Improving REST Service Discovery with Unsupervised Learning Techniques

Juan Manuel Rodriguez*‡, Alejandro Zunino*, Cristian Mateos*, Felix Oscar Segura† and Emmanuel Rodriguez†
*ISISTAN - Reaserch Institude
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN)
Tandil, Argentina
† Facultad de Ciencias Exactas
UNICEN
Tandil, Argentina
‡e-mail: juanmanuel.rodriguez@isistan.unicen.edu.ar

***Abstract*—Discovery and replacement are two of the main promises of Service Oriented Computing. There has been much research on this topic for traditional SOAP-based Web Services. Although the original propose for REST service lacks of this feature, some researchers have study how to perform discovery for REST services using both IR based techniques and semantic techniques. This work presents a novel IR-based discovery approach for REST services described via WADL files. Our approach takes advantage of unsupervised machine learning techniques for improving discovering results. This approach relies on clustering algorithms, such as K-means or X-means, to reduce the search space for a given query. The experimental results shows that using an appropriated a clustering technique, our approach reported nearly 4 times higher F-measure than a traditional IR-based search engine, namely Apache Lucene. Additionally, the experiments report other metrics, such as Recall, Precision, Precision at-10 and Recall at-10, that also point out that the proposed approach outperforms Lucene. Finally, another important contribution is a set of queries and WADL files gathered from Internet that can be used for evaluating future discovery proposals.**

***Index Terms*—REST; Service Discovery; Information Retrieval; WADL**

I. Introduction

Representational State Transfer (REST) is a service paradigm that was firstly proposed by Roy Fielding in 2000 [1]. REST is an architectural style for creating scalable Web Services, called REST services. REST services are based on entities, which are known as resources, and can be identified by a URI. REST APIs defined Create, Read, Update and Delete (CRUD)-like operations over the entities. These operations are mapped to the well known HTTP request methods (POST, GET, PUT and DELETE).

Besides providing a simple four-operation API, REST has been proved to be interoperable [1] because it is based on well-known Internet protocols. Furthermore, REST network requirements tend to be significantly lower than SOAP-based Web Service requirements, which a desirable characteristic for many systems, particularly for systems running in mobile devices [2].

In traditional Service Oriented Computing (SOC), discovering service is a key feature. This is because it enables service providers to easily advertise their services, while service consumers can find the services that they need [3]. In SOAP-based Web Services, this was originally provided by the Universal Description, Discovery and Integration (UDDI) standard. Although UDDI has not been widely accepted, the importance of service discovery has motivated a wide range of research [4], [5]. Because of its importance, discovery has been also a research topic in REST services [6], [7], [8]. Like in SOAP-based Web Services, there are two main approaches for REST discovery: information retrieval (IR)-based [8] and semantic based [6].

In this paper, we present a novel IR-based discovery approach for REST services. Our approach automatically indexes REST services described by means of their Web Application Description Language (WADL[1]) files. WADL is an XML-based language for describing resources and the available operations for these resources. In addition to traditional IR techniques, our approach uses clustering techniques for reducing the search space and incrementing Recall and Precision in order to achieve high effectiveness.

The rest of the paper is organized as follows. Section II discusses previous works in service discovery and particularly in REST discovery. Section III outlines the proposed approach. Then, Section IV presents the evaluation of the approach. Finally, Section V concludes the paper and introduces future research topics.

II. Background

Regarding describing REST services, there is no standard, but several alternatives have been proposed by both industrial and academical entities. hRESTS [9] was proposed in 2008 as an extension to HTML for describing REST services. It allows adding a machine readable description of the REST service to the page that describes it. Another proposal for representing REST services is WADL; WADL was submitted

