Zoè Lacroix
Edna Ruckhaus
Maria-Esther Vidal (Eds.)

RED'11

Fourth International Workshop on REsource Discovery

Workshop co-located with the 8th Extended Semantic Web Conference (ESWC 2011)

Heraklion, Greece, May 30, 2011

Proceedings

Editors' addresses:
Arizona State University
{zoe.lacroix}@asu.edu
Universidad Simón Bolívar
Department of Computer Science
Valle de Sartenejas
Caracas 1086, Venezuela

{ruckhausl | mvidal}@ldc.usb.ve

Preface

This volume contains abstracts from the technical program of the Fourth International Workshop on REsource Discovery, held on May 30, 2011. After three successful events, first in Linz, Austria, joined to IIWAS (2008), then in Lyon, France, colocated with VLDB (2009), and finally in Pontoise, France, joined again to IIWAS (2010), the fourth International Workshop on REsource Discovery (RED 2011) was held together with ESWC in Heraklion, Greece.

A resource may be a data repository, a database management system, a linked data endpoint, a link between resources, a semantic wiki, or a Web service. Resources are characterized by core information including a name, a description of its functionality, its URLs, and various additional Quality of Service parameters that express its non-functional characteristics. Resource discovery is the process of identifying, locating and selecting existing resources that satisfy specific functional and non-functional requirements. Current research includes crawling, indexing, ranking, clustering, and rewriting techniques, for collecting and consuming the resources for a specific request.

The Fourth International Workshop on REsource Discovery is aimed at bringing together researchers from the database, artificial intelligence and semantic web areas, to discuss research and experiences in developing and deploying concepts, techniques and applications that address various issues related to resource discovery.

We received 11 submissions, out of which we selected nine for inclusion in the digital and printed proceedings. We set up an exciting program which included a talk on Linked Services given by invited speaker, Andreas Harth. We organized four sessions, two sections on Scalable architectures for Resource Discovery, one section on Ontology-based Resource Discovery, and one section on Applications in Life Sciences.

We thank the 21 members of our Program Committee, the invited speaker and the authors for their valuable contribution to the workshop. We are also grateful to ESWC organizers for their support in making this meeting successful. We kindly acknowledge the National Science Foundation for supporting student travel (grant IIS 0944126), and the DID-USB.

May 2011

Zoé Lacroix, Edna Ruckhaus,
Maria-Esther Vidal

Workshop Chairs and Organizing Committee

Zoè Lacroix, Arizona State University and Translational Genomics Research Institute, USA
Edna Ruckaus, Universidad Simón Bolívar, Venezuela
Maria-Esther Vidal, Universidad Simón Bolívar

Program Committee

Mohammad Alrifai, University of Hannover, Germany.
José Luis Ambite, University Southern California, USA.
Yudith Cardinale, Universidad Simón Bolívar, Venezuela.
Vassilis Christophides, ICS-FORTH, Greece.
Oscar Corcho, Universidad Politecnica de Madrid, Spain.
Joyce El Haddad, Universite Paris-Dauphine, France.
Alberto Fernández, Universidad Juan Carlos I, Spain.
Manolis Gergatsoulis, Ionian University, Greece.
Marlene Goncalves, Universidad Simón Bolívar, Venezuela
Andreas Harth, AIFB, Karlsruhe Institute of Technology, Germany.
H.V. Jagadish, University of Michigan, USA.
Gunter Ladwig, AIFB, Karlsruhe Institute of Technology, Germany.
Maria Maleshkova, KMI, The Open University, United Kingdom.
Kunal Patel, Ingenuity Systems, USA.
Marta Rukoz, Paris Ouest Nanterre La Defense University, France.
Fatiha Sais, IUT Orsay (Paris-Sud 11 University), France.
Sherif Sakr, National ICT Australia (NICTA) and University of New South Wales (UNSW), Australia.
Miguel-Angel Sicilia, University of Alcala, Spain.
Hala Skaf-Moli, Nantes University, LINA, France.
Dimitrios Skotas, University of Hannover, Germany.
Maciej Zaremba, DERI and National University of Ireland, Ireland.

Contents

Scalable Discovery of Linked Services

Barry Norton and Steffen Stadtmüller

AIFB, Karlsruhe Institute of Technology, Germany
firstname.lastname@kit.edu

Abstract. Linked Services bring together the principles and technology which define Linked Data with those of RESTful services. Services and APIs are thus enriched by, and contribute to, the Web of Data. A number of approaches to the semantic description of services have chosen SPARQL graph patterns as a means to describe input and output expectations. This paper describes two contributions. Firstly, the extension of discovery mechanisms offered over such Linked Service descriptions for use at application design time. Secondly, the realisation of this discovery approach in a distributed fashion using the Hadoop implementation of MapReduce, and an evaluation using private cloud infrastructure.

