UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

LEVERAGING WEB SERVICES DISCOVERY WITH CUSTOMIZABLE HYBRID MATCHING

Natallia Kokash, Willem-Jan van den Heuvel and Vincenzo D'Andrea

July 2006

Technical Report # DIT-06-042

Leveraging Web Services Discovery with Customizable Hybrid Matching

Natallia Kokash[1], Willem-Jan van den Heuvel[2], and Vincenzo D'Andrea[1]

[1] DIT - University of Trento, Via Sommarive, 14, 38050 Trento, Italy
{kokash,dandrea}@dit.unitn.it
[2] Infolab, Tilburg University, 5000 LE, PO Box 90158, Tilburg, The Netherlands
wjheuvel@uvt.nl

Abstract. Improving web service discovery constitutes a vital step for making the Service Oriented Computing (SOC) vision of dynamic service selection, composition and deployment, a reality. Matching allows for comparing service requests of users with descriptions of available service implementations, and sits at the heart of the service discovery process. During the last years, several matching algorithms for comparing user requests to service interfaces have been suggested, but unfortunately there are no consistent comparative experimental evaluations of existing service discovery methods. This paper firstly reviews several state-of-the-art approaches to matching using syntactic, semantic and structural information from service interface descriptions. Secondly, it evaluates the efficacy of several key similarity metrics that underpin these approaches, using a uniform corpus of web services. Thirdly, this paper develops and experiments with a novel style of matching that allows for blending various existing matching approaches and makes them configurable to cater service discovery given domain-specific constraints and requirements.

1 Introduction

Service Oriented Architectures (SOAs) offer tantalizing possibilities for enterprizes by allowing large-scale reuse of loosely-coupled services, defining service description, discovery and composition at heart of its paradigm [1]. Web services seem to become the preferred implementation technology for realizing the SOA promise of service sharing and interoperability. By now, a stack of standards and specifications supports the description, discovery, invocation, composition, security and deployment of web services, and many tools to develop applications from web services have become commercially available. However, the vision of dynamic composition of heterogeneous web services still seems far away.

Web service discovery is generally perceived a key step to reach automation of service composition, and further realizing the SOA vision. Web service discovery is concerned with locating web services that match a set of functional and non-functional criteria [2]. Given the fact that the description of web services has been standardized on WSDL, many of today's discovery approaches confine themselves to locating web services on the basis of their functional description,

but some novel techniques that take into consideration extra-functional descriptions, such as performance, are under investigation.

Discovery involves three interrelated phases: (1) matching, (2) assessment and (3) selection. During the first phase, the description of a service is matched to that of a set of available resources. Next, the result of matching (typically a set of ranked web services) is assessed, filtered by a set of criteria. Finally, services are actually selected so they may be subsequently customized and combined with others. This paper focuses on the first phase of service discovery - matching.

One of the prime ramifications of SOA in general, and web services more in particular, entails heterogeneity. Heterogeneity makes service discovery, and in particular service matching, a daunting task. Actually, service heterogeneity comes in many disguises, including syntactic-, semantic- and pragmatic heterogeneity. Syntactic heterogenity pertains to structural conflicts between services, whilst their semantics may actually be the same. Semantic heterogeneity often occurs at the level of lexical incompatibility, as the vocabulary that is used in web service descriptions may be different in various domains. Pragmatic heterogeneity refers to incompatibilities between web service protocols and QoS requirements of services [3]. While syntactic incompatibilities may still be detected based on WSDL specifications, semantic- and especially pragmatic heterogeneity conflicts between web services increasingly need richer service descriptions.

Existing approaches to web service matching tend to address syntactic and/or semantic matching. To analyze their merits, it is useful to further classify them as uniform or hybrid. Uniform matching approaches refer to atomic matching techniques that can not be any further decomposed in finer-grained matching techniques. Hybrid matching approaches on the other hand may combine various matching methods (e.g., syntactic and semantic) into a composite algorithm. Unfortunately, in many cases an evaluation of the proposed matching techniques is lacking. And, if available, each experimental evaluation uses its own corpus, making an objective judgement about their merits very cumbersome.

In the next section, we survey the prevalent styles of web service matching and review key research results for each of them. Section 3 then introduces a novel web service matching approach that draws upon existing approaches and serves to highlight the basic workings of hybrid matching. Next, Section 4 evaluates this new approach and compares it to other related matching techniques using a similar WSDL corpus. Based on these evaluations, Section 5 then proposes a customizable approach towards hybrid service matching. Section 6 concludes this paper by summarizing our main findings and exploring new research areas.