[1]WADL Submission to W3C: http://www.w3.org/Submission/wadl/

Listing 1. WASL example

```
<application ...>
  <grammars>
     <include
       href="dataTypes.xsd"/>
     ...
  </grammars>
  <resources base="http://.../V1/">
     <resource path="resource1">
       <doc title="Resource␣1">
         This is documentation....
       </doc>
        <method name="GET" id="id1">
         <request>
           <param name="param1"
             type="xsd:string"
             style="query"/>
           ...
         </request>
         <response status="200">
           <representation
             mediaType="application/xml"
             element="yn:dataType"/>
         </response>
          ....
      </method>
       ...
     </resource>
     ...
   </resources>
</application>
```

by Sun Microsystems to the W3C in 2009. One of the main differences between hRESTS and WADL is that the later does not rely on HTML. In WADL, REST services are represented as resources, which are identified by an URI, the methods that these resources supports and its representations. Listing 1 provides an example of WADL structure. Regarding semantic descriptions, SA-REST[2], similarly to hRESTS, allows adding information to the Web pages that describe REST services. Unlike hRESTS, SA-REST is intended to add semantic information, which can be used for several purposes, including service discovery. Finally, WSMO-lite [10] is a lightweight ontology for semantic Web Services. One of the main features of WSMO-lite is that provides support for both SOAP-based Web Services and REST services. WSMO-lite is based on an SAWSDL for SOAP-based Web Services and MicroWSMO, which is an extension of hRESTS, for REST services.

Discovery in SOAP-based Web Services was originally materialized by means of UDDI. Although UDDI was not widely adopted, there are several works [11], [4], [5] that aim at Web Service discovery. Basically, there are two main approaches for service discovery: based on information retrieval or based on semantics (ontologies). The former only requires the standard description for the Web Service by means of the Web Service Description Language (WSDL), while the later also requires service developers to add semantic information in the description.

Semantic Web Services and, in particular, Semantic Web Service registries, have been criticized for several reasons [4]. Firstly, ontologies, even two ontologies for the same domain, are not compatible among them; therefore, they need to be *mapped* to make them compatible [12]. Secondly, processing ontologies might have a very high computational complexity; even more, some ontologies might be non-computable at all, such as the ones defined by means of OWL-Full[3]. Finally, since SOAP-based service developers, even in large enterprises [13], tend not to care about textual comments and names in their WSDL documents [14], it is improbable that they would correctly annotate services with complex ontological information.

Although researchers have proposed several approaches for implementing SOAP-based Web Service registries, REST service registries present new challenges. The first problem is that there is no uniform manner of describing REST services. Despite the existence of specific language, such as WADL, for describing REST services, many developers describe their services in ad-hoc Web Sites [15], [8]. This is true even for large Internet based companies, like Google[4] or Facebook[5]. Considering this, Liu et al. [8] have developed a neuronal network based classifier for determining whether a Web site describes a REST service. For training this classifier, the authors extract very particular features of the Web pages, such as if they contain keywords like API, REST, or WADL. Using this classifier, the authors propose an automatic crawler for a REST service registry. The final REST service registry is implemented using Apache Lucene[6], a well-know framework for implementing search engines. Notice that the main contribution of [8] is not the search engine, but the REST service identification process for automatic crawling, e.g., identifying Web pages that describe REST services.

In [7], the authors present a framework for describing and discovering REST services through semantic information. Like in SA-REST, this work proposes to describe REST services via Web pages annotated with semantic information using RDF. The proposed representation focuses on resources and their relationships. There is a particular kind of relation that is a link between a service and its consumer. These relations are used to discover and link services that are relevant for a particular service consumer based on services consumed by the same kind of consumers. Although this work presents a complete framework for describing and discovering services, there is not a comprehensive evaluation of this framework, but a case study.

There are works that aim at improving Web Service registry performance through machine learning techniques [16], [17]. Web Service Query By Example (WSQBE) [16] is a Web Service registry that uses Java interfaces source code to search SOAP-based Web Services with a similar structure. WSQBE uses the Rocchio classification technique for reducing the

[2]SA-REST Submission to W3C: http://www.w3.org/Submission/SA-REST/

[3]OWL sub-language: http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3

[4]Google Map API: https://developers.google.com/maps/documentation/webservices/

[5]Facebook Graph API: https://developers.facebook.com/docs/graph-api

[6]Apache Lucene: http://lucene.apache.org/

search space. In [17], the authors propose a SOAP-based Web Service registry that uses clustering for reducing the search space, while improving the results quality. Experimental results show that the proposed methodology outperforms previously presented Web Service registries. One of the main advantages of using clustering over classification is that the former does not require a manually labeled data-set to be trained.