1 Introduction

Linked Data is identified with the application of a set of principles, and related best practice, regarding the use of RDF, SPARQL and HTTP in the publication of data. Linked Open Data is the result of the application of these principles in combination with the push to opening up public sector and other data.

Linked Open Services (LOS) [3] and Linked Data Services (LIDS) [9] look beyond the exposure of a fixed datasets using HTTP, SPARQL and RDF and consider how the data that results from computation over input can carry explicit semantics and be interlinked with existing Linked Data sets, and other dynamic data, thus also contributing to the Web of Data. It is natural that Linked Data's use of HTTP should be extended, consistently with REST [2], and that RDF should be made available, at least as an alternative via Content Negotiation, to encode representations to achieve these aims.

Other work [6] — which introduces the name 'Linked Services' that we consequently use to generalise over the different approaches to the combination of Linked Data and RESTful service approaches — introduces the idea that service descriptions should also be exposed as Linked Data. The original approach, pursued in the SOA4All project[1] and leading to the iServe service repository[2], uses a lightweight RDF version of the models used over many years of the literature on 'Semantic Web Services', the so-called 'Minimal Service Model'[3]. This uses semantic technology to characterise the input and output expectations of a service by annotating structural message parts with ontology classes.

[1] http://www.soa4all.eu/
[2] http://iserve.kmi.open.ac.uk/
[3] http://cms-wg.sti2.org/minimal-service-model/

LIDS and LOS both base their service descriptions on the notion that Linked Data provides a better description for services' input and output requirements: the graph patterns provided by the SPARQL query language. These provide the advantage of familiarity to Linked Data producers and consumers, but also of a more thorough description of what should be communicated and the possibility for increased tool support. One such kind of tool support is the extension of the notion of service discovery to better support a data-oriented view. This paper is organised as follows: Section 2 explores this motivation further and introduces the idea of design-time discovery of services using graph patterns; Section 4 sketches our approach to implementing such a mechanism in a distributed, scalable fashion; Section 5 details our evaluation of the current status of this implementation and justifies its general feasibility; Section 6 then provides conclusions and introduces our planned future work.

2 Motivation

In order to motivate the advantages of graph-based IO description we introduce an example based on wrapping some social network platform that only provides an API, not a dump of Linked Data (there are many such, so we do not need to be specific). A common feature of these social networks is the display of a 'birthday list', i.e. a list of people with a birthday on the day of access. Encouraging use of APIs, rather than crawling, a call might be provided to retrieve such a list. The Linked Services approach would be to wrap this call to provide the information in RDF, reusing existing vocabularies. Here the natural choice is the 'Friend of a Friend' (FOAF) vocabulary[4].

A wrapping for such a service then, putting aside issues of security and credentials for which our solution would build on WebID[5], would make explicit the input of a person identified in that social network, and an output that gives the age of certain friends. Furthermore, following Linked Open Service principles[6], reproduced in Table 1, we would use graph patterns to do so (Principle 1), where the `foaf:knows` predicate is the link between the input and output (Principle 3). The resulting input and output descriptions for the service are represented in Table 2

1. Describe services' input and output as **SPARQL** graph patterns
2. Communicate **RDF** by RESTful content negotiation
3. The **output** should make explicit its **relation** with the **input**

Table 1. (Core) Linked Open Service Principles

[4] `http://xmlns.com/foaf/spec/`
[5] `http://www.w3.org/wiki/WebID`
[6] `http://linkedservices.org/wiki/LOS_Principles`

Input:	{?user a foaf:Person; sn:id ?uid.}
Output:	{?user foaf:knows [sn:id ?fid; foaf:age ?age].}

Table 2. Basic Description for Example Service

We elide the namespace prefix declarations — for `foaf` and `sn`, the social network — as these are defined across the service description. Note that the reuse of the variable `?user` across input and output imply that this will have the same binding in the output as provided in the input. In SPARQL terms these would be 'safe variables' if we considered the service to be a CONSTRUCT query from the input to the output patterns (which indeed is the approach taken in Linked Data Services, where the same type of service descriptions are given) .

The advantages of this graph-based approach to the description of inputs and outputs can be seen in comparing the description to a traditional 'semantic web service' model such as OWL-S[7], where syntactic messages (with the implicit assumption that these will be bourne in plain XML) are annotated simply with classes. Here both the input and output message of the service would simply have to be annotated with the `foaf:Person` class. Any other information on which is required for input, and can be expected from output, would either be hidden in non-semantic transforms for 'lifting' and 'lowering' (usually using XSLT) and/or in pre- and post-conditions in a non-standard rule language (usually SWRL). In fact it is not clear whether properties applied to instances in a postcondition are expected to the returned in the services communications as these are also used to indicate changes in the state of the world outside of the communications (e.g., the dispatch of a physical book in a book ordering service).

The Web Service Modeling Ontology