Stefan Schulte, Ulrich Lampe, Julian Eckert, Ralf Steinmetz: LOG4SWS.KOM: Self-Adapting Semantic Web Service Discovery for SAWSDL. In: IEEE 2010 Fourth International Workshop of Software Engineering for Adaptive Service-Oriented Systems at 2010 IEEE 6th World Congress on Services (SERVICES 2010), pp. 511-518, IEEE Computer Society, July 2010. ISBN 9780769541297.

LOG4SWS.KOM: Self-Adapting Semantic Web Service Discovery for SAWSDL

Stefan Schulte, Ulrich Lampe, Julian Eckert, and Ralf Steinmetz

Multimedia Communications Lab, Technische Universität Darmstadt, Germany
schulte@KOM.tu-darmstadt.de

Abstract

In recent years, a number of approaches to semantic Web service matchmaking have been proposed. Most of these proposals are based on discrete and thus relatively coarse Degrees of Match (DoMs). However, different basic assumptions regarding the generalization and specialization of semantic concepts in ontologies and their subsequent rating in matchmaking exist. Hence, most matchmakers are only properly suitable if these assumptions are met.

In this paper, we present an approach for mapping subsumption reasoning-based DoMs to a continuous scale. Instead of determining the numerical equivalents of the formerly discrete DoMs manually, these values are automatically derived using a linear regression model. This permits not only easy combination with other numerical similarity measures, but also allows to adapt matchmaking to different basic assumptions.

These notions are implemented and tested in LOG4SWS.KOM – a matchmaker for SAWSDL that provides very good evaluation results with respect to Information Retrieval metrics such as precision and recall.

1. Introduction

The discovery of Web services which provide the right functionality with respect to a given service request is one of the essential requirements if regarding the vision of automated service invocation. In order to overcome the shortcomings of syntax-based service descriptions, several researchers have proposed the usage of semantic information in Web services, resulting in the concept of semantic Web services (SWS) [13]. Today, SWS are a prominent field of research and have resulted in a number of different approaches and specifications like *OWL-S* or *SAWSDL*, i.e., service description languages which explicitly make use of semantic technologies in different parts of a service description. One of the primary application areas of SWS is service discovery. Discovery of services is comprised by three aspects: (i) The ability of service providers to describe their services, (ii) the ability of requesters to describe their requirements towards services, and (iii) the effectiveness of the service matchmaker, i.e., an algorithm that finds the best fitting services from a set of offers based on a given service request [8].

Service matchmaking that considers semantic information is contemplated by a very agile research community, with a large number of different approaches having been proposed in recent years. A lot of experimentation is conducted concerning which elements from a service description should be regarded, applied similarity measurements, and how the resulting similarity values are combined [8].

In this paper, we present LOG4SWS.KOM ("Logic-based Matchmaking for Semantic Web Services"), a service matchmaker for WSDL 2.0 and SAWSDL-based service descriptions. Based on an Ordinary Least Squares (OLS) estimator, it maps the usually discrete subsumption reasoning-based DoMs to numerical equivalents, allowing the arbitrary combination with other numerical similarity measures, such as path length between concepts or common measures from the field of Information Retrieval (IR). LOG4SWS.KOM is not restricted to the service signature (i.e., inputs and outputs), but integrates information from different service abstraction levels.

LOG4SWS.KOM is, per se, purely semantic-based, based on the notion that explicit semantic information provides far more certainty than non-logic information can [12]. However, we provide a syntax-based fallback strategy which can be applied if the semantic annotations on a certain service abstraction level are incomplete or missing. As will be presented in the evaluation, LOG4SWS.KOM is capable of competing with and outperforming state-of-the-art matchmakers for SAWSDL like URBE [17] or SAWSDL-MX [9] with respect to metrics such as precision and recall.

The remaining part of this paper is structured as follows: In the next section, we analyze the shortcomings of current matchmaking approaches, which can potentially lead to suboptimal matchmaking results or a limited applicability. Afterwards, SAWSDL, the service standard that LOG4SWS.KOM is applied to, is briefly presented. In Sec-

978-0-7
DOI 10.

tion 4, the general considerations leading to the implementation of LOG4SWS.KOM will be presented. This includes a detailed discussion of the OLS estimator. We evaluate differently configured variants of LOG4SWS.KOM and compare the results with other matchmaking approaches for SAWSDL (Section 5). A discussion on related work can be found in Section 6. The paper closes with a conclusion and a brief outlook on our future work.