III. CE-RSR: Cluster Enhance REST Service Registry

In this work, we propose an IR-based REST service registry, called Cluster Enhanced REST Service Registry (CE-RSR), that take advantage of clustering technique for improving its performance. The proposed registry indexes REST services using their description in WADL. CE-RSR uses a six-step process for indexing each REST service. This process is depicted in Figure 1 . These steps are further described below:

1) **Extracting Terms**: this step consists in processing the WADL files in order to obtain all the relevant terms. Firstly, a set of strings is retrieved from the WADL. Since these string might be composed by more than one term, they are split assuming cammel case convention, i.e., compound names are written such that each word begins with a capital letter. For example, "compoundName" would be split into "compound" and "name". The strings are extracted from the following properties and tags of the WADL files:
    a) Value of the `path` property for each `resource` tag. Although this string is part of the resource URI, it is usually a human friendly text.
    b) Value of the `name` property for each `param` tag.
    c) Value of the `id` property for each `method` tag.
    d) Value of the `name` property for each `doc` tag.
    e) The content of each `doc` tag, which is the documentation for the WADL file.
2) **Removing Stop Words**: this is a common preprocessing step in text mining that consists in removing common words that do not provide relevant information. Examples of these words for English are "a", "the", "with" and "to".
3) **Porter Stemming**: this step consists in reducing words to their morphological root. For example, related words, such as computing, computer and compute, are reduced to the same string, e.g., "comput". For this step we use the well-known Porter stemming algorithm [18].
4) **Calculating TF-IDF**: this step consists in calculating the vectors that represent the WADL files using the well-known TF-IDF technique for weighting the terms. TF stands for Term Frequency and represents the importance of a term in a document. IDF stands for inverse document frequency and represents how infrequent is the term in the document corpus. Formally, TF-IDF is defined as:
    $TF-IDF(term, Doc, Corpus) = TF(term, Doc) * IDF(term, Corpus)$

    $TF(term, Doc) = \frac{\#occurencies(term, Doc)}{|Doc|}$

    $IDF(term, Corpus) = log\frac{|Corpus|}{|\{d\varepsilon Corpus : term\varepsilon d\}|}$
5) **Clustering**: this step consists in creating groups of REST services according to their WADL files similarity. CE-RSR supports different clustering techniques. Currently, CE-RSR supports the following clustering technique: K-means, Bisecting K-means (BK-means), X-means [19], Hierarchical and Expectation Maximization (EM). All these technique implementations, but Bisecting K-means, are provided by Weka 3.6.10 [20]. Bisecting K-means was implemented using Weka K-means implementation.
6) **Indexing Cluster N**: this step is repeated for each cluster resulting from the previous step. It involves creating an index for each cluster, i.e., each cluster would have its own index for the WADL files in that cluster. Notice that, depending on the selected clustering technique, the number of cluster, therefore the number of indexes, might or might not be known a-priori. For creating the indexes, we have relied on the Lucene framework.

The querying process of CE-RSR follows a similar six-step process to the one used for indexing the documents. Figure 2 outlines the querying process. The first step, namely Extracting Terms, is an optional step that depends on the query format. CE-RSR accepts queries in form of a set of strings or, as WSQBE [16], in form of a Java class. In the later, the Extracting Terms step analyses the Java source code for extracting names and comments. Then, steps 2 through 4 are exactly the same as in the indexing process. Then, the fifth step is cluster related, but instead of creating the clusters, the query is compared against the cluster models generated during the indexing process to retrieve the most likely cluster for the query. At this point, CE-RSR automatically dismisses REST services that are not in the selected cluster as irrelevant. This effectively reduces the search space. The final step consists in querying the Lucene index for the selected cluster, this step is performe by the Lucene framework.

IV. Evaluation

In order to assess CE-RSR effectiveness, a 293 WADL file data-set was gathered from the Internet. This data-set was manually gathered by UNICEN system engineering major students. The students used Google for searching file of the type WADL and manually determined which of the Google results were real WADL files. For this data-set, we created a set of 14 queries and manually determine which WADL files are relevant to each query. This mapping query-WADL file was used to assess CE-RSR effectiveness. We use three classic IR metrics, namely Recall, Precision and F-Measure, to assess CE-RSR and determine which clustering technique performed better [16].

