating service similarities, perfect match of service features is desired, but rarely found, due to some continuous values of the features. For example, looking for a service offering the weather temperature with an accuracy of 0.1^oF is not feasible. A query on what month the temperature is 67.3F might have no match; but, for a given location, a query for what month shows [75-80] ^oF might be answered by April or May, if a Mediterranean area. We identify two possible relaxations when performing the matching.

D.1 Context-based feature migration

In time, and based on business models or customer feedback, some primary features become secondary, and vice-versa. Even more, at the same time, in different contexts, a feature can belong to either primary or secondary feature sets.

Let $C = \{c_i\}$ a set of contexts and

$$s1 ::= (A_1 U A_2)_{context = c1}, \text{ with } A_1 \cap A_2 = \phi$$

$$s1 ::= (A'_1 U A'_2)_{context = c2}, \text{ with } A'_1 \cap A'_2 = \phi \quad (12)$$

then, the following is possible:

$$s_1 \sim_{context = c1} s_2$$

$$s_1 \sim_{context = c2} s_3 \quad (13)$$

D.2 Feature relaxation-based similarity

Service features are not always perfectly matching (so goes for query matching, as well). Most of the time, the exact matching is not mandatory, e.g., if a service feature has a numeric value, a variation of α_{1i} (usually symmetric, but not necessarily) of +/- α_{1i} is allowed. As a result, the similarity metrics presented in II.B can be relaxed. The same relaxation can be applied for similarity on data type/subtype, for similarity concerning the set of interface operations, or similarity concerning variations of an algorithm implementation. For example, when a query (with explicit relaxation of +/- 2ms) targets a service with a response delay of 10ms, any service offering a delay within [8ms, 12ms] is a desired matching. With no explicit relaxation delay, 10ms is mandatory. In this case,

$$s1 \sim_{a1i +/- \alpha 1i} s2 <=> b_{2i} \in [a_{1i} - \alpha_{1i}, a_{1i} + \alpha_{1i}] \quad (14)$$

where a_{1i} and b_{2i} are the corresponding features of s_1 and s_2, respectively.

E. Recommender-based similarity

Recommender mechanisms rank [1] the products or services based on feedback received after a series of recommendations and successful transactions. The *ranking* is subject to incomplete, fictitious feedback, volume of transactions for a given product or provider, and confidence in feedback. Based on statistics, the recommender computes its own *ranking* per product, defining the reputation (r) of a service/product.

Considering a set of service clusters a recommender builds based on type of services/products,we define:

Cluster = $\{cluster_i\}$
with $s_1 \in cluster_i$ and $s_2 \in cluster_{i,}$ for a given service feature

$$s_1 \sim_{feature = ai} s_2 ::= |rank_{s1} - rank_{s2}| < \varepsilon_{ai} \quad (15)$$

In general,

$$s_1 \sim_{Uai} s_2 ::= \max \{|rank_{s1} - rank_{s2}|\} < \min \{\varepsilon_{ai}\} \quad (16)$$

F. Customer feedback reputation-based similarity

Based on customer individual metrics, context, and potential query with relaxation, a reputation is associated with a service/product. Heuristics for updating the reputation have been presented in [1][2]. In general, the following information is available:

s <r>: each service <s> has an associated reputation <r>
P_i<s,r_i>: each provider offers a service with its associated reputation
P_j<s,r_j>: another provider can offer the same service with a different associated reputation
u <e, c>: a user *u* has a credibility and confidence metrics associated with it
For simplicity, we consider that <e, c> are the same for any service.

For a given user, we define similarity in terms of r_s

$$s_1 \sim_{feedback} s_2 ::= |r_{s1} - r_{s2}| < \varepsilon_0 \text{ with } e > e_0 \text{ and } c > c_0 \quad (17)$$

In the following, the newly introduced model is used by an algorithm to identify the most suitable service to satisfy a query for a service.

III. ALGORITHM FOR SERVICE RETRIEVAL USING SIMILARITY

We introduced a similarity model and classes of similarity that allow a user (invoker) to use a service in a given context, allowing or not precise relaxation for some service features, and under different types of similarity (strong, weak, normal). Distance metrics were also adopted for services, in order to cluster the most suitable services for a particular query, before computing the similarity.

Based on the model introduced in Section III and on the user model [2] and reputation [1], a query for a service *s* can be expressed as

Q (s, similarity type, context, with/without relaxation on $\{a_{1i}\}$)

The algorithm presented below illustrates the main steps to reach a service proposal that can be a set, a given service, or no service at all.