2. Background

Many semantic-based service matchmakers are based on the groundbreaking work of Paolucci et al., which defines the four distinct DoM levels of *exact*, *plugin*, *subsume*, and *fail* [16]. The DoMs are based on logic subsumption matching, i.e., the ancestry relationships between concepts in an ontology and can be found in similar or extended form in, e.g., [1, 4, 9]. As the DoMs are distinct, it is not possible to directly combine them with numerical similarity values from the field of IR, which are also frequently applied as supplement to the logic-based DoMs (e.g., by Klusch et al. [9]). Here, we propose the usage of numerical equivalents for the DoMs. While this has been done in other approaches, the authors do not provide a justified relationship between numerical similarity values and discrete DoMs depending on the Web services regarded. Instead, e.g., Liu et al. or Fernández et al. arrange for the usage of numerical values but do not specify how these values should be calibrated [6, 11].

Usually, there is a predetermined order of DoMs – e.g., Paolucci et al. define this as $exact > plugin > subsumes > fail$. Paolucci et al. reverse the meaning of *plugin* and *subsumes* for outputs and inputs – for former, a *plugin* match specifies that the class used in a service offer is more *generic* than this used in a service request. For inputs, a *plugin* match indicates a more *specific* class used in a service offer than in the service request. Bellur et al. and Cardoso reverse this ranking regarding *subsumes* and *plugin* respectively invert the meaning of these DoMs, which leads to a different ranking [1, 4]. Cardoso argues that a more generic *input* (i.e., a *subsumes* relationship) is favorable, because it will certainly accept the input which has been specified in the service request. In case of *outputs*, a reversed ranking scheme is proposed: Service offers with a more specific output are favored over those with a more generic output. Here, Bellur et al. argue that Paolucci's assumption that a service provider who advertises a certain output (i.e., a semantic concept describing the output) will deliver every subclass of this output, is not realistic.

Both ranking scheme approaches have their justification, depending upon which assumption regarding the generalization and specialization of semantic concepts in an ontology holds true. Hence, it would be useful to derive

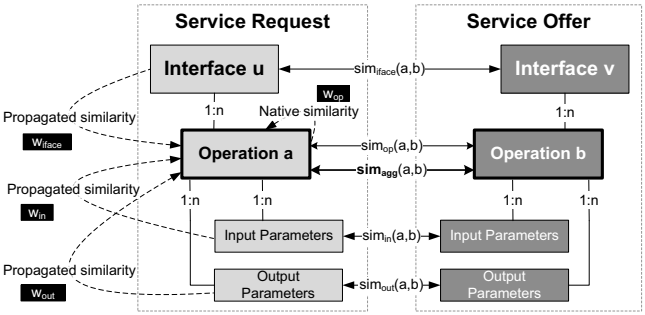

Figure 1. Matchmaking Process

the ranking of these DoMs automatically. This applies to SAWSDL in particular, as it does not define the "meaning" of semantic annotations on different service abstraction levels (cp. Section 3). With LOG4SWS.KOM, this ranking is conducted automatically – as the numerical equivalents are automatically derived, it is not necessary to predefine a DoM ranking.

As a short wrap up, LOG4SWS.KOM provides the following benefits: (i) It is possible to easily combine different syntax- or semantic-based similarity measures without dropping the usage of subsumption reasoning-based DoMs, (ii) it is not necessary to predefine a ranking of DoMs, (iii) instead of being restricted to the service signature, LOG4SWS.KOM integrates information from all available service abstraction levels. LOG4SWS.KOM complements and pursues already existing work in the field by combining accepted techniques and new ideas, resulting in very good service discovery results (cp. Section 5).

3. Service Descriptions using SAWSDL

SAWSDL ("Semantic Annotations for WSDL and XML Schema" [5]) is the W3C's recommendation for augmenting WSDL-based service descriptions ("Web Service Description Language" [2]) with semantic information. It provides a bottom-up, lightweight approach to SWS.

The abstract part of a WSDL document contains components with different levels of abstraction. For WSDL 2.0, these service components are interfaces, operations, and message parameters (defined as XML schema (XSD) types) [2]. Interfaces constitute the highest level of abstraction in WSDL; they subsume one or more operations. Operations point to XSD types, i.e., parameters, via their input and output message components. Thus, parameters are the lowest level of abstraction. As parameters refer to inputs and outputs, they are also called "service signature". For an overview, please refer to Figure 1, which shows the different levels of abstraction in SAWSDL.

SAWSDL introduces three new XML attributes into

WSDL, which can be used in conjunction with the already existing WSDL components [5]. In the context of this paper, *modelReferences* are the most important element of SAWSDL. modelReferences point to one or more URIs of arbitrary concepts that semantically describe a WSDL component. These semantic annotations allow the deriving of a service's purpose or the automatic conducting of comparisons between a pair of services. SAWSDL attributes are defined for WSDL interfaces, operations, and faults, as well as XSD simple and complex types, elements, and attributes.