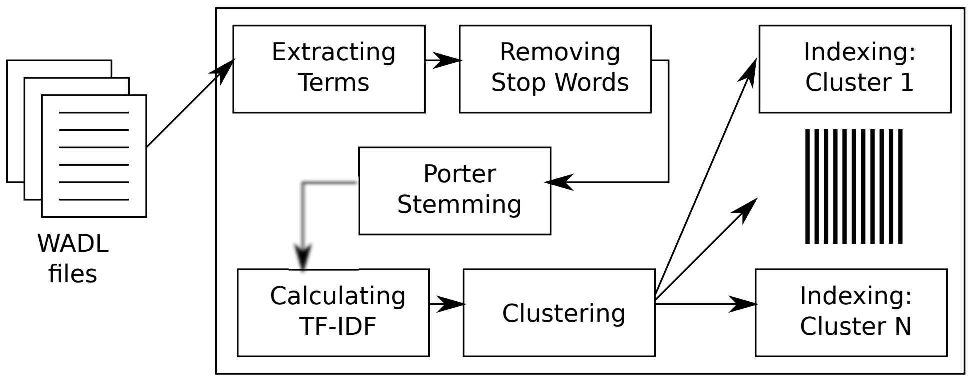

Figure 1. CE-RSR indexing process

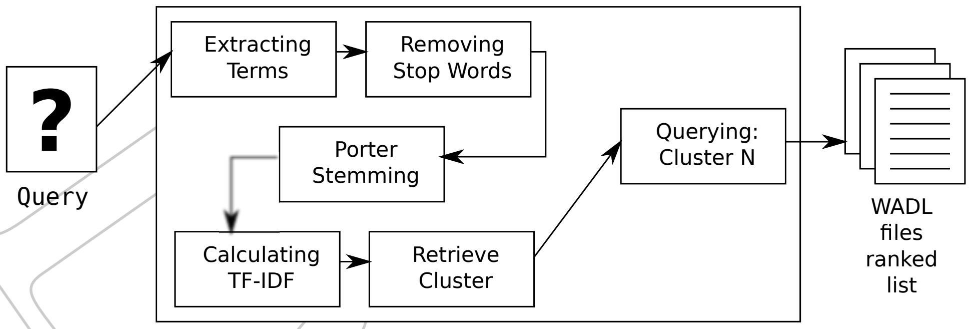

Figure 2. CE-RSR Querying Process

Recall is a metric that measures the fraction of relevant elements that were retrieved by the registry. For example, if the data-set contains 4 WADL files relevant for a given query and the retrieved list of WADL files only has 1 of them, Recall is 0.25. Formally, Recall is defined as:

$$Recall = \frac{|retrivedRelevantDocuments|}{|relevantDocuments|}$$

Precision measures the fraction of relevant elements in the retrieved list. For example, if the registry retrieved 5 WADL files for a given query, but only 2 of them are relevant for the query, Precision is 0.2. Formally, Precision is defined as:

$$Precision = \frac{|retrivedRelevantDocuments|}{|retrievedDocuments|}$$

Finally, F-Measure metric combines the two previous metrics using the harmonic mean. F-Measure is defined as:

$$F - Measure = 2 * \frac{Precision*Recall}{Precision+Recall}$$

The reported results are the arithmetic means of the values obtained for each query. We have also used variants of this metrics known as Recall at-X, Precision at-X and F-Measure at-X. These metrics are defined as above, but only the first X retrieved documents are considered. This was done because users who search in the Internet are highly likely to consider only the first results and disregard the rest. Furthermore, the probability that a user considers a given results decreases drastically with the result position [21], being nearly nonexistent after the tenth position. Therefore, we have considered Recall at-10, Precision at-10 and F-Measure at-10. Finally, we have also calculated Precision at-1, which can be seen as the fraction of queries for which the first retrieved result was relevant.

Table I
LUCENE WITHOUT CLUSTERING

	Lucene
F-Measure	0.244
Recall	1.000
Precision	0.139
F-Measure at-10	0.412
Recall at-10	0.833
Precision at-10	0.273
Precision at-1	0.933

To provide a baseline, we indexed the WADL files using Lucene without any clustering technique. However, we performed the same prepossessing that CE-RSR applies to the WADL files. Otherwise, the WADL keywords might introduce irrelevant terms that negatively impact on Lucene performance. Notice that this evaluation is also relevant because a previous work [8] have suggested using Lucene for indexing/retrieving REST services.

