Ontology-Based Feature Aggregation for Multi-valued Ranking

Nathalie Steinmetz[1,2] and Holger Lausen[2]

[1] Semantic Technology Institute (STI) Innsbruck, University of Innsbruck, Technikerstrasse 21, A-6020 Innsbruck, Austria
nathalie.steinmetz@sti2.at
[2] seekda GmbH, Grabenweg 68, A-6020 Innsbruck, Austria
Holger.Lausen@seekda.com

Abstract. In the last years we see a clear trend in the Computer Science area to a move towards Service-Oriented Architectures (SOAs). Research in the service domain encompasses its whole life-cycle, including topics as creation, discovery, selection, ranking and composition. This paper focuses on the ranking of discovered Web Services, proposing a novel approach based on non-functional properties of services: information that is available about services by analyzing their description that is available on the Web, their hyperlink relations, monitoring information, etc. The approach is making use of semantic technologies, aggregating the various real-world service aspects as described above in a unified model and providing different rank values based on those aspects.

1 Introduction

Within the last years we could observe a clear trend both on the Web and in Computer Science in general to a move towards Service-Oriented Architectures (SOAs). While the Web of documents as we used to know it is changing to a Web of services, the traditional software domain is altering as well: software starts being seen more and more as resource that itself is a service in a cloud (following the software model 'Software as a Service', SaaS). Using Web Service technologies all possible functionalities can be exposed and flexibly integrated in all kinds of applications (e.g., traditional software as well as Web pages). This way Web Services provide new means for interoperability of business logics.

Research in the Web Service domain encompasses the whole service life-cycle, including topics as creation, discovery, selection, ranking and composition. This paper focuses on the ranking aspect, proposing a novel approach based on non-functional properties of services: information that is available about services by analyzing their description that is available on the Web, their hyperlink relations, and similar aspects. After discovery of services, ranking is one of the most important steps to support them in selecting the most best-fitting service fulfilling their needs. Regarding the large size of publicly available services on the Web (e.g., more than 28.000 Web Services found by the seekda Web Service search

A. Dan, F. Gittler, and F. Toumani (Eds.): ICSOC/ServiceWave 2009, LNCS 6275, pp. 258–268, 2010.

engine[1] and more than 5.000 Web APIs and Mashups published on the specific Web API portal ProgrammableWeb[2]), it would be a tedious task for a human to filter out the services that are relevant to him.

Many current service ranking approaches work with the assumption that services are either semantically well-described or that we have detailed Quality of Service (QoS) information about them available (e.g., availability and response time of services). [5] addresses the ranking of services based on the semantic description of their non-functional properties, including aspects like locative, temporal, availability, obligation, price, payment, etc. This approach has the downside that the required, rather complex semantic descriptions are rather hard for non-experts to provide. [6] follows an approach for QoS-based ranking with trust and reputation management. This approach assumes that QoS properties such as availability, acceptable response time, through-put, etc. are provided by the service provider.

In contrast to these approaches, that require the active participation of the service providers to describe their services in one way or another, we base our ranking approach on the 'real-world' information that is available about them, like their descriptions, their hyperlink relations, some monitoring information, etc. We do not assume unrealistically - that we know how a service behaves on execution, what functionality he delivers, etc. (as would be the case in a man-in-the-middle approach, where we would assume to have such knowledge before doing the ranking).

Most publicly available Web Services are published using either the WSDL (Web Service Description Language)[3] standard or following a RESTful (Representational State Transfer)[1] approach. Our ranking approach supports both types of services and is targeted to work over a large number of services crawled from the Web and lightweight annotations automatically aggregated from them. [3] provides more information on the service crawling and information aggregation approach.

The rest of this paper is structured as follows: in Section 2 we describe the aggregation of features, based on ontologies, that is underlying our ranking approach. Section 3 provides our ranking algorithms, Section 4 gives a short overview of how the ranking approach is implemented and Section 5 finally concludes the paper with an overview of future work.