SAWSDL imposes no restrictions on what a semantic annotation actually means or the type of semantic concept that is addressed. Regarding the former aspect, the meaning of, e.g., a modelReference on interface level, is not defined. However the SAWSDL specification does state that on interface level, a modelReference might be a categorization, while on operation level, a modelReference might specify a high level description of the operation. On message respectively XSD level, modelReferences most likely define data semantics [5]. This meaning of semantic annotations is compliant with the classification made for *WSMO-Lite* by Vitvar et al. [19] and will be the foundation for the matchmaker presented in the work at hand.

Likewise, SAWSDL does not restrict the type of semantic concepts a modelReference should point to. The only requirement is that the concepts are identifiable via URI references. On one hand, this allows for a maximum of flexibility. On the other hand, it poses an obstacle with respect to automatic processing and interpretation of concepts. In the context of our work, we will thus assume the semantic concepts to be formally defined in an OWL DL ontology. A second constraint it that we only consider the first URI from a modelReference; this is done as it is unclear if a further URI addresses an alternative semantic concept, an additional information or something else. Third, service requests are defined as a service that would perfectly match the request. Fourth, the matchmaker returns a result set arranged in a descending order regarding the computed similarity between service request and service offers. All these are common assumptions made in conjunction with the use of SAWSDL in matchmaking.

4. LOG4SWS.KOM: Self-Adapting Logic Subsumption Matching

To put it simply, matchmaking can be divided into the identification of data items to be matched, the measurement of similarities, and the actual matching of components. In the following, the way in which LOG4SWS.KOM addresses these three aspects of service matchmaking will be presented. The approach is implemented in Java using *Pellet* 2.0 for logical reasoning and *JWNL* 1.4 for accessing the *WordNet* ontology [14].

4.1. Operations-focused Matching

LOG4SWS.KOM employs a matching approach which focuses on the matching of *operations*. The underlying idea is that operations provide the essential functionality a service requester is looking for. Thus, for each requested operation, the best matched operation in the service offer should be identified. Accordingly, the overall similarity of two services relates to the degree to which their respective operations match, i.e., to which the offer provides the requested functionality.

The overall matchmaking process is depicted in Figure 1. For each pair of operations in service request and offer, their respective input (sim_{in}), output (sim_{out}), native operation (sim_{op}), and interface (sim_{iface}) level similarity is computed. These individual similarities are then combined using specified weights, w_{in}, w_{out}, w_{op}, and w_{iface}, yielding an aggregated similarity value sim_{agg} for each pair of operations. Formally, for a pair of operations, a and b, we define:

$$w_{iface} + w_{op} + w_{in} + w_{out} = 1 \tag{1}$$

$$\begin{aligned} sim_{agg}(a,b) = & sim_{iface}(a,b) * w_{iface} \\ & + sim_{op}(a,b) * w_{op} \\ & + sim_{in}(a,b) * w_{in} \\ & + sim_{out}(a,b) * w_{out} \end{aligned} \tag{2}$$

Once similarities between all pairs of operations in a service request and service offer have been computed, the overall service similarity sim_{serv} is derived by finding an optimal matching of operations. I.e., the final matching for a pair of services is conducted between their respective union set of operations, disregarding how the operations are actually organized into interfaces. Formally, let I and J be the sets of operations in a service request R and offer O respectively. Let x_{ij} be a binary variable, indicating whether $i \in I$ has been matched with $j \in J$. Then,

$$sim_{serv}(R,O) = \frac{1}{|I|} * \sum_{i \in I, j \in J} x_{ij} * sim_{agg}(i,j) \tag{3}$$

The matching of sets of components (specifically, inputs, outputs, and operations) is based on bipartite graphs as presented by Bellur et al. [1]. It perceives the sets of components of a service request and offer as two partitions of nodes in a graph. Each node in the first partition is connected with each node in the second partition through a weighted edge. The edge weights correspond to the respective similarity between two components. Using the *Hungarian* (or *Kuhn-Munkres*) algorithm, the bipartite graph matching algorithm computes a 1-on-1 assignment of components. Each component of the request is matched with one component of the service offer while maximizing the overall edge weight. In order to meet differing cardinalities

of the two sets, an extension of the Hungarian algorithm is applied [3].

Subsequent to the matching process, the edge weights of all matched edges are summed up and divided by the cardinality of the original sets. This yields the similarity for two sets of components. If the cardinality of the two sets differs, the following strategy is followed: Generally, the cardinality of the set associated with the service request is decisive. This implies, if an offer lacks requested operations or outputs, its overall similarity decreases. For inputs, the cardinality of the set associated with the service offer is decisive. I.e., if an offer requires more inputs than the request provides, its overall similarity decreases.

