Evaluation of Semi-Automatic Acquisition of Semantic Descriptions of Web Services

Shahab Mokarizadeh$^1$, Peep Küngas$^2$, Mihhail Matskin$^{1,3}$

$^1$ Royal Institute of Technology (KTH), Stockholm, Sweden
$^2$ University of Tartu (UT), Tartu, Estonia
$^3$ Norwegian University of Science and Technology (NTNU), Trondheim, Norway

$^1$shahabm@kth.se, $^2$peep.kungas@ut.ee, $^3$misha@kth.se

Abstract—This paper introduces and evaluates a novel Web service interface annotation and matching scheme designed for processing large sets of Web service interface descriptions. The matching scheme relies on a set of rules, which exploit the structural relationships encoded in an automatically learned ontology to discover matching interface elements. The proposed scheme is evaluated on a Web services interface corpus consisting of WSDL descriptions. The experimental results show that the proposed matching scheme can be used for bootstrapping large-scale Web service interface annotation.

Keywords—Web service annotation; schema matching; XSD; WSDL

I. INTRODUCTION

Lack of good benchmarks has hindered proper evaluation of proposed Web service discovery and composition solutions in practical settings where many Web services are available. To overcome this shortage, there has been recently some activities [3][4][5] in the field for tackling the analysis of essential properties of web services and their networks with the aim to provide feedback to web services discovery and composition problems from the search space topology perspective. Since semantic annotations enable construction of dataflow- and workflow-based networks, which are needed for analysis, there is a need for a fully automatic cost-effective web service annotation mechanism. Previous works in web service analysis suffer from following drawbacks: a) due to lack of semantic annotations in the vast majority of existing web services only pure syntactic matching has been applied [3]; b) only small sets of semantically annotated web services have been examined [5] due to the high costs related to labour-intensive manual annotation of web services [1]; and c) they rely on the assumption that a supporting reference ontology is provided [4].

In this paper, we report work in progress in developing and evaluating a semi-automatic cost-effective semantic annotation approach, which combines a previously proposed ontology learning method [6] and a cost-effective semantic annotation method [1], for bootstrapping analysis of large repositories of web services, which do not include semantic annotations. The ultimate goal of the reported work is to provide means for bootstrapping large scale annotation of Web services to enable further advances in the field relying on these annotations.

We evaluate the approach both qualitatively and quantitatively on our corpus of Web service interface descriptions in two stages. First we manually create a so called golden ontology and corresponding annotation heuristics, which will be used then for automated annotation as suggested in [1]. Then we automatically learn a reference ontology and corresponding annotation heuristics by using the scheme proposed in this paper for the same corpus as in the preceding stage and assess the quality of completely automatically generated annotations with respect to manually created ones. The evaluation confirms that the proposed matching scheme can be used for bootstrapping annotation of large scale of Web service interfaces in a semi-automatic way with some certain error margins.

The rest of this paper is organized as follows. In Section 2 we outline our Web service annotation and matching scheme. In Section 3, we discuss our approach to evaluate the quality and quantity of applied annotations. Experimental results of applying the annotations and matching scheme are presented in Section 4. Finally, Section 5 reviews related work, while conclusions and future work are presented in Section 6.

II. WEB SERVICE ANNOTATION AND MATCHING SCHEME

Different Web service annotation and matching schemes have been adopted to deal with integration of heterogeneous information sources. We employ our ontology learning approach [6] to first generate a reference ontology from our collection [1] of Web service corpus (ca 15000 WSDL documents collected from various repositories in the Web) and then utilize the generated reference ontology to annotate the services. In the reference ontology, instances are referring to the terms while classes refer to conceptual representation of the underlying terms. In this work, term refers to an XML schema basic element name or a message part name in the

---

1Available online at : http://www.soatrader.com/web-services
web WSDL document and our goal is to annotate the identified terms. The extracted terms from the WSDL documents in our collection builds up the dataset which is considered for ontology learning, annotation and matching. Our ontology learning mechanism [6] exploits frequently observed naming patterns in the given dataset and relies on morphological analysis to identify concepts and relations between and generate an ontology accordingly. In the generated ontology, instances refer to the terms and the conceptual classes are interrelated further using ontological properties namely: hasProperty, isSynonymOf and isSimilarTo. While isSynonymOf property conveys that the concepts on the both side of relation are lexically synonyms, the isSimilarTo relation expresses a weaker degree of lexical similarity between two concepts.