2 Feature Aggregation Based on Ontologies

This section describes the data aggregation for our ranking approach. We use semantic technologies to aggregate various aspects related to Web Services in a unified model, aspects that encompass information that is available about services by analyzing their description and their hyperlink relations, by talking to their

[1] http://webservices.seekda.com/
[2] http://www.programmableweb.com/
[3] http://www.w3.org/TR/wsdl

hosting server, etc. We do not rely on handcrafted, manually added, information, but only take into account real-world information that is anyway available.

As already mentioned in the introduction, the services are gathered by crawling the Web. Together with the services the Web is fostered for related documents, e.g. service descriptions, help pages, etc. The data that is resulting from the crawler comes together with RDF metadata that describes amongst others the relation from services and their related documents (in the case of WSDL services) or that tells us to what extent we believe that a certain Web resource is a Web API (in the case of RESTful services). More details on the crawl data and the corresponding RDF metadata can be found in [4]. For our multi-value ranking approach we take into account aspects like the number and the quality of related documents, classification scores of Web APIs, live monitoring data and metrics from the WSDL descriptions, as e.g. how much documentation is provided for a service. Based on our crawling experience and on our work on the seekda Web Service search engine (http://webservices.seekda.com) we see that the aspects upon which we base the ranking approach are realistically available: from more than 28.000 publicly available services approximately 20.000 contain relations to other Web resources; around one fourth of all WSDL descriptions contain some documentation on the service or operation level; all available services are monitored on a daily basis by the seekda search engine.

In the following we will first outline what RDF metadata we use for ranking; next we will describe the WSDL metrics and the monitoring data that we build upon and will in a last step provide an overview on existing and new ontologies that we use for modeling the ranking. The data described in this section is used for the Multi-valued ranking approach as described in Section 3, where we will describe in detail the calculation of the ranking.

We use the following namespaces and prefixes in the above mentioned subsections:

- Service-Finder Service Ontology - `sf:` `http://www.service-finder.eu/ontologies/ServiceOntology#`
- seekda Crawl Ontology - `sco:` `http://seekda.com/ontologies/CrawlOntology#`
- seekda Ranking Ontology - `sro:` `http://seekda.com/ontologies/RankingOntology#`
- XML Schema - `xsd:` `http://www.w3.org/2001/XMLSchema#`

Crawl Meta-data. In the case of WSDL services (and related resources) the meta-data delivered by the crawler consists mainly of annotations to the single Web documents, tying them on the one side to a service and describing on the other side of what kind the relation to the service is. As information for ranking we will use (a) the number of related documents per service, and (b) the kind of relation from document to service. The meta-data is stored using elements of the Service-Finder Service Ontology , as shown in Listing 1 as RDF triples.

```
<sf:DirectInLink > <sf:isAboutEntity> <sf:Service>
<sf:DirectInLink > <sf:belongsToDocument> <sf:Document>
<sf:DirectOutLink> <sf:isAboutEntity> <sf:Service>
<sf:DirectInLink > <sf:belongsToDocument> <sf:Document>
<sf:TermVectorSimilarityAssociation> <sf:isAboutEntity>
    <sf:Service>
<sf:TermVectorSimilarityAssociation> <sf:belongsToDocument>
    <sf:Document>
```

Listing 1. WSDL service and related documents meta-data used for ranking

In the case of Web APIs the crawl metadata describes some specific features of the Web document (e.g. number of external links, number of camel-case tokens) and provides (a) single scores that specify to what extent the two crawl classifiers (see [3]) believe that a given resource is a Web API and (b) a confidence score that is built from the single scores for convenience reasons. For ranking we will use (a) the Web API Confidence score of a document and (b) which classifier has classified the document as Web API. The meta-data is stored using elements of the Service-Finder Service Ontology and of the seekda Crawl Ontology, as shown in Listing 2 as RDF triples.

```
<sf:DocumentAnnotation> <sf:hasScore> <xsd:number>
<sf:Document> <sco:hasWebAPIConfidenceScore> <xsd:number>
<sf:Annotation> <sf:source> <sf:Agent>
<sf:Annotation> <sf:isAboutEntity> <sf:AnnotatableEntity>
<sf:DocumentAnnotation> <sf:belongsToDocument> <sf:Document>
```