This procedure does not exclude any services offers due to a mismatch in the number of parameters or operations. Instead, these offers are implicitly punished by a reduction in service similarity. The approach is based on the notion that such service offers may still be able to provide a part of the initially requested functionality or outputs, or may be invoked by providing additional inputs.

If sim_{serv} is identical for two service offers in the result set, the offers are alphabetically arranged.

4.2. Assignment of Similarities

Our primary strategy for similarity assessment is based on the traditional DoMs from logic subsumption reasoning, as suggested by Paolucci et al. [16]. We slightly depart from their definition by defining generic types of matches that can be applied to each service abstraction level and type of parameter. Given two arbitrary concept, A and B, where A is defined in the service request and B is defined in the service offer, the DoM is given by

$$DoM(A,B) = \begin{cases} exact & if A \equiv B \\ super & if A \sqsubseteq B \\ sub & if A \sqsupseteq B \\ fail & else \end{cases} \quad (4)$$

According to the scheme by Paolucci et al. [16], our DoM of *super* corresponds to *subsumes* and *plugin* for inputs and outputs respectively; for *sub*, the order is reversed, and it corresponds to *plugin* and *subsumes* for inputs respectively outputs. In contrast to other approaches (cp. Section 2), we do not predefine a ranking of DoMs.

These DoMs provide a discrete scale and thus only allow a coarse-grained ranking of services. Yet, a more sophisticated ranking might be possible, for instance based on the path length between two concepts or the overall number of siblings a concept has. Thus, we map the four discrete DoMs onto an continuous numerical scale, ranging from 0 (no similarity at all) to 1 (perfect similarity). This approach allows the combination with other numerical measures and a much more fine-grained ranking of services. In detail, on each individual matching level $L \in \{iface, op, in, out\}$, each DoM $D \in \{exact, super, sub, fail\}$ is assigned a numerical equivalent, $d_{L.D}$, in the range $[0;1]$. Formally,

$$d_{L.D} \in [0;1] \quad \forall \quad \begin{array}{l} L \in \{iface, op, in, out\}, \\ D \in \{exact, super, sub, fail\} \end{array} \quad (5)$$

To allow a more fine-grained similarity assessment, in case of a super or sub match, the DoM's numerical equivalent is merged with the path length between two concepts. One intuitive approach to conduct such merging is simply dividing the numerical equivalent by the path length, based on the assumption that the similarity between two concepts (linearly) shrinks with their distance in an ontology.

Formally, let $PL(A,B)$ denote the shortest path between the two concepts A and B in an ontology. Furthermore, let L be the level on which the matching of components that point to these concepts is conducted. Then, the similarity $cs(A,B)$ between A and B (and thus, the two underlying components) is given by:

$$cs(A,B) = \begin{cases} d_{L.exact} & if A \equiv B \\ d_{L.super}/PL(A,B) & if A \sqsubseteq B \\ d_{L.sub}/PL(A,B) & if A \sqsupseteq B \\ d_{L.fail} & else \end{cases} \quad (6)$$

4.3. OLS-based Determination of Numerical DoMs

While mapping discrete DoMs to a continuous numerical scale offers advantages, it involves the ambiguous process of defining equivalents. To circumvent this issue, LOG4SWS.KOM applies an OLS estimator for the determination of optimal numerical DoM equivalents.

The process is based on the notion that a dependent variable $y_L^{a.b}$ – corresponding to the similarity of two operations, a and b, on a certain matching level, L, – can be derived through the linear combination of a set of independent variables $x_{L.D}^{a.b}$, corresponding to the frequency of a certain DoM D when matching a and b on that level. I.e., we assume that the weighted linear combination of the different DoM's frequencies predicts the similarity of two operations.

The OLS estimation is independently conducted for each matching level. I.e., the numerical weights differ for inputs, outputs, operations, and interfaces. As training data, a set of services is required along with a predefined similarity (or relevance) rating. A subset of a test collection, such as the SAWSDL-TC (cp. Section 5.1), fulfills this requirement.

In the training phase, LOG4SWS.KOM matches all pairs of operations in all service requests and offers. For each pair and each matching level, it stores the types of subsumption matches in the matched components along with

the path length, and determines the predefined similarity between the two operations (or, if the similarity is unavailable at the operations level, of the parent interfaces or services). Latter yields the vectors of predictors (y_{iface}, y_{op}, y_{in}, and y_{out}) for the OLS process, with each entry corresponding to a pair of operations.