2 Related Work

We may discern between two streams in web service matching for the purpose of web service discovery: those based on WSDL using classical information retrieval techniques, such as the keyword-based search, and those assuming semantic descriptions of web services in semantically enriched versions of WSDL, most notably WSDL-S. The first category is predominantly applied in many commer-

cial discovery techologies, relying on standards such as UDDI and ebXML. The second category is experimented in the domain of the Semantic Web.

UDDI registries[3], the dominating technological backbone for web service discovery, rely mainly on the exhaustive categorization of the advertised services by their providers. The UDDI Registries' API supports only simple keyword-based matching, facilities which have proven to be insufficient due to their low precision (a fraction of retrieved documents that are relevant) and low recall (a fraction of relevant documents that have been retrieved). Several Information Retrieval (IR) techniques have been proposed as a way to improve the efficacy of service matching in UDDI. Sajjanhar et al. [4] have studied Latent Semantic Indexing (LSI), the prevailing method for small document collections, to capture the semantic associations between short textual advertisements of registered web services. Recently Corella et al. [5] have developed a heuristic approach for the semi-automatic classification of web services, assuming that a corpus of previously classified services is available.

EbXML registries[4] provide an alternative way to register, locate and access information about offered web services. This information largely exceeds the level of service definitions as captured in WSDL documents, and allows trading partners to advertise their business processes, core components and context descriptions (e.g., access/control constraints). The ebXML registry is equipped with a more sophisticated query model than UDDI, allowing the usage of custom ad-hoc and filtered queries (in SQL-92 syntax).

Now, we will outline several contemporary approaches to assess the similarity of web services based on WSDL specifications as they are the de-facto vehicle for representing services, although WSDL does not (yet) provide advanded means to capture semantic information. In fact, WSDL's only means to describe semantic information is using the `documentation` tag that may embrace service documentation, while assuming that tag descriptors of WSDL constructs such as operations and data types, are semantically meaningful.

In particular, Bruno et al. [6] have experimented with automated classification of WSDL files using Support Vector Machines (SVM). Moreover, Stroulia and Wang [7] have developed a suite of algorithms for WSDL similarity assessment. To enable semantic matching of web service specifications, the WordNet lexicon [8] was employed. WordNet entails a lexical database with words organized into synonym sets representing a lexical concept. The main drawback of this method is that poor, unnormalized heuristics (such as matching scores of 5 or 10) in assigning weights for term similarity are used. Dong et al. [9] present a search engine focused on retrieval of WSDL operations. The underlying assumption of their method is that parameters tend to reflect the same concept if they often occur together. In [10] web service similarity is determined using a WordNet-based distance metric. Zhuang et al. [11] apply a conceptually similar approach. Key shortcomings of this work however include the lack of automated preprocessing of WSDL files, complex name handling and structural informa-

[3] `http://www.uddi.org`

[4] `http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrim.pdf`

tion analysis. In [12], the COSMOS approach is introduced that is built on top of WordNet, adopts the Vector-Space Model (VSM) [13] and incorporates preprocessing techniques to clean-up terminology, stem WSDL elements, and the such.

Striving for the automated web service discovery and composition has lead to the idea of annotating services with semantic information. Two main proponents of semantic annotation entail WSDL-S and OWL-S. WSDL-S [14] provides a way to inject more real-world semantics to service specifications, being closely aligned to WSDL 2.0. Semantic descriptions are maintained outside of WSDL documents and referenced to using extensibility elements. OWL-S on the other hand, attaches description logic based meta-data to web services. Possible semantic meta-data include preconditions, inputs, outputs and effects of service operations. In fact ontological meta-data is not restricted to the just mentioned categories, but also includes information about domain-specific service capabilities, available resources, security and transactional behavior, and the like.

The METEOR-S [15] project combines semantic web technology (semantic annotation) with classical IR, providing a solution that extends UDDI registries with semantic information to leverage matching of web services in a P2P-environment. This solution also allows for semi-automatically mapping of WSDL elements to ontological concepts. This is achieved in the following manner. First, the linguistic similarity between concepts and elements is calculated. Then, the structural similarity between sub-trees of the concepts and elements is determined, after which the overall similarity is computed as the geometric mean of the structural and linguistic similarities. Syeda-Mahmood et al. [16] use domain-independent and domain-specific ontologies to match service descriptions. They combine both cues to determine an overall semantic similarity score and argue that the hybrid matching algorithm gives the better relevancy results than can be achieved using any one cue alone.

The above demonstrates the existence of many techniques to match web services. Unfortunately, in many cases empirical experiments to validate their efficacy are lacking. Clearly, in some cases precision and recall of matching algorithms has been determined, but, unfortunately, no uniform corpus of WSDL documents has been used, making a comparison between them infeasible.