Table I presents the results obtained for Lucene. Notice that the achieved Recall is 1, i.e., a perfect score. However, the achieved Precision is 0.139, i.e., which is very low. The Recall is 1because Lucene retrieves a large list of candidate WADL files in which there are all the relevant REST services. However, the Precision is 0.139 because only a few of the retrieved REST services are relevant. This means that the retrieved REST service list comprises all the relevant REST services plus many more that are not relevant. This negatively impacts on the F-Measure for which the obtained value is also low. However, when considering the first 10 results, Precision and F-Measure are significantly higher, though the obtained Recall value is lower. This indicates that Lucene tends to retrieve the relevant documents in the first positions. Furthermore, Precision at-1 was 0.933 meaning that for most of the queries a relevant WADL file was retrieved in the first position.

We evaluated CE-RSR using different clustering techniques provided by Weka, namely K-means, Hierarchical, X-means and EM. We also have implemented BK-means, which consists in a hierarchical cluster of N levels in which a K-means of 2 clusters is applied. For example, BK-means with 1 level divides the input set into 2 cluster. However, when BK-means is run using two levels, the input set is firsly divided into 2 cluster, and then each cluster is also divided into 2 cluster resulting in 4 final clusters.

The behavior of these clustering algorithms can be modified using different parameters. For instance, K-means can use different distance metrics, such as Euclidean or Manhattan distance; besides, the number of the cluster is also a parameter for K-means. In our experiments, we have only varied the parameters related with the number of cluster. The selected values for the other parameters were the default Weka parameters for each clustering technique. Regarding the parameters related with the number of clusters, these were the selected value:

- the number of clusters for K-means was set in values ranging from 5 to 45
- the number of clusters for Hierarchical was set in values ranging from 5 to 35
- the maximum number of clusters for X-means was set to 50
- the maximum number of clusters for EM was set to no limit.
- finally, for BK-means, which is not included in Weka, we evaluated different number of levels ranging from 2 to 7.

Table II presents the obtained results for CE-RSR with K-means clustering. Table III shows CE-RSR with BK-means results and Table IV shows the results for CE-RSR with Hierarchical clustering, X-means and EM. The three tables present not only the metric values, but also the improvement when compared to the Lucene baseline. The improvement is enclosed within parentheses and can be negative, which means that for that particular metric the CE-RSR performed worse than Lucene. Finally, the better results within the three tables are highlighted in bold.

The greater improvement for F-Measure was obtained using K-means, particularly K-means with 45 clusters obtained the greatest F-Measure. Using BK-means also caused a strong improvement in most cases, but F-Measure resulted in a similar level when using BK-means with 6 levels, i.e., 64 clusters. CE-RSR with Hierarchical clustering brings about mixed results, generally unfavorable. X-mean improved the results of F-Measure in approximately a 100%. Although this value is high, the improvements are significantly lower than the ones obtained with K-means. Finally, EM cluster results were fairly similar to the ones achieved using Lucene.

Regarding Recall, Lucene achieved a perfect score. Therefore, CE-RSR did not improve Recall, and generally performed slightly worse. Since the Recall metric cannot be improved, all the improvements in F-Measure stemmed from the improvements in Precision. In particular, in the CE-RSR using K-means with 45 clusters, which had the highest F-Measure improvement, the Recall was 0.85, i.e., 15% less than the one obtained by Lucene. However, the Precision for CE-RSR using K-means with 45 clusters was 0.688, which is almost 4 times the Precision obtained by Lucene.

Regarding F-Measure at-10, Recall at-10 and Precision at-10, the best results were also obtained by CE-RSR using K-means with 45 clusters. Notably for this scenario, these three metric values were the same to the ones for F-Measure, Recall and Precision. Despite this, the improvements were significantly lower because F-Measure at-10, Recall at-10 and Precision at-10 for Lucene were higher than their unrestricted counterpart, namely F-Measure, Recall and Precision.

The final metric in Tables II, III and IV is Precision at-1. For this metric, Lucene had reached an score of 0.933, i.e. a near maximum score. As a result, Lucene outperformed CE-RSR with most clustering techniques. However, CE-RSR using K-means with 45 clusters obtained the same score as Lucene.