The design matrices (X_{iface}, X_{op}, X_{in}, and X_{out}) are derived in the following manner: Each pair of operations yields one row with four entries, where each entry corresponds to the frequency of a certain type of DoM with respect to all matched components on a certain level. In detail, the frequency count is incremented by 1 for an exact and fail match between two components. For super and sub matches, it is incremented by 1 divided by the path length. The row is finally divided by the total number of matched components on the current level.

Assume, for instance, that a pair of operations, a and b, is matched. If each operation contains two inputs, there will be two input matches. Let the first match be an *exact* match and the second a DoM of *super* with a path length of 2. This yields the following row in the input level's design matrix (where the first entry corresponds to an *exact* DoM, followed by *super*, *sub*, and *fail*):

$$x_{in}^{a.b} = \begin{pmatrix} 1/2 & \frac{1}{2}/2 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0.5 & 0.25 & 0 & 0 \end{pmatrix} \quad (7)$$

Under the assumption that the operations a and b are relevant with respect to a binary relevance grading (which translates into a similarity of 1), the corresponding entry in the vector of predictors is simply $y_{in}^{a.b} = (1)$.

Given a design matrix and vector of predictors, the standard OLS estimator can be applied in the following manner on each matching level L:

$$\hat{\beta}_L = (X_L' X_L)^{-1} X_L' y_L \quad (8)$$

$\hat{\beta}_L$ corresponds to the optimal estimate of numerical weights. To derive the actually utilized vectors of weights, d_L, all entries are mapped to the range $[0;1]$. For that matter, the minimum value in the vector is added to all entries. Then, all entries are divided by the new maximum value. This ensures that all similarity values will also be in the specified range.

For the purpose of cross-validation (cp. Section 5.1), the design matrices and vectors of predictors are adapted. Specifically, all rows are removed from both the matrix and vector that either refer to the currently validated service request or do not refer to a service offer in the previously determined set of relevant service offers. This process ensures that no information which derives from the currently validated service request is used in the OLS regression in the evaluation.

4.4. Fallback Strategy & Caching

If there are no semantic concepts associated with components or the processing fails, a fallback strategy comes into effect. More precisely, the similarity between associated concept (and alternatively, component) names for a pair of components is computed using the WordNet ontology [14]. Again, the similarity is specified by a numerical value between 0 and 1.

All names undergo the following processing: First, they are split into individual tokens, based on commonly used separators, such as underscore ("_") and dash ("-"), or the popular *camelCase* notation. Tokens that do not correspond to a word in the WordNet ontology are additionally scanned for meaningful substrings in a recursive manner. That way, names such as "get_flight_price", "getFlightPrice", and "getflightprice" can be effectively split into individual (English) words. Each set of words constitutes a partition for a bipartite graph. The edge weight corresponds to the inverse of the minimal distance of a pair of words in WordNet. Consecutively, bipartite graph matching is employed, with the average edge weights in the matching yielding the similarity of the two names and thus, two service components.

To improve the performance of LOG4SWS.KOM in terms of query response time, we utilize different caches which may be populated both at registration and query time and may be permanently stored for future reference. In detail, caches exist for the types of subsumption matches and path lengths between concepts, WordNet distances, and splitting of names into words. The caches are filled when the corresponding data item is referenced in a service request for the first time.

5. Experimental Evaluation

5.1. Evaluation Setup

SAWSDL-TC1[1] has been adopted as test data collection for LOG4SWS.KOM. SAWSDL-TC1 was released in 2008 and consists of 894 semantically annotated WSDL 1.1-based Web services, which cover differing domains from education and medical care to food and travel, etc. The set contains 26 queries usable for matchmaking evaluation. A relevance set is provided for each query which can be applied in order to compute IR evaluation measures. SAWSDL-TC and its equivalent for OWL-S (*OWLS-TC*) are well-accepted in the SWS matchmaking research community and are for example applied in the S3 Contest [10].

As SAWSDL-TC1 is WSDL 1.1-based, it was necessary to convert the test collection to WSDL 2.0, which is the

[1] http://www.semwebcentral.org/projects/sawsdl-tc

designated service format in the work at hand. As no new semantic annotations have been added to or eliminated from the single service descriptions, it is possible to compare the matchmaking performance of LOG4SWS.KOM with matchmakers which make use of SAWSDL-TC1.

It is necessary to note that in SAWSDL-TC, semantic annotations exist solely at input/output parameter level. As mentioned in Section 4, LOG4SWS.KOM incorporates information from the interface, operation, and parameter level of SAWSDL. This means that the full potential of LOG4SWS.KOM should only be revealed if the annotations are more comprehensive and are addressed at all service abstraction levels. But as we will see in the evaluation results, LOG4SWS.KOM is able to provide competitive results under these circumstances.