3 The WSDL Matching Method (WSDL-M2)

In this section we propose a web service matching algorithm, named WSDL-M2, that was inspired by several existing discovery methods. It combines two techniques: *lexical matching* to calculate the linguistic similarity between concept descriptions, and *structural matching* to evaluate the overall similarity between composite concepts.

The overall matching process is shown in Figure 1. First, all files from the collection of WSDL specifications are parsed (by means of `wsdl4j` library[5]) in order to allow extraction of their structured content. In the second step, the

[5] `http://sourceforge.net/projects/wsdl4j`

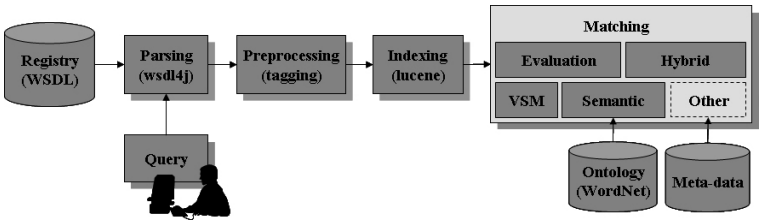


Fig. 1. Steps of WSDL-M2

parsed document is tagged to enable lexical analysis. The method considers five WSDL concepts that are supposed to contain meaningful information: *services*, *operations*, *messages*, *parts* and *data types*. Each element has a *description*, i.e., a vector that contains semantic information about this element extracted from the specification. To obtain concept descriptions from compound terms we excerpt (i) sequences of an uppercase letter and following lowercase letters, (ii) sequences of more than one uppercase letters in a row, (iii) sequences between two non-word symbols. For example, from `tns:GetDNSInfoByWebAddressResponse` we get the following set of words: *tns, get, dns, info, by, web, address, response*.

In the third step, the tagged WSDL specifications can be further analyzed and subsequently indexed using different IR models. The most widely-used IR technique constitutes the Vector-Space Model [13]. It weights indexed terms to enhance retrieval of relevant WSDL documents and then computes a similarity coefficient after which they may be ranked. Traditionally, term weights are assigned using the *tf-idf (term frequency-inverse document frequency)* measure. The `Lucene` library[6] was tailored to allow assigning relative coefficients. After this assignment, the similarity between descriptions is determined using associative coefficients based on the inner product of the description vectors.

To address the major shortcoming of VSM, the fact that VSM considers words at the syntactic level only, our method expands both the query and the WSDL concept descriptions using synonyms that are extracted from WordNet. Next, we compare the obtained word tuples as in VSM.

Finally, we used a measure that reflects the semantic relation between two concepts using the WordNet lexicon[7]. Formally, it is defined as $sim(c_1, c_2) = 1 - \big(ic_{wn}(c_1) + ic_{wn}(c_2)\big)/2 + \max_{c \in S(c_1,c_2)} ic_{ws}(c)$, where $ic_{wn}(c)$ denotes information content value of a concept c and $S(c_1, c_2)$ is a set of concepts that subsume c_1 and c_2. More details of this algorithm are outlined in [17]. Please note that in principle any other algorithm could have been chosen instead.

In the remainder of this article, we will refer to the above metrics as to $A1$ (VSM), $A2$ (VSM+WordNet) and $A3$ (Semantic). Using these metrics, the matching phase then continues with the comparison of service descriptions and the set of service operations, which are later combined in a single-number mea-

[6] `http://lucene.apache.org/`

[7] Java implementation of the algorithm that defines the semantic similarity of two terms is available on `http://wordnet.princeton.edu/links.shtml`.

sure. The similarity between operations, in their turn, is assessed based on the descriptions of operations and their input/output messages. To compare message pairs we again evaluate similarity of their descriptions and parts. Since one part with a complex data type or several parts with primitive data types can describe the same concept, we compare message parts with subelements of complex data types as well. We rely on "relaxed" two-level structural matching of data types since too strict comparison can significantly reduce recall. At the first level, description of complex types are compared. At the second one, all atomic subelements of the complex type are compared. For each element, names of higher-level organizational tags such as `complexType` or `simpleType` and composers such as `all` or `sequence` are included in the element description. Order constraints are ignored since parser implementations often do not observe them. This does not harm well-behaved clients and offers some margin for errors.