Listing 2. Web API meta-data used for ranking

WSDL Metrics. A WSDL describes a Web Service from an operational point of view: services, their operations, messages, message formats, endpoints, network bindings, etc. Here the documentation of the single elements is worth being taken into account for ranking: a well documented WSDL improves the ranking of the corresponding service. We take into account the documentation of the service and of the operations. The data is stored using elements of the Service-Finder Service Ontology, as shown in Listing 3 as RDF triples.

```
<sf:Service> <sf:hasDescription> <xsd:string>
<sf:Operation> <sf:hasDescription> <xsd:string>
<sf:Service> <sf:implementsInterface> <sf:Interface>
<sf:Interface> <sf:hasOperation> <sf:Operation>
```

Listing 3. WSDL meta-data used for ranking

Monitoring Information. Interesting criteria for service ranking are related to Quality of Service information. One such information is the availability of services, i.e. their liveliness. This data is monitored and stored by seekda (Web Service search engine at http://webservices.seekda.com/) on a daily basis. The availability is based upon the endpoint of a service and is only available for WSDL services. Monitoring the liveliness of a service does not mean that the functionality of the service is tested in any kind; it expresses whether the server

where the service is hosted is reachable or not, checks at the same time whether the server is correctly implementing the SOAP protocol, whether the page needs an authentication, and more, based on the HTTP response codes.

We work with the average (percentage) availability of a service over the last 6 months, the last month and the last week (if possible). Listing 4 shows the elements that we use from the Service-Finder Service Ontology to store this data.

```
<sf:Endpoint> <sf:availabilityLast6Months> <xsd:number>
<sf:Endpoint> <sf:availabilityLastMonth> <xsd:number>
<sf:Endpoint> <sf:availabilityLastWeek> <xsd:number>
<sf:Service> <sf:hasEndpoint> <sf:Endpoint>
```

Listing 4. Monitoring meta-data used for ranking

Ranking Ontologies. To structure and store the data as described above we rely, as already mentioned, on ontologies. Where possible, we reuse the Service-Finder Service Ontology and the seekda Crawl Ontology. Furthermore we have developed a new seekda Ranking Ontology that allows us to express the new ranking specific information that is not yet expressible within the other two ontologies (shown in Listing 5).

```
<sf:Service> <sro:hasNumberOfRelatedDocuments> <xsd:number>
<sf:Service> <sro:hasRelatedDocsRank> <xsd:number>
<sf:Service> <sro:hasWebAPIScoreRank> <xsd:number>
<sf:Service> <sro:hasWSDLMetricRank> <xsd:number>
<sf:Service> <sro:hasMonitoringRank> <xsd:number>
<sf:Service> <sro:hasGlobalRank> <xsd:number>
```

Listing 5. seekda Ranking Ontology

All ranking values are expressed by numbers between 0 and 1, 1 being the best-possible rank. We calculate a rank for each of the criteria that we take into account from the crawl data and the monitoring. Then we calculate a final rank for each service. Details on how the ranks are calculated will be provided in the following section 3.

3 Multi-valued Ranking

In Section 2 we have outlined what aspects of services we aggregate to build our ontology-based feature aggregation for multi-valued ranking approach and what ontologies we use. The fact that we have all the service meta-data that we need for our ranking available as semantic data allows each semantic-aware client to build its own ranking based on the same service meta-data creating individual rules (e.g. using SPARQL). In the following we will present the way we combine the data gathered as described in Section 2 to get a global ranking value.