Algorithm for finding a requested service query **Q**, *based on similarity between potential satisfying services*

1: **begin**
2: **identify** the service cluster [see (4)]
3: **select** a distance metric [see (5)-(9)]
4: **calculate** distance between all s_i in the cluster
5: **select** a subset $\{s_k$ with min $\{d(s_i, s_j) < \varepsilon\}$
6: **if** Q with relaxation
7: **apply** (10) and (11) for all mentioned features
8: **if not**
9: **if** Q with context
10: **apply** (12) and (13)
11: **if not**
12: **compute** a subset {si} of the set found before step12
13: **select** {sl} from the subset of step 12, with rank $(s_l) > \delta_1$ and $r_{feedbakc} > \delta_2$ [see (16) and (17)]
14: **select** a subset for the subset of step 13
15: **return** the subset of step 14
16: **end**

Note that the output of step 15 might be an empty set, or a set having many recommended services complying to the query conditions.

The complexity of the algorithm is given mostly by the number of service features that can be considered with relaxations.

A variation of the algorithm was experimented with relaxation conditions for a set of contexts. The number of features with relaxation, the number of contexts, and the number of services into a cluster determine the performance of the algorithm.

Different experiments on the on-line Barnes & Nobles system (on-line bookstore) show a reasonable improvement on the precision the algorithm returns after running various numbers of queries and varying different conditions.

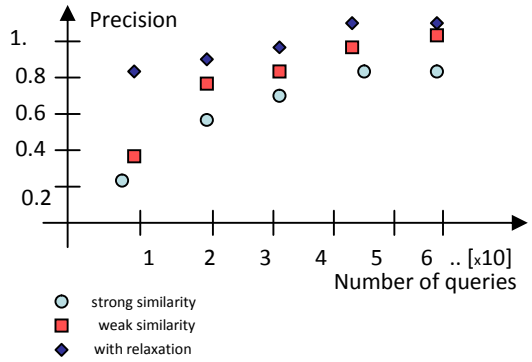

Figure 3. Precision of service returned to queries with different types of similarities

It is no surprise that a service satisfying a query with relaxation reaches faster and with a higher precision the query expectation.

IV. CONCLUSION AND FUTURE WORK

In this paper, we presented an approach for service invocation using similarity taxonomy with weak, strong, and normal similarity. Practically, services are clustered and service distance/similarity metrics were adopted from text-based domains. A reputation-based mechanism (introduced in [1][2]) is used in combination to context-based similarity and feature relaxation methods to identify a set of services that better serve a given query.

We also introduced the techniques of feature aggregation when similarity is evaluated, and the continuous update of feature classification, i.e., primary/secondary, according to the context. More work should be done on these two items, as semantic-based aggregation should be considered.

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-105-2

REFERENCES

1. O. Dini, P. Lorenz, and H. Guyennet; An Enhanced Architecture for Web Recommenders, SERVICE COMPUTATION 2009, IEEE Press, pp. 372 – 378, ISBN: 978-1-4244-5166-1, Athens, Greece
2. O. Dini, P. Lorenz, A. Abouaissa, and H. Guyennet, Dynamic Feedback for Service Reputation Updates, ICAS 2010, pp. 168-175 ISBN: 978-1-4244-5915-5, Cancun, Mexico
3. C. Wu and E. Chang, Searching Services 'in the web': A Public Web Services Discovery Approach, SITIS 2007, The Third IEEE Conference on SignalImage Technologies and Internet-based Systems, pp. 321-328.
4. M. Paolucci, B. Shishedjiev, Xh. Zenuni, and B. Raufi, GHSOM-based Web Service Discovery, 2010 European Computing Conference, ISSN: 1790-5117, 2010
5. M. Szomszor, C. Cattuto, H. Alani, K. O'Hara, A. Baldassarri, V. Loreto, and V. D. Servedio, "Folksonomies, the semantic web, and movie recommendation," In 4th European Semantic Web Conference, Bridging the Gap between Semantic Web and Web 2.0, 2007.
6. X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, Similarity Search for Web Services, The 30th VLDB Conference, Toronto, 2004
7. S. Cost and S. Salzberg, A Weighted Nearest Neighbor Algorithm for Learning Symbolic Features. Machine Learning, No. 10, 1993, pp. 57-78
8. L.S. Larkey and W. Croft, Combining Classifiers in text Classifications Techniques, ACM SIGIR 1998.
9. H.-H. Do and E. Rahm, COMA – A System for flexible Combination of Schema Matching Approaches, VLDB 2002
10. A.M. Zaremski and J.M. Wing, Specification matching of software components. TOSEM, No. 6, pp. 333-369, 1997
11. C. Bouras and V. Tsogkas, Improving text summarization using noun retrieval techniques, LNCS, Knowledge-based Intelligent Information and Engineering Systems, vol. 5178/2008, pp. 593-600

Copyright (c) IARIA, 2010 ISBN: 978-1-61208-105-2