The structural matching is treated as *Maximum Weight Bipartite Matching* problem that can be solved in polynomial time using, for example, Kuhn's Hungarian method [18]. We use this method for two purposes: (i) to calculate semantic similarity between concept descriptions, (ii) to compute similarity of complex WSDL concepts taking into account their constituents (sub-types). Weight w_{ij} of each edge is defined as a lexical similarity between elements i and j. The total weight of the maximum weight assignment depends on the dimensions of the graph parts. There are many strategies to acquire a single-number dimension-independent measure in order to compare sets of matching pairs. The simplest of them is to calculate the matching average. Alternatively, we can consider two elements $i \in X$, $j \in Y$ to be similar if $w_{ij} > \gamma$ for some parameter $\gamma \in [0, 1]$. In this case, Dice, Simpson, and Jaccard coefficients may be applied.

Due to space reasons, we were not able to include the entire formal details of the matching algorithm. We refer to [19] for a detailed formal treatment.

4 Comparative Analysis of WSDL Matching Algorithms

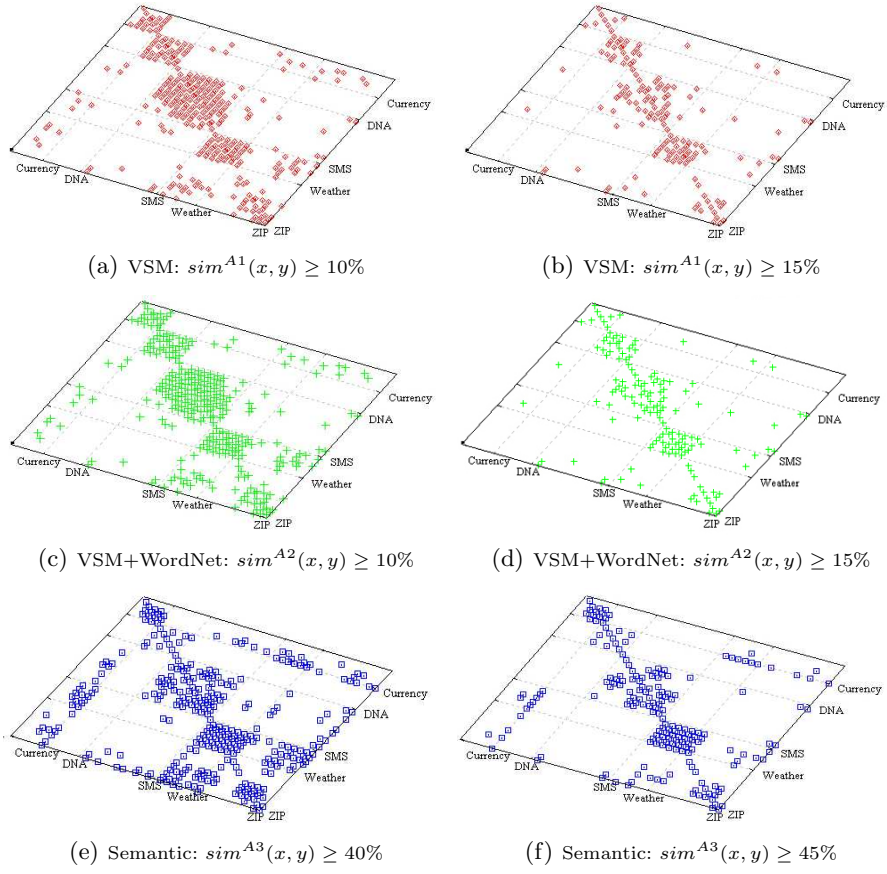
In this section we report on a series of experiments that we have conducted to evaluate the efficacy of the presented matching method. Further, we compare the efficacy of the *tf-idf* heuristic, its WordNet-based extension and the semantic similarity metric presented in the previous section.

We ran our experiments using a collection of 40 XMethods' web services classified into five categories [10]: *ZIP*(11), *Weather*(6), *DNA*(6), *Currency* (5) and *SMS*(12). Services from each group were verified manually to provide at least one common operation. In addition, we have experimented with a larger collection to increase confidence in some of our initial observations. For these purpose we decided to reuse the WSDL corpus that was developed by Stroulia and Wang [7]. It consists of 447 services[8] divided into 68 groups.

For each service we queried the complete data set for the relevant services (i.e., services classified in the same group). The matching results for different

[8] Stroulia and Wang describe a collection of 814 services. However, we excluded a group of 366 unclassified WSDL specifications and 1 WSDL file was not parsed correctly.

thresholds are shown in Figure 2. The similarity assigned to different files with respect to the query can be treated as a level of the algorithm *confidence*. It ranges from 0 (no match) to 1 (WSDL documents contain all concepts from the query regardless of the order). To avoid dependency from the chosen similarity threshold, the efficacy of the matching algorithm was accessed by *average precision* that combines precision, relevance ranking, and overall recall [20]. For each group, average precision value among all queries is given in Figure 3. A comparison of processing times is presented in Figure 4.