3.1 Rules for a Global Multi-valued Rank

The ontology-based feature aggregation for multi-valued ranking approach differs for the two types of services that we support: WSDL services and Web APIs.

For WSDL services we first calculate three independent ranking values (based on crawl meta-data, WSDL metrics and monitoring data) that are then combined to one global rank. For Web APIs we so far only take into account the Web API confidence score. The following subsections present the rules that we apply to calculate the single rank values, described in a procedural pseudo-code. It is nevertheless as well possible to describe and implement the rules in a declarative language, or, e.g., using SPARQL.

Related Documents Rank. This rank is based on the crawl meta-data that is delivered by the crawler, as shown in Listing 1 and will be calculated based on the following information and assumptions:

- How many related documents does a service have? We need to check the document annotations that belong to a service and then count the unique documents that are tied to the annotations. We improve the ranking of a service that is related to other Web resources as in that case the probability is higher that the service contains some relevant documentation, pricing information, etc.
- How is the document related to a specific service? After manual analysis of a number of services and their related documents we got to the conclusion that important information bits are mostly contained in the services' inlinks (i.e., pages that link to the WSDL document), as well as in pages that are related via term vector similarity (e.g., pages that speak about the service but are not related to it via a direct link).

In a first step we thus need to calculate the number of related documents per service. To do so it is not enough to just take the number of document annotations as one document might have several annotations (e.g. a document that has a `DirectOutLink` annotation and a `TermVectorSimilarityAssociation`). We need to first extract all annotations to get the identifiers of all documents that correspond to them. Now we count the documents, counting multiple occurrences of the same document only once. This value is then stored using the `hasNumberOfRelatedDocuments` relation of the seekda Ranking Ontology (see Section 2).

Now the related documents rank is calculated as follows (described in pseudo-code) in Listing 6:

```
// get the average number of related docs per service
average = totalNumberOfRelatedDocs / numberOfServices;
// get the root mean square deviation of the distribution of
// related docs per service
sumDeviationFromAverage = 0;
for (Service s : allServices) {
    sumDeviationFromAverage += s.numberOfRelatedDocs   average;
}
variance = power(sumDeviationFromAverage, 2) /
        numberOfServices -1;
rootMeanSquareDeviation = positiveSquareRoot(variance);
// get max outliers values
maxOutlier = average + (2.5 * rootMeanSquareDeviation);
```

```
// take into account the kind of relation from document to
// service. If a service has a number of related documents
// that is outside of the max outlier value we set the number
// to the average of related documents per service in order
// to not allow spam to influence the ranking value
for (Service s : allServices) {
    temporaryRank = 0;
    if (s.numberOfRelatedDocs > maxOutlier) {
        s.numberOfRelatedDocs = maxOutlier;}
    if (s.hasInlink) {
        temporaryRank = s.numberOfRelatedDocs * 5;}
    if (s.hasTermVectorAssociatedDoc) {
        temporaryRank += s.numberOfRelatedDocs * 4;}
    if (s.hasOutlink) {
        temporaryRank += s.numberOfRelatedDocs * 2;}
    s.finalRank = temporaryRank / maxOutlier / 11;
}
```

Listing 6. Calculation of the Related Documents Rank

The single values that are used for the single kinds of related documents to calculate the temporary rank are currently experimental. These might be changed on a frequent basis until we discover the values that seem optimal for our needs. The final rank is stored for each service using the hasRelatedDocsRank relation of the seekda Ranking Ontology.

WSDL Metrics Rank. This rank is based on metrics that we extract from the WSDL descriptions. We currently take into account the documentation of (a) the service element, and (b) the operations. As we mentioned already in Section 2, approximately a fourth of all service descriptions contain some documentation on the service or operation level. The rank is calculated as follows in Listing 7:

```
for (Service s : allServices) {
    finalRank = 0;
    if (s.hasServiceDocumentation) {s.finalRank = 1;}
    if (s.hasOperationDocumentation) {s.finalRank += 3;}
    s.finalRank = finalRank/4;
}
```

Listing 7. Calculation of the WSDL Metrics Rank

We put more importance on the documentation of the single operations than of the service documentation, as we think that the operation might contain useful information regarding the functionality provided by the operation and regarding its invocation. We currently do not differentiate between whether all operations of a service are documented or only one or some. The final rank is stored for each service using the hasWSDLMetricRank relation of the seekda Ranking Ontology.