We have performed six evaluation runs using different settings. In detail, we applied three different configuration of level weights both to a manually and OLS-tuned version of LOG4SWS.KOM. In these configurations, the level weights, $(w_{iface}, w_{op}, w_{in}, w_{out})$ have been set to (0, 0, 0.5, 0.5), (0.25, 0.25, 0.25, 0.25), and (0.1, 0.1, 0.4, 0.4) respectively. The first setup (referred to as *sig* in the following) solely operates on the service *signature*, i.e., input and output parameters, while the second (*eq*) gives *equal* weight to all levels. The third variant (*comp*) constitutes a *compromise*, putting the dominating weight on the parameter level. This accounts for the fact that the interface and operation levels are not semantically annotated in the test collection, which requires the application of the more error-prone fallback strategy. For the *manual* setup (referred to as *man* in the following), the numerical DoM equivalents have been manually set to 1 (for an exact DoM), 0.5 (super and sub), and 0 (fail). The choice of 0.5 is based on the notion that a super or sub match lies in between exact and fail matches, which intuitively correspond to 1 and 0 and has been originally proposed by Syeda et al. [18]. For the *OLS*-based evaluation runs (referred to as *OLS* in the following), the numerical equivalents are identified using k-fold cross-validation [15]. In the example at hand, $k = 26$ as every query and corresponding relevance set from SAWSDL-TC serves as a partition from the service set.

We used the tool SME2[2] to compare our results with other state-of-the-art matchmaking algorithms for SAWSDL.

5.2. Applied Metrics

We made use of the following metrics, which SME2 automatically provides, in order to evaluate our matchmaking algorithm: $Precision = \frac{|A \cap B|}{|B|}$ and $Recall = \frac{|A \cap B|}{|A|}$ where A is the set of all relevant documents for a request and B is the set of all retrieved documents for a request.

[2] http://projects.semwebcentral.org/projects/sme2/

Table 1. Evaluation Results

	AP	RP	P(5)	P(10)
sig, man	0.692	0.613	0.831	0.773
sig, OLS	0.731	0.671	0.854	0.762
eq, man	0.711	0.646	0.931	0.819
eq, OLS	0.723	0.672	0.923	0.846
comp, man	0.722	0.649	0.954	0.815
comp, OLS	0.741	0.678	0.954	0.835
SAWSDL-MX2 [9, 10]	0.679		N/A	
URBE [10, 17]	0.727	0.651	0.867	0.796

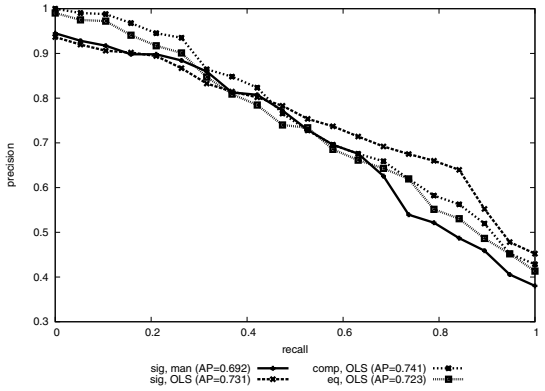

Figure 2. Evaluation Results

In order to compute the mean precision for answer sets returned at standard recall levels, we also made use of Macro-Averaged Precision $Precision_i = \frac{1}{|Q|} \times \sum_{q \in Q} \max\{P_0 | R_0 \geq Recall_i \wedge (R_0, P_0) \in 0_q\}$. SME2 uses equidistant steps $\frac{n}{\lambda}, n = 1 \ldots \lambda$ for the single recall and precision levels. Here, we made use of the default value $\lambda = 20$. To measure the precision on relatively low recall levels, Precision at o (*P(o)*) is employed. This value specifies the precision for the first ranked o data items; common values are, e.g., $o = 5$ or $o = 10$ (named *P(5)* or *P(10)*) [12]. R-Precision *(RP)* defines the precision for a certain number of returned (ranked) results to a specific query. Where P(o) defines one o over all queries, in RP this o differs from query to query and is equivalent to the relevant services for this query (i.e., the size of the result set provided in SAWSDL-TC). For RP, recall and precision are the same, hence it is equivalent to the *break-even point* of recall and precision. Last but not least, the Average Precision *(AP)* corresponds to the mean precision rate over all recall levels [12]. All these metrics are macro-averaged over all queries.