(a) VSM: $sim^{A1}(x, y) \geq 10\%$ (b) VSM: $sim^{A1}(x, y) \geq 15\%$

(c) VSM+WordNet: $sim^{A2}(x, y) \geq 10\%$ (d) VSM+WordNet: $sim^{A2}(x, y) \geq 15\%$

(e) Semantic: $sim^{A3}(x, y) \geq 40\%$ (f) Semantic: $sim^{A3}(x, y) \geq 45\%$

Fig. 2. Results of WSDL matching

As can be seen from our experimental results, the VSM was the most effective method for the overwhelming majority of the queries. Yet please note that it is limited to syntactic matching only. In contradiction to our intuition, application of the *tf-idf* heuristic applied on the WSDL specifications enriched with synonyms from the WordNet lexicon, did not improve the quality of the

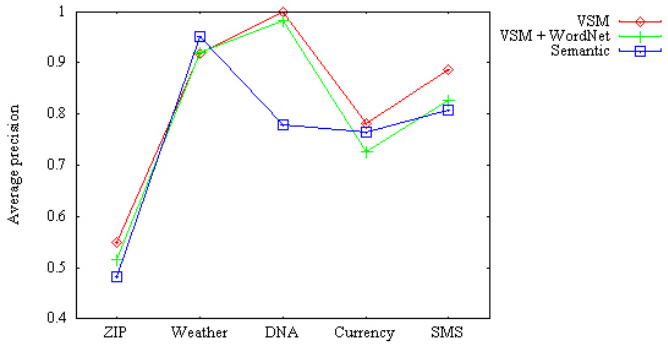


Fig. 3. Average precision on the first data set

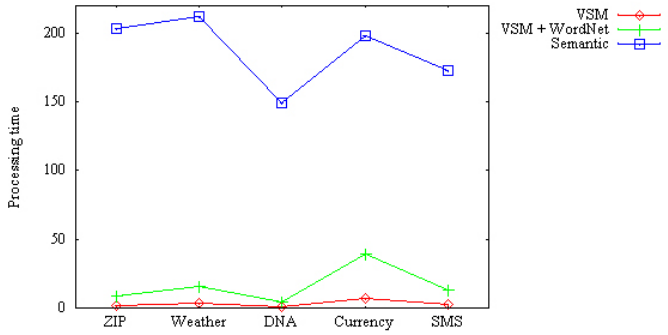


Fig. 4. Processing time on the first data set

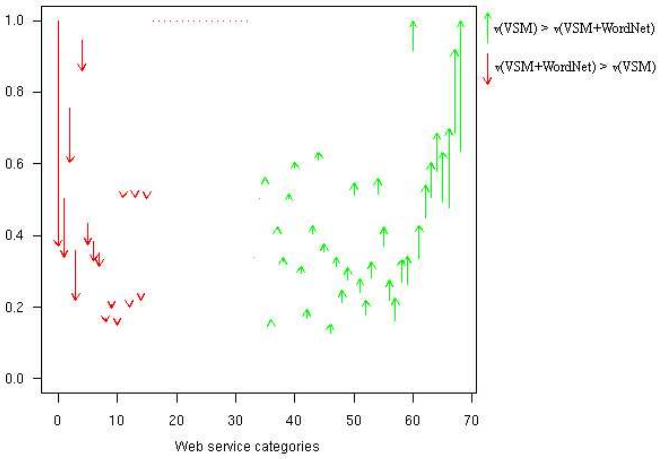


Fig. 5. Average precision of two matching methods on the second data set

matching results for the first collection. This can be observed in Figures 2(a)-2(d): the same files are classified as relevant by both methods. For the second data set, a slight gain in the average precision for 8 groups, along with notable decrease for 28 groups from the 68 examined, was observed 5. We carried out statistical analysis in order to further analyze the performance results. Wilcoxon signed rank tests [21] indicate that they are significantly different (p-value = 0.00693 < 0.01) and prove that the average precision of the $A1$ is consistently better (p-value = 0.003465 < 0.01) than the average precision of the $A2$. The semantic correlations between WSDL concepts found with the help of the WordNet-based semantic similarity measure in most cases are wrong or too weak. As a result many irrelevant files for the query have a high similarity score. This can be seen in Figure 2(e), where many off-diagonal elements corresponding to irrelevant services appear. Aside from the efficacy of this particular application of *tf-idf*, the semantic matching approach has a significantly lower performance than the previous two approaches.