Monitoring Rank. This rank is based on the liveliness information of a service, e.g., is the server reachable, does it correctly implement the SOAP protocol, etc. This liveliness information is delivered by seekda on a weekly basis as shown in Listing 4. The availability score is a number between 0 and 1 that is set depending on the endpoint check result. The score is, e.g., 0 for read time-outs or errors and 1 if, based on the resulting payload (e.g., XML fault), we are rather

sure to be talking to a WSDL over SOAP. In-between different scores are set to express pages that are not found, pages that require a login or an authentication, etc., mostly based on the HTTP response code.

We get the average service availability score for different time periods: last week, last month and last 6 months. We assume that the long-time availability of a service is more relevant than only the short-time availability over one week. It is important to note that this rank does not state anything about whether the functionality that the service announces is correctly implemented or not. The rank is calculated as follows in Listing 8 and is stored for each service using the hasMonitoringRank relation of the seekda Ranking Ontology:

```
for (Service s : allServices) {
    finalRank = ((s.availabilityLastWeek * 1.5) +
            (s.availabilityLastMonth * 2.5) +
            (s.availabilityLast6Months * 6)) / 10;
}
```

Listing 8. Calculation of the Monitoring Rank

Web API Rank. For ranking Web APIs we currently only take into account the Web API confidence score. This score is calculated based on two classifiers within the crawler that check whether a Web resource might be a Web API or not. The rank is based on the crawl meta-data that is delivered by the crawler, as shown in Listing 1 and will be calculated based on the following information and assumptions:

- What is the Web API Confidence score of a document? This score is a final confidence score that is calculated from single scores provided by two Web API classifiers.
- Which crawler classifier has classified the document as Web API? As described in [3], one automatic classifier (SVM Classifier) has been trained on a set of data, while the other classifier (Web API Evaluator) performs structural and term vector analyses of the resources and assigns scores for specific indicators.

To calculate the rank we thus need to extract both the score and the component that has assigned the score. Based on first evaluations of the classifiers, we deem the score of the SVM classifier more important than the one of the Web API Evaluator. Listing 9 shows how the rank is calculated:

```
for (Service s: allRESTServices) {
    finalRank = 0;
    if (s.hasSVMClassifierAnnotation) {
        finalRank = s.hasWebAPIConfidenceScore * 3;}
    if (s.hasWebAPIEvaluatorAnnotation) {
        finalRank += s.hasWebAPIConfidenceScore * 1;}
    s.finalRank = finalRank/4;
}
```

Listing 9. Calculation of the Web API Rank

Global Rank. As already mentioned above, the calculation of the global rank differs depending on whether the ranked service is a WSDL-based service or a Web API. For WSDL services we calculate the global rank based on the Related Documents Rank, the WSDL Metrics Rank and the Monitoring Rank. The single ranks are numbers between 0 and 1, and from these we calculate the global rank as follows in Listing 10, putting equal relevance on the availability of documentation (related documents being estimated more important than the documentation within the WSDL) and on the liveliness of a service. This way services that are on the side reliable and on the other side (assumably) well documented are ranked best. The global rank is stored for each service using the hasGlobalRank relation of the seekda Ranking Ontology.

```
for (Service s : allServices) {
   s.globalRank = (s.hasRelatedDocsRank * 0.35) +
         (s.hasWSDLMetricRank * 0.15) +
         (s.hasMonitoringRank * 0.5);
}
```

Listing 10. Global Rank Calculation for WSDL-based services

For Web APIs, we currently only dispose of one kind of rank: the WebAPI rank, that is thus at the same time the global rank of the service.