The best performance for CE-RSR was obtained using K-means with 45 clusters. This combination reported the highest scores for six out of the seven metrics. Notice that for Precision at-1 Lucene also achieved the highest score. The only metric for which CE-RSR using K-means with 45 clusters did not reach the highest score was Recall. In this case, Lucene outperformed CE-RSR using K-means with 45 clusters by a 15%. However, the Recall metric value was 0.85, which is a high Recall score. Furthermore, the reduction on this metric is compensated by the gain in the Precision metric resulting in an increase of the F-Measure of 211%.

Finally, since K-means was the clustering algorithm that obtained the best performance, we further evaluate this algorithm adding a wider range of clusters. For this experiment, the number of cluster ranged from 5 to 50 with an step of 5. Figure 3 details the experimental results. Recall varies from 0.65 to 0.85, but the value is relatively stable as the number of clusters increases. In contrast, Precision tends to grow when more clusters are added. As a result of this, F-Measure also tends to grow when more cluster are added. After 40 clusters, Precision seems to be stable. Once again, the best results for CE-RSR was obtained using 45 clusters, but K-means with both 40 and 50 clusters reported similar results.

V. Conclusions

This paper presents a novel approach for searching REST services. This approach combines techniques from IR with clustering techniques. Our results point out that using an appropriate clustering algorithm can improve IR based REST service registries effectiveness. The experiments have taken

Table II
CE-RSR RESULTS I

	K-means 5	K-means 15	K-means 25	K-means 35	K-means 45
F-Measure	0.369 (51.22%)	0.406 (66.39%)	0.587 (140.57%)	0.696 (185.24%)	**0.76 (211.47%)**
Recall	0.844 (-15.6%)	0.64 (-36%)	0.767 (-23.3%)	0.823 (-17.7%)	0.85 (-15%)
Precision	0.236 (69.78%)	0.297 (113.66%)	0.476 (242.44%)	0.603 (333.81%)	**0.688 (394.96%)**
F-Measure at-10	0.426 (3.39%)	0.411 (-0.24%)	0.57 (38.34%)	0.67 (62.62%)	**0.762 (84.95%)**
Recall at-10	0.778 (-6.6%)	0.573 (-31.21%)	0.7 (-15.96%)	0.757 (-9.12%)	**0.85 (2.04%)**
Precision at-10	0.293 (7.32%)	0.32 (17.21%)	0.481 (76.19%)	0.601 (120.14%)	**0.691 (153.11%)**
Precision at-1	0.733 (-21.43%)	0.667 (-28.51%)	0.8 (-14.25%)	**0.933 (0%)**	**0.933 (0%)**

Table III
CE-RSR RESULTS II

	BK-means 2	BK-means 4	BK-means 6	BK-means 7
F-Measure	0.446 (82.78%)	0.53 (117.21%)	0.587 (140.57%)	0.247 (1.22%)
Recall	0.757 (-24.3%)	0.668 (-33.2%)	0.593 (-40.7%)	0.933 (-6.69%)
Precision	0.317 (128.05%)	0.44 (216.54%)	0.581 (317.98%)	0.142 (2.15%)
F-Measure at-10	0.456 (10.67%)	0.507 (23.05%)	0.551 (33.73%)	0.388 (-5.82%)
Recall at-10	0.69 (-17.16%)	0.601 (-27.85%)	0.527 (-36.73%)	0.767 (-7.92%)
Precision at-10	0.34 (24.54%)	0.439 (60.8%)	0.578 (111.72%)	0.26 (-4.76%)
Precision at-1	0.8 (-14.25%)	0.733 (-21.43%)	0.667 (-28.51%)	0.867 (-7.07%)