Enriching element descriptions by synonyms from the WordNet lexicon leads to significant increase in index size (170% for the first collection and 80% for the second one) with no gain in average precision. We believe that the reason of this result is caused by an excessive generality of the WordNet[9]. At the same time, WordNet is restricted to a fixed set of six semantic relationships (including: hypernyms, synonyms, entailment, etc.). As a consequence, words "currency" and "country" are not recognized as related concepts, and the operations `getRateRequest(country1, country2)` and `conversionRate(fromCurrency, toCurrency)` had significantly lower similarity score than they were expected to have. Nevertheless, given a country the user can get the currency used in this country and invoke a service to exchange money that accepts currency codes as input. The similarity measure that we need is the relation "can be converted" rather than the general lexical (synonymy) similarity between concepts from the WordNet database. In addition, most of documentation tags in WSDL descriptions are empty. Since significantly more efforts are required to provide formal ontology-based descriptions of web services, this argues against our belief that semantic web services will be widely used.

For the collection of the classified web services WSDL-M2 proved to be very effective (46-100%). For several categories of the second collection average precision was dramatically low (15-40%) reaching 100% for the other groups in the same time. This can be explained by the fact that the most categories in the corpus were very coarse-grained and too generic in nature. As claimed before, due to the absence of a standard corpus in combination with the usage of different data models, WSDL-M2 method cannot be compared quantitatively with the existing approaches for which some empirical validations are available, notably, [7][9][10]. A challenging anomaly occurs: groups with better precision in [10] correspond to the groups with worse average precision in our experiments. This may be caused

[9] E.g., the following set of synonyms corresponds to the term *batch*: {*deal, flock, hatful, spate, lot, muckle, great, deal, wad, mickle, mint, clutch, mass, quite a little, good, deal, heap, peck, stack, pile, plenty, mess, raft, pot, whole, lot, sight, slew, tidy sum, whole slew*}. However, none of them is likely to be used in the context of concise web service specifications.

by a different proportion (weight) of structure vs. semantic similarity impact on the final similarity score. Further, our experiments do not support the conclusions by Stroulia and Wang [7] that enriching of the WSDL descriptions with synonyms from the WordNet leads to a better matching precision.

The above provides empirical evidence that one of the key factors that influence on the performance of the matching methods that we studied this far, is the quality of the vocabulary which was used in different categories. So, for categories "currency" and "weather" the semantic matching algorithm is quite effective while for "DNA" syntactic matching shows the better result. These observations point towards the idea that a *customizable hybrid* matching is needed to increase confidence in matching results. Hybrid matching can help to reduce the processing time of semantic matching (see Figure 4), using structural matching to reduce the number of WSDL documents to be compared.

5 Customizable Hybrid Matching Approaches

In this section, we introduce a *customizable hybrid approach* towards matching of web services. The significance of customization lies in the fact that it caters for composition of a new hybrid approach from various existing techniques. Hence, this approach may in fact be perceived as a *meta-matching strategy*, which is very flexible as it is not restricted to any matching technique, but enables ad-hoc composition of several (pre-existing) matching approaches.

It is of critical importance to make matching methods *customizable* so that they may be tailored to meet organization-, domain- and/or context-specific constraints and needs, including, (non-)absence of domain-specific taxonomies, quality of the request/available web service descriptions, availability of textual descriptions, number of available services, usage of topic-specific terminology and the such. For example, hybrid matching seems a viable solution in case one has more confidence in structural than semantic matching due to the fact that WSDL labels carry poor semantics and service descriptions are lacking. The level of confidence may be expressed by parameterizing the hybrid algorithm so that more weight can be assigned to structural and less weight to semantic matching.

Let us illustrate our approach using a simple example. Suppose that two kinds of matching algorithms are available: Sy, which compares service descriptions using syntax driven techniques and Se, that relies on semantic matching. Let $sim^{Sy}(q, x)$ designate a similarity score between query q and web service (operation) x defined by the syntactic matching algorithm, and $sim^{Se}(q, x)$ be a similarity score between query q and web service (operation) x defined by the semantic matching algorithm. Given query q and threshold $\gamma > 0$, let

$$X^A(q, \gamma) = \{x | sim^A(q, x) > \gamma\}$$

denote a set of services (operations) found by the algorithm A. Now, we propose three compositional operators to combine matching approaches:

- *Mixed* - a series of matching techniques are executed in parallel. In fact, these matching techniques may be homogenous (e.g., all of them of the same

type) or heterogenous (various types of matching, e.g., a mixture of semantic and syntactic matching). This type of composition combines sets of services found by the different matching techniques in a single list. For example, syntactic and semantic matching algorithms are grouped in a single ranking list, i.e.,

$$X^{H1a}(q,\gamma) = \{x | sim^{H1a}(q,x) > \gamma\},$$

where

$$sim^{H1a}(q,x) = f\{sim^{Sy}(q,x), sim^{Se}(q,x)\}$$

such that $f = \{max, min\}$. Alternatively, the results of the two approaches are fused based on weights allocated to each matching constituent, i.e.,

$$X^{H1b}(q,\gamma) = \{x | sim^{H1b}(q,x) > \gamma\}$$

such that

$$sim^{H1b}(q,x) = w_1 sim^{Sy}(q,x) + w_2 sim^{Se}(q,x), \mid w_1 + w_2 = 1,\ 0 \leq w_1, w_2 \leq 1.$$

– *Cascade* - each matching technique that is part of the composite matching algorithm reduces the searching space of relevant service specifications. In other words, each subsequent matching algorithm refines the matching results of the previous one. For example, given a set of services found by the syntactic matching algorithm, choose those services whose similarity that is computed by the semantic matching algorithm is higher than a predefined threshold, i.e.,

$$X^{H2a}(q,\gamma_1) = \{x | sim^{H2a}(q,x) > \gamma_1\},$$

where

$$sim^{H2a}(q,x) = sim^{Sy}(q,x) \mid x \in X^{Se}(q,\gamma_2).$$

Alternately, from the set of services found by the semantic matching algorithm, we select those services whose syntactic similarity is higher than a predefined threshold, i.e.,

$$X^{H2b}(q,\gamma_2) = \{x | sim^{H2b}(q,x) > \gamma_2\},$$

where

$$sim^{H2b}(q,x) = sim^{Se}(q,x) \mid x \in X^{Sy}(q,\gamma_1).$$

– *Switching* - this category of composition allows to switch between different matching algorithms. In principle, the decision to switch to another matching technique is driven by predefined criteria. For example, based on the number of the found services for a query after using a uniform algorithms we can alter between cascade and mixed combinations.

Parametrization entails a prime mechanism to allow for customization. Principally, the weights may be applied to hybrid matching techniques, being assembled using mixed, cascading and switching styles of composition. In fact, configuration of hybrid matching may not only be achieved at the level of matching techniques, but also at the level of the data models underlying them. There exist two fundamental choices to combine data-models underpinning hybrid matching:

– *Combination* - different data models are combined by a single matching algorithm. For example, a hybrid matching technique may inject a cocktail of syntactic and semantic data that is extracted from WSDL files.
– *Augmentation* - denotes an alternate combination strategy in which various data-models are used sequentially to enrich the information that serves as an input to the matching process. This style of data-model combination is most effective for cascading or switching style of composition. For example, in the first step of the matching process pure syntactic data may be considered, after which lexical information is extracted from WSDL interfaces, which in the next step is enriched by OWL-S descriptions, and then even further with domain- or community specific knowledge. Typically, this combination strategy involves interaction with the service requester and/or information that is gathered from a service monitor.

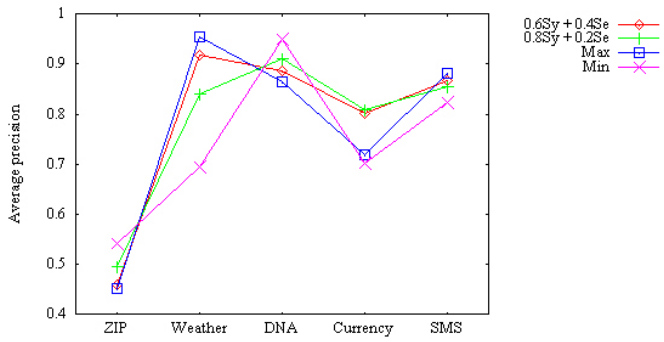


Fig. 6. Average precision of four hybrid algorithms on the first data set

We have experimented with a combination of the approaches presented in Section 3 to increase confidence in matching results and demonstrate the potential of parameterizing hybrid matching more in general. Some preliminary experiments using the same corpus as before, yielded the outcome drawn in Figure 6. Four mixed algorithms were tested: in the first two approaches different weights for structural matching were assigned, 60% and 80% correspondingly. Other two methods experimented with ranking of the retrieved services using maximum and minimum similarity scores between those assigned by the semantic and syntactic uniform algorithms. As this figure shows, for several categories hybrid approaches over-perform the purely semantic matching.

6 Conclusions and Outlook

Web service discovery plays a pivotal role in the SOC paradigm. In this paper, we have reviewed various ways of web service matching, assessing their merits

and disadvantages. In addition, we have introduced the WSDL-M2 matching algorithm that was implemented in a prototypical toolset. We have conducted a comparative analysis of WSDL-M2 with three lexical similarity measures: *tf-idf* and two WordNet-based metrics, using a uniform corpus.