4 Evaluation

The ranking approach that we present in this paper is especially valuable in service search in an open, large scale environment, i.e. search over a large number of services whose degree of documentation is unknown in advance and where no agreements with the service providers have been made beforehand concerning the service documentation. The rank does not take into account specific user requirements, as the features we include in our rank calculation are of such nature that we assume they are always relevant for users (e.g., we assume that it is always important for a user to know whether a service is available or not) and are thus adequate for a generic ranking. Although our final global rank does not take into account specific user requirements, this rank can be easily modified and composed in a different way by any RDF aware client as we return all single rank values as simple RDF triples.

As our ranking approach is still very new, especially concerning the inclusion of related Web resources for WSDL-based services and Web APIs in general, we cannot yet provide a fully-fledged evaluation for it. Parts of the ranking approach have though been evaluated in the scope of the seekda Web Service search engine: the WSDL-based services are ranked as presented in Section 3, basing upon the WSDL metrics and the monitoring information as described in the sections 2 and 2. The search engine ranking has been evaluated by experts in the Web Service domain; they have manually assessed the relevance of the discovered services with different ranking aspects (such as availability or documentation) taken into account and different combinations of the weight of these aspects being applied.

Providing a complete evaluation of our ranking approach is part of our future work. We plan to conduct an empirical evaluation, employing techniques as, e.g., user interviews and questionnaires to assess the importance and relevance of our ranking value(s). Also we would like to evaluate our approach with regard to other ranking approaches (such as provided in [2], [6] or [7]).

5 Conclusion and Future Work

In the previous sections we described our approach for multi-valued ranking of Web Services - both WSDL-based and Web APIs - based on ontology-based feature aggregation. We outlined what real-world data our ranking values rely upon and provided an overview on how the different rank values are calculated.

The approach we follow with our multi-valued ranking in general and the export of the ranking related data in a semantic format has major advantages (and novelties). These include (a) the fact that each RDF aware client can understand the rational of a ranking, i.e. it can work with the final ranks, but can as well analyze the data on which the ranking is based. Also it can perform, if desired, its own ranking calculation with the service meta-data that is provided. Another innovative aspect is (b) the fact that our ranking approach covers and combines many aspects of a service, like its documentation, the existence of related information that is available on the Web, the liveliness data (which is a strong QoS criteria), and, in the case of Web APIs ranking, the classification score of services.

As future work we see first the evaluation of our ranking approach and second the further enhancement of it. We plan, e.g., to enlarge the number of criteria that we take into account for the ranking. We expect that especially for the Web APIs we will take into account more metrics; we will base our improvements on evaluations of the current approach. Also we can image to actively taking user feedback and requests into account, i.e. providing the user with a possibility to select what features are most important to him.

Acknowledgements

The work is funded by the European Commission under the FP7 project SOA4All and by the FFG (Österreichische Forschungsförderungsgesellschaft mbH) under the FIT-IT project Service-Detective.

References

1. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine (2000)
2. Pajntar, B., Grobelnik, M.: Searchpoint - a new paradigm of web search. In: 17th World Wide Web (WWW) Conference - Developers Track (2008)

3. Steinmetz, N., Lausen, H., Brunner, M.: Web service search on large scale. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 437–444. Springer, Heidelberg (2009)
4. Steinmetz, N., Lausen, H., Brunner, M., Martinez, I., Simov, A.: D5.1.3 - second crawling prototype (2009)
5. Toma, I., Roman, D., Fensel, D., Sapkota, B., Gomez, J.M.: A multi-criteria service ranking approach based on non-functional properties rules evaluation. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 435–441. Springer, Heidelberg (2007)
6. Vu, L.-H., Hauswirth, M., Aberer, K.: QoS-based Service Selection and Ranking with Trust and Reputation Management. Technical report, EPFL (2005)
7. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)