5.3. Results

Table 1 and Figure 2 show the evaluation results (for the purposes of Figure 2, some variants have been omitted). As is shown by the figures, the variants of LOG4SWS.KOM, which apply solely to the service signature, provide relatively low precision values for low recall levels (<0.35) while the variants that also incorporate interface and operation levels perform very well for these recall levels. This observation is supported by the P(5) and P(10) values, where the latter variants outperform the signature-based ones. All OLS-based variants perform very well, even at very high recall values and provide a precision value of >0.40 for recall = 1. The results for the altered weights for interface and operation levels show that the assumption that these levels should be weighted less than the semantically annotated inputs and outputs has proven right as $(comp, OLS)$ provides the overall best results.

All OLS-based variants perform better than their manual counterparts, however the gaps differ: While for the signature-only variants the gap is relatively large, it is smaller for the variants that also apply to the interface and operation level. This can be explained by the fact that the interface and operation levels are not semantically annotated in the test collection, i.e., OLS can only be applied to the parameter levels.

We compare LOG4SWS.KOM with SAWSDL-MX2 [9] and URBE [17] which were both contestants in the S3 Contest 2009 [10]. As depicted in Table 1, the (sig, OLS) and $(comp, OLS)$ variants of LOG4SWS.KOM outperform both matchmakers regarding the AP. All things considered, LOG4SWS.KOM performs very well with respect to common IR metrics. To the best of our knowledge, it is, so far, the best performing matchmaker for SAWSDL regarding the IR metrics applied in our evaluation. This demonstrates that the combination of different similarity measures – which may be relatively simple and coarse individually – can be an efficient approach for a fine-grained ranking of services.

Through the integration of OLS, the process of mapping DoMs to numerical equivalents can be conducted in a manner that is not subject to ambiguity. It also is an intuitive approach for adapting a matchmaker to a certain service domain. This includes, for instance, the training based on a representative selection of services which have been manually classified by a domain expert.

We would finally like to address a potential drawback of LOG4SWS.KOM. In matchmaking approaches that solely apply discrete DoMs, as proposed Paolucci et al. [16], the computed DoM for any operation can be assumed as a guaranteed lower bound of similarity for the request. In contrast, LOG4SWS.KOM determines the overall similarity of two services, which does not include such bound, or *global DoM*. Both approaches have their pros and cons: On the one hand, a global DoM guarantees a certain degree of compatibility with all elements – most importantly, parameters – in the request. On the other hand, LOG4SWS.KOM is rather tolerant toward outliers, i.e., lower DoMs that will else have a large impact on the overall DoM. Thus, LOG4SWS.KOM may also identify services that are suitable in principle, given some adaptations to, for instance, their parameters. Important to note, URBE, which provided the best results in the S3 Contest 2009 in terms of the AP, also omits the concept of a global DoM [17].

As a conclusion, we can say that the integration of OLS-based ranking/weighting of distinct DoMs leads to improved matchmaking results. The integration of interface and operation descriptions also leads to some improvements, especially regarding low recall levels. Regarding the question as to which of the evaluated variants of LOG4SWS.KOM should be used, it needs to be noted that this depends upon the regarded service domain. However, in our evaluation domain, we recommend to make use of the variant $(comp, OLS)$ as it provides outstanding results for low recall levels as well as provides the best AP (0.741).

6. Related Work

The number of matchmakers for SAWSDL-based Web services is, to the best of our knowledge, still quite manageable. From the last S3 Contest for SAWSDL matchmakers [10], URBE [17] and SAWSDL-MX [9] have been the most prominent challengers to LOG4SWS.KOM (which participated as a beta version in the contest).

URBE utilizes linguistic as well as logic information. The authors use a bipartite graph matching algorithm for both the inputs and outputs of a given service request and offer. Weights for the edges are calculated using two functions, i.e., by first measuring the similarity between input and output names. The second function assesses the associated XSD data type for a given pair of inputs or outputs, where a predefined similarity value is used, depending on the information loss that occurs in converting between the two types. As an extension, the authors also present a semantic similarity function, which replaces the name similarity measure if semantic annotations are available. When provided with two concepts, this function measures the path length between them in an ontology [17].

Extending the family of "MX"-matchmakers, Klusch et al. provide SAWSDL-MX in different variants [9]. Their approach calculates three kinds of similarity, based on logic, textual information, and structure and adaptively learns the optimal aggregation of those measures using a given set of services. Matching is employed on the level of operations using a bipartite graph matching algorithm. Logic similarity is measured using an extension of the classic discrete

DoMs. Textual similarity is calculated using a variety of measures, based on a textual representation of the request's and offer's service signature. For a structural comparison of two services, the *WSDL Analyzer* tool is used, which takes into account name and type similarity between inputs and outputs.