To leverage WSDL matching, we have proposed a multi-dimensional composition model, having matching techniques and data models as its main constituents. The research findings that were presented in this paper are core results in nature. More research is needed in various directions. Though promising in nature, empirical evidence for hybrid matching is in need of experimentation in larger settings. We intend to conduct more experiments with hybrid matching approaches, equipping them with learning strategies.

Also, we believe that the augmentation strategy towards data model composition of matching approaches is a promising research direction. This composition model assumes that extra information may be gathered from monitoring tools. In particular, we plan to scrutinize application of the following types of knowledge:

- *Service knowledge* - knowledge about the existing services and their features, such as service documentation, interface description, ontology-based semantic extensions, service reputation and monitored information.
- *Client knowledge* - client's profile that includes his/her area of expertise, location, history of searches and previously used web services.
- *Functional knowledge* - knowledge required by the matching algorithm to map between the client needs and the services that might satisfy those needs. The chain *query* → *knowledge-based reasoning* → *response* is implied. For example, if the client asks for a currency exchange web service, and the algorithm knows that given a particular currency the client can define the country where this currency is used, it may recommend service `conversionRate(fromCurrency, toCurrency)`.

Additionally, in future work we are going to consolidate and extend our current empirical study with other IR models such as Latent Semantic Indexing. Further experiments should be conducted to evaluate the composition model of hybrid methods. Lastly, we plan to develop a case tool to support composition of- and experimentation with matching techniques.

References

1. Papazoglou, M., Georgakapoulos, G.: Introduction to the special issue about service-oriented computing. Communications of the ACM **46**(10) (2003) 24–29
2. Dan, A., Davis, D., Kearney, R.e.a.: Web services on demand: Wsla-driven automated management. IBM Systems Journal **43**(1) (2004) 136–158
3. Singh, M.: The pragmatic web. Internet Computing (2002) 4–5
4. Sajjanhar, A., Hou, J., Zhang, Y.: Algorithm for web services matching. In: Proc. of APWeb. Volume 3007 of LNCS., Springer (2004) 665–670
5. Corella, M.A., Castells, P.: Semi-automatic semantic-based web service classification. In: Proc. of the International Conference on Knowledge-Based Intelligent Information and Engineering Systems. LNAI, Springer (2006)

6. Bruno, M., Canfora, G., Penta, M.D., Scognamiglio, R.: An approach to support web service classification and annotation. In: Proc. of IEEE International Conference on e-Technology, e-Commerce and e-Service. (2005) 138–143
7. Stroulia, E., Wang, Y.: Structural and semantic matching for accessing web service similarity. International Journal of Cooperative Information Systems **14**(4) (2005) 407–437
8. Miller, G.: Wordnet: A lexical database for english. Communications of the ACM **38**(11) (1995) 39–41
9. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: Proc. of the 30th VLDB Conference. (2004) 372–383
10. Wu, Z., Wu, Z.: Similarity-based web service matchmaking. In: Proc. of the IEEE International Conference on Services Computing. Volume 1. (2005) 287–294
11. Zhuang, Z., Mitra, P., Jaiswal, A.: Corpus-based web services matchmaking. In: Workshop on Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing. (2005) 46–52
12. Van den Heuvel, W.: Aligning Modern Business Processes and Legacy Systems: A Component-based Approach. MIT Press (2006) to appear.
13. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley (1999)
14. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., Verma, K.: Web service semantics - WSDL-S. Technical note, International Business Machines Corporation and University of Georgia (2005)
15. Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., Miller, J.: Meteors wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. Journal of Information Technology and Management. Special Issue on Universal Global Integration **6**(1) (2005) 17–39
16. Syeda-Mahmood, T., Shah, G., Akkiraju, R., Ivan, A.A., Goodwin, R.: Searching service repositories by combining semantic and ontological matching. In: Third International Conference on Web Services. (2005) 13–20
17. Seco, N., Veale, T., Hayes, J.: An intrinsic information content metric for semantic similarity in wordnet. In: Proc. of the European Conference on Artificial Intelligence, IOS Press (2004) 1089–1090
18. Galil, Z.: Efficient algorithms for finding maximum matching in graphs. ACM Computing Surveys **18**(1) (1986) 23–38
19. Kokash, N.: A comparison of web service interface similarity measures. Technical Report DIT-06-025, DIT-University of Trento (2006)
20. Buckley, C., Voorhees, E.M.: Evaluating evaluation measure stability. In: Proc. of SIGIR. (2000) 33–40
21. Hollander, M., Wolfe, D.A.: Nonparametric statistical inference. John Wiley and Sons (1973)