We employ the heuristic-based annotation mechanism reported by Küngas and Dumas [1] to annotate Web services. In this mechanism annotation heuristics are represented as rules in the following form: entity reference $\leftrightarrow$ synset (e.g. Password $\leftrightarrow$ (password, pwd, strPassword, authpassword, pass)). The meaning of such a rule is that an XML schema element matched by any element in the synset is mapped to the entity reference (in our case a concept identifier in the automatically constructed ontology) on the left-hand side of the rule. A synset, or a synonym ring, is a group of labels (i.e. terms) that are considered semantically equivalent. We construct synsets from the labels of particular instances in an ontology. Thus according to the heuristic rule example – if Password is a concept identifier, then password, pwd, strPassword, authpassword and pass are labels of instances in the generated reference ontology (i.e. terms).

By utilizing the generated ontology and annotating respective Web service elements, we promote the process of correlating Web service inputs and outputs from pure syntactic level to ontological instance matching level. The annotation process simply annotates those extracted schema element names/ WSDL message part names (terms) with their respective concepts in the generated ontology. In order to match inputs and outputs of Web services through annotated elements, we rely on a set of matching rules. Here matching refers to the process of finding relationship or correspondence between instances (terms) in an ontology through utilization of any of following rules adopted from rules proposed by[11][12]. By definition two instances are matched if and only if one of the following matching rules is true:

**Rule-1:** They both belong to a same concept (e.g. \{loc, location\} isInstanceOf Location).

**Rule-2:** They belong to lexically synonym or similar concepts (e.g. \{loc isInstanceOf Location\}, and \{place isInstanceOf Place\} where Place isSynonymOf Location).

**Rule-3:** One of the instances belongs to a concept which subsumes the concept representing the second instance (e.g. pair of \{ContractId , Id\} where \{ContractId isInstanceOf ContractIdentifier\} and \{ContractIdentifier isSubClassOf Identifier\}) and \{id isInstanceOf Identifier\}

**Rule-4:** One of the instances belongs to a synonym or similar concept which subsumes the concept representing the second instance (e.g. pair of \{bidUId , Id\} where \{bidUId isInstanceOf BidUniqueCode\}, \{ BidUniqueCode isSynonymOf ContractIdentifier\})

**Rule-5:** The instances belong to two concepts inter-related by other ontological properties (e.g. Address hasProperty Address_PhoneCode).

### III. Evaluation Approach

In our evaluation, we initially employ our annotation scheme to annotate certain elements (terms) from Web service corpus by generating reference ontology. Next, we evaluate quality and quantity of matching cases discovered using our introduced matching rules operating on the reference ontology.

Since automatic Web service matching is the target use-case for annotated Web services, we measure the quality of pair-wise matches between annotated XML schema element/ message part names. In order to be able to verify the quality of the generated reference ontology and subsequent annotations, initially we limit evaluation dataset to 2000 most frequent terms extracted from our collection of WSDL documents. Using this dataset we create two independent ontology resources. While the first ontology is handcrafted by a human expert (i.e. an ontology engineer), the second one is constructed automatically using our ontology learning mechanism, explained at [6]. We refer to the former case as Golden ontology while the latter one is called the Generated ontology. We acknowledge that Golden ontology might suffer from bias introduced by the human expert due to the lack of documentation in the underlying Web services. Next, we align the Generated ontology with Golden one, using Falcon-AO ontology matching tool [8] and harvest only aligned concepts and their underlying instances. We refer to this set of instances as aligned instances and they account for 968 cases [6]. The evaluation goal is to determine how many true matching cases between every two aligned instances we will gain, provided that the precision and recall of the aligned concepts are ideal (however in practice there exist some limitations and error margin due to the exploited tool [8]).