To the best of our knowledge, Kiefer and Bernstein [7] have been the first to apply an adaptive approach to service matchmaking. Here, the authors use different linguistic-based similarity measures from *SimPack* and apply them to OWL-S service descriptions. The weighting of these measures is automatically determined using machine learning approaches from *Weka* and *LibSVM*.

7. Conclusion

In this paper, we presented LOG4SWS.KOM, a semantic matchmaker for SAWSDL. To the best of our knowledge, LOG4SWS.KOM provides the best results for a SAWSDL matchmaker regarding IR metrics like AP, P(5), and P(10). A special focus should be placed on the performance on low recall levels where very good precision values have been accomplished. There are still some possible enhancements that we wish to approach in our future work. First of all, it may be possible to improve matchmaking results by using syntax-based similarity values to complement the semantic-based DoMs. Second, it would be interesting to evaluate LOG4SWS.KOM regarding *graded* relevance (contrary to the *binary* relevance used in SAWSDL-TC1) which will hopefully be featured in a future version of SAWSDL-TC. Third, our fallback strategy is quite lightweight and could be replaced by a more sophisticated approach.

Acknowledgements

A special thanks goes to Patrick Kapahnke and Matthias Klusch from DFKI for the development of SME2. This work is supported in part by the E-Finance Lab e. V., Frankfurt am Main, Germany (`www.efinancelab.de`).

References

[1] U. Bellur and R. Kulkarni. Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching. In *2007 IEEE International Conference on Web Services (ICWS 2007)*, pages 86–93, 2007.

[2] D. Booth and C. K. Liu, editors. *Web Service Description Language (WSDL) Version 2.0 Part 0: Primer*. W3C Recommendation, 2007.

[3] F. Bourgeois and J.-C. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802–804, 1971.

[4] J. Cardoso. Discovering Semantic Web Services with and without a Common Ontology Commitment. In *IEEE Services Computing Workshops (SCW '06), Third International Semantic and Dynamic Web Processes Workshop (SDWP 2006)*, pages 183–190, 2006.

[5] J. Farrell and H. Lausen, editors. *Semantic Annotations for WSDL and XML Schema*. W3C Recommendation, 2007.

[6] A. Fernández, A. Polleres, and S. Ossowski. Towards Fine-grained Service Matchmaking by Using Concept Similarity. In *First International Joint Workshop SMR^2 2007 on Service Matchmaking and Resource Retrieval in the Semantic Web at ISWC 2007*, volume 243 of *CEUR Workshop Proceedings*, 2007.

[7] C. Kiefer and A. Bernstein. The Creation and Evaluation of iSPARQL Strategies for Matchmaking. In *5th European Semantic Web Conference (ESWC 2008)*, volume 5021 of *LNCS*, pages 463–477, 2008.

[8] M. Klusch. Semantic Web Service Coordination. In M. Schumacher, H. Helin, and H. Schuldt, editors, *CASCOM: Intelligent Service Coordination in the Semantic Web*, chapter 4, pages 59–104. Birkhäuser Verlag, Basel, Switzerland, 2008.

[9] M. Klusch, P. Kapahnke, and I. Zinnikus. SAWSDL-MX2: A Machine-Learning Approach for Integrating Semantic Web Service Matchmaking Variants. In *2009 IEEE International Conference on Web Services (ICWS 2009)*, pages 335–342, 2009.

[10] M. Klusch, A. Leger, D. Martin, M. Paolucci, A. Bernstein, and U. Küster. 3rd International Semantic Service Selection Contest – Retrieval Performance Evaluation of Matchmakers for Semantic Web Services (S3 Contest). 2009.

[11] C. Liu, Y. Peng, and J. Chen. Web Services Description Ontology-Based Service Discovery Model. In *2006 IEEE / WIC / ACM International Conference on Web Intelligence (WI'06)*, pages 633–636, 2006.

[12] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[13] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[14] G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[15] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, USA, 1997.

[16] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic Matching of Web Services Capabilities. In *First International Semantic Web Conference (ISWS 2002)*, volume 2342 of *LNCS*, pages 333–347, 2002.

[17] P. Plebani and B. Pernici. URBE: Web Service Retrieval Based on Similarity Evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 21(11):1629–1642, 2009.

[18] T. Syeda-Mahmood, G. Shah, R. Akkiraju, A.-A. Ivan, and R. Goodwin. Searching Service Repositories by Combining Semantic and Ontological Matching. In *2005 IEEE International Conference on Web Services (ICWS 2005)*, pages 13–20, 2005.

[19] T. Vitvar, J. Kopecký, J. Viskova, and D. Fensel. WSMO-Lite Annotations for Web Services. In *5th European Semantic Web Conference (ESWC 2008)*, volume 5021 of *LNCS*, pages 674–689, 2008.