Table IV
CE-RSR RESULTS III

	Hierarchical 5	Hierarchical 15	Hierarchical 25	Hierarchical 35	X-means	EM
F-Measure	0.287 (17.62%)	0.248 (1.63%)	0.197 (-19.26%)	0.11 (-54.91%)	0.492 (101.63%)	0.249 (2.04%)
Recall	0.967 (-3.3%)	0.883 (-11.7%)	0.75 (-25%)	0.617 (-38.3%)	0.739 (-26.1%)	**1 (0%)**
Precision	0.168 (20.86%)	0.144 (3.59%)	0.113 (-18.7%)	0.06 (-56.83%)	0.369 (165.46%)	0.142 (2.15%)
F-Measure at-10	0.421 (2.18%)	0.36 (-12.62%)	0.29 (-29.61%)	0.231 (-43.93%)	0.493 (19.66%)	0.412 (0%)
Recall at-10	0.844 (1.32%)	0.717 (-13.92%)	0.583 (-30.01%)	0.594 (-28.69%)	0.672 (-19.32%)	0.833 (0%)
Precision at-10	0.28 (2.56%)	0.24 (-12.08%)	0.193 (-29.3%)	0.143 (-47.61%)	0.389 (42.49%)	0.273 (0%)
Precision at-1	**0.933 (0%)**	0.8 (-14.25%)	0.6 (-35.69%)	0.533 (-42.87%)	0.8 (-14.25%)	**0.933 (0%)**

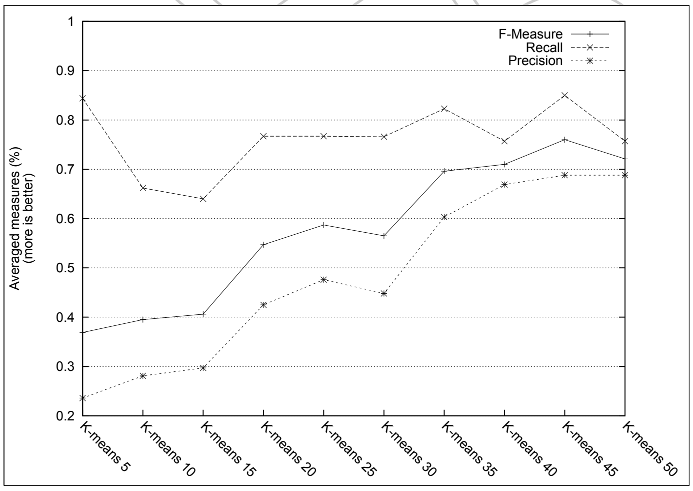

Figure 3. CE-RSR with K-means results

as a baseline the well-known Lucene framework, which has already been used as a REST service registry in a previous work [8]. Another contribution of this paper is a 293 WADL file data-set gathered from the Internet. This data-set can be used for evaluating future discovering approaches as well as REST service developing practices.

Currently, the main limitation of CE-RSR is that it indexes only WADL files, but there is no widely accepted standard for defining REST services [9], [8], [10], [15]. In future extensions, we plan to add the ability of indexing REST service described using other technologies and even simple Web pages, which is a very common practice [15]. Using Web pages for describing REST service introduces new problems that should be tackled. For instance, a Web page is likely to contain many more terms than a WADL file. This might negatively affect IR based registry performance when indexing both kind of documents. Furthermore, information in Web pages might be scattered in different pages, e.g., pages connected through hyperlinks. Even more, the content of the Web page might be dynamically generated through a combination of both server side programs and client side scripts.

Another future work is adding to CE-RSR the ability for dynamically exchanging REST services in runtime. Additionally, this feature might be used for automatically or semi-automatically generating mash-ups of REST services. Web Service mash-up is one of main features of SOAP based Web Services [22], and we think that similar approaches can be useful for REST services.

Finally, we think that the current trend of implementing REST services with no machine readable descriptions along with the fact that REST service provides disregard API quality [15] is harmful for service consumers. The lack of machine readable descriptions results in the impossibility of automatically generating stub code, which is common place in SOAP Web Services [23]. This, in turn, implies that human resources should spend significant time for implementing the glue code for each different REST service. For instance, Google have implemented Java and .Net libraries for consuming their Contacts REST service API[7]; also, there is a Java library for consuming Twitter REST services[8]. Additionally, changes in the REST API cannot be automatically detected and this might result in misbehavior of the service consumer system. Therefore, we will study how to generate these machine readable specifications in a semi-automatic manner from Web pages that describe REST services.

ACKNOWLEDGMENTS

We acknowledge the financial support by ANCPyT (grant PICT-2013-0464) and CONICET (grant PIP 2013-2015, code 11220120100185CO).

[7]Google Contact API: https://developers.google.com/google-apps/contacts/v3/

[8]JTwitter: http://www.winterwell.com/software/jtwitter.php

REFERENCES

[1] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[2] M. Arroqui, C. Mateos, C. Machado, and A. Zunino, "Restful Web Services improve the efficiency of data transfer of a whole-farm simulator accessed by android smartphones," *Computers and Electronics in Agriculture*, vol. 87, pp. 14–18, 2012.