We adopt Euzenat and Shaiviko’s[7] terminology to describe our instance matching process. The result of instance matching process is a set of correspondence elements. Each correspondence element implies that a relation holds, according to a particular matching rule, between two instances in an ontology. A correspondence element $Ont_{ci,j}$ is a triple $<a_i, b_j, R>$ where $i\neq j$; $i,j=1...N$; $N$ is the number of instances; $a_i, b_j$ refer to i-th and j-th instance in the ontology referenced by $Ont_i$; $k$ is the identifier of the ontology, and finally $R$ specifies the matching rule that reveals kind of semantic relationship holding between two instance $a_i$ and $b_j$. If two instances are not matched, then we use notation of NM (NotMatched) instead of the matching rule. For evaluation purposes, we compare the matching rules $R$ and $R'$ in $Ont_{GenC_{i,j}}=<a_i, b_j, R>$ and $Ont_{goldC_{i,j}}=<a_i, b_j, R'>$ where $Ont_{GenC_{i,j}}$ denotes the correspondence element obtained in the Generated ontology ($Ont_{Gen}$) while $Ont_{goldC_{i,j}}$ refers to the
computed correspondence element for the same pair of instances \(a_i\) and \(b_j\) in Golden ontology (\(Ont_{Gold}\)).

IV. Evaluation Results

In this section, we present the experiments\(^2\) made using the top 2000 recurrent terms to generate reference ontology, annotate Web service elements and perform Web service element (i.e. instances in the reference ontology) matching. Automatic instance matching between aligned instances (968 items) using Golden and Generated ontology results in 2362 and 3797 correspondence elements respectively. We perform evaluation based on comparison between \(R\) to \(R'\) in \(OntGoldC'_{ij}=\langle a_i, b_j, R'\rangle\) and \(OntGenC_{ij}=\langle a_d, b_j, R\rangle\) for all correspondence elements, as pointed out in Section 3. The comparison results are grouped into three groups based on the exploited matching rules:

**Matched in both** \((R\neq NM & R'\neq NM)\): This category embodies the correspondence elements which are matched in both ontologies and it covers 2837 cases (75% of those discovered by Golden ontology) and majority of them (2279 cases) are resulted by Rule-1. Our observation reveals that these identified correct correspondence elements belong to those instances with clear lexical semantic, and to some extent having context independent semantic or those conveying concrete concepts, for example birthday, Social Security Number, ISBN, authCode, pwd, etc.

**Missing matches** \((R=NM \& R \neq R')\): This group consists of instances that are matched by Golden ontology but not by Generated ontology and it accounts for 952 cases (25% of those discovered by Golden ontology). Besides to typical WordNet \([15]\) limitations \([10]\) one major reason is the fact that concepts in Golden ontology are more inter-related (well-organized through handcrafted taxonomical relationships) than those in Generated ontology. This leads to loss of possible matches between instances by the ground of subsumption relation between representing concepts (i.e. by Rule-3). This flaw is a direct result of inferring taxonomical relation solely relying on linguistic synthesis. For example, terms “Caller” (as person who makes phone call) and “Person” cannot be correlated, because they are neither synonym according to dictionary nor appeared together in a compound noun, whereas in Golden ontology they are assigned to two subsuming concepts. This situation can be alleviated by introducing a new module in our ontology organization step where additional domain resources such as domain ontologies, and taxonomies are incorporated to reorganize hierarchy of generated concepts.

**Extra introduced matches** \((R'=NM \& R \neq R')\): This is the group of correspondence elements which are only discovered using Generated ontology but not Golden one and it consists of 1980 cases. Introduction of these extra matches in Generated ontology is due to following reasons: 1) Golden ontology lacks any kind of property relations (such as isSymonymOf, hasProperty, etc); hence it cannot correlate instances by ground of Rules 2,4, and 5. The matching cases resulted by Rule-5 are intuitively reflecting PartOf or Having relationships between matching instances. 2) in few cases there exist kind of lexical similarity between concept labels , which is not covered by Golden ontology for example for the cases when two instances convey a similar semantic but in a different context (e.g. \{taskName, jobName\}). In both cases we need to assign a confidence degree expressing trustworthiness of the matching (how close is the matching to ideal case based on contextual information). In this work, we rely on human assessment to distinguish correct cases (true positives) from incorrect ones (false positives). It should be noted the accuracy of these introduced matchings should be considered as lower quality matchings compared to those identified in previous group (i.e. “matched in both”).

Fig. 1 shows the assessment results over introduced correspondence elements categorized based on the exploited matching rule. Accordingly, the incorrect figures refer to quantity of cases that two instances are matched while they do not have same/related semantic in practice, while correct cases point to semantically meaningful matched pairs. Accordingly, the quantity of incorrect matches resulted from Rule-4 is the highest among all (89%), while Rule-2 and Rule-5 are equally showing the least amount of incorrect matches (around 40%) and Rule-3 lies almost in the border, with 51% incorrect cases. Our observation over cases in group of Rule-3 shows that occurrence of false positive cases are mostly resulted from generic concepts. For example concept Code , in Generated ontology, is so generic that can subsumes any of CurrencyCode, LicenseStatusCode, and, DesignCode concepts while in Golden ontology, these concepts are more accurately classified by looking into their precise semantic of the term in the Web service corpus. For instance CurrencyCode refers to the currency abbreviation rather than a digital code like that of LicenseStatusCode. This deficiency can be correlated to limitation of using lexical resources as they do not cover with the same detail different domains of knowledge and also many of domain independent terms, as reported by Bergamashi and Sorrentino \([2]\).

On the other hand, it can be seen in Fig. 1 that subsumption (Rule-3) is reasonably convincing for most of the specialized (concrete) concepts, which in our case are mainly compound nouns. For example, StartDate can subsume any of futureLaunchDate, flightDepartureDate terms. However, without further investigation we cannot provide any firm theory supporting the aforementioned patterns at this point. The high number of false positives of Rule-4 are due to two

\(^2\) Available at: http://www.isk.kth.se/~shahabm/AnnotationAnalysis

---

Fig. 1 Quantity of “extra introduced matches” discovered by matching Rules 2-5 using Generated ontology
Research in knowledge acquisition methods for the purpose of (semi) automatic annotation of Web service interfaces can be roughly divided into the following categories: 1) methods which utilize linguistic resources and NLP techniques; 2) methods which rely on machine learning techniques to classify or cluster similar (or related) names and generalize them to their ontological concepts; 3) the approaches that combine the aforementioned methods with online resources.

The works in the first category, exemplified by [11][12], capture relationships between WSDL elements and transform them into ontological concepts and relationships, typically using simple lexico-syntactic patterns and WordNet dictionary. They exploit textual descriptions of Web services to improve or enrich the quality of the generated ontology.

However, the machine learning techniques, belonging to the second category, exploit NLP techniques only to normalize their input datasets and then continue with machine learning techniques. In this light, Heß et al. [13] developed a classifier system which initially needs to be trained in order to generalize (semantic of training data) and predict semantic labels for (similar) unseen Web services. As we target a large repository of absolutely non-annotated ad-hoc Web services from different domains, applicability of such techniques is not clear. In contrast the clustering methods such as the one proposed by Dong et al. [14], which relies on the co-occurrence of parameter names as an heuristic for identifying ontological concepts, are more appealing since they do not require any training dataset.

Finally, the approaches in the third category exploit Web resources (such as search engines) as sources for knowledge acquisition and augment the Web results with machine learning and/or NLP techniques. In this direction, Segev and Sheng [9] combined TF/IDF measures with Web search results to discover proper domain concepts representing WSDL elements and then validated it using textual documentations in WSDL documents. The main obstacle of utilization of some of the aforementioned work, namely [9][11][12], is the fact that around 94% of WSDL documents in our collection lack any textual documentation [6], hence, utilization of those approaches is not feasible in our case.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we proposed and evaluated a scheme suitable for automated annotation of a large set of Web service corpus. The experimental results revealed that utilization of certain rules of the proposed matching scheme can be used for bootstrapping large-scale annotation of Web service interfaces in a semi-automatic way. However, we still need to enhance the quality of generated ontologies by potentially incorporating external resources such as domain ontologies or Web resources to the annotation scheme. Since the number of available Web services and therefore our corpus is expected to increase, keeping the generated reference ontology updated will be challenging. Hence, it is not cost-effective to incorporate the entire dataset for ontology learning purpose. As part of our future work, we are aiming to discover and annotate only a subset of whole dataset which its network properties exhibit closer approximation to the already observed properties in the Web service networks [3][4][5].

ACKNOWLEDGMENT

This research is partly funded by ERDF via the Software Technology and Applications Competence Centre (STACC) and ESF via the DoRa program.

REFERENCES