[3] T. Erl, *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Wiley, 2005.

[4] M. Crasso, A. Zunino, and M. Campo, "A survey of approaches to Web Service discovery in Service-Oriented Architectures," *Journal of Database Management*, vol. 22, pp. 103–134, 2011.

[5] C. Wu, "WSDL term tokenization methods for ir-style Web Services discovery," *Science of Computer Programming*, vol. 77, no. 3, pp. 355–374, March 2012.

[6] J. Lathem, K. Gomadam, and A. Sheth, "SA-REST and (s)mashups : Adding semantics to RESTful services," in *Semantic Computing, 2007. ICSC 2007. International Conference on*, Sept 2007, pp. 469–476.

[7] D. John and M. S. Rajasree, "A framework for the description, discovery and composition of RESTful semantic Web Services," in *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, ser. CCSEIT '12. New York, NY, USA: ACM, 2012, pp. 88–93.

[8] Q. Liu, C. Liu, H. Li, X. Xu, and L. Gao, "Towards automatic discovering for a real-world RESTful Web Service," in *Web Information Systems and Applications Conference (WISA), 2012 Ninth*, Nov 2012, pp. 39–42.

[9] J. Kopecky, K. Gomadam, and T. Vitvar, "hrests: An html microformat for describing RESTful Web Services," in *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on*, vol. 1, Dec 2008, pp. 619–625.

[10] D. Roman, J. Kopecký, T. Vitvar, J. Domingue, and D. Fensel, "WSMO-Lite and hRESTS: Lightweight semantic annotations for Web Services and RESTful APIs," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. In Press, pp. –, December 2014.

[11] Y. Hao, Y. Zhang, and J. Cao, "Web Services discovery and rank: An information retrieval approach," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1053–1062, October 2010.

[12] A. Khattak, Z. Pervez, W. Khan, A. Khan, K. Latif, and S. Lee, "Mapping evolution of dynamic web ontologies," *Information Sciences*, vol. 303, pp. 101–119, May 2015.

[13] J. Rodriguez, M. Crasso, C. Mateos, A. Zunino, and M. Campo, "Bottom-up and top-down cobol system migration to Web Services: An experience report," *Internet Computing, IEEE*, vol. 17, no. 2, pp. 44–51, March-April 2013.

[14] J. M. Rodriguez, M. Crasso, A. Zunino, and M. Campo, "Improving Web Service descriptions for effective service discovery," *Science of Computer Programming*, vol. 75, no. 11, pp. 1001 – 1021, 2010.

[15] T. Espinha, A. Zaidman, and H.-G. Gross, "Web API growing pains: Loosely coupled yet strongly tied," *Journal of Systems and Software*, vol. 100, pp. 27–43, February 2015.

[16] M. Crasso, A. Zunino, and M. Campo, "Easy Web Service discovery: a Query-by-Example approach," *Science of Computer Programming*, vol. 71, no. 2, pp. 144–164, 2008.

[17] L. Chen, G. Yang, W. Zhu, Y. Zhang, and Z. Yang, "Clustering facilitated Web Services discovery model based on supervised term weighting and adaptive metric learning," *International Journal of Web Engineering and Technology*, vol. 8, no. 1, pp. 58–80, 2013.

[18] M. F. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, pp. 130–137, 1980.

[19] D. Pelleg and A. W. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," in *Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, 2000, pp. 727–734.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[21] E. Agichtein, E. Brill, S. Dumais, and R. Ragno, "Learning user interaction models for predicting web search result preferences," in *29th Annual International ACM SIGIR Conference on Research and*

Development in Information Retrieval (SIGIR '06), Seattle, Washington, USA. New York, NY, USA: ACM Press, 2006, pp. 3–10.

[22] M. Blake and M. Nowlan, "Knowledge discovery in services (kds): Aggregating software services to discover enterprise mashups," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 6, pp. 889–901, 2011.

[23] J. M. Rodriguez, M. Crasso, C. Mateos, and A. Zunino, "Best practices for describing, consuming, and discovering Web Services: A comprehensive toolset," *Software: Practice and Experience*, vol. 43, pp. 613–639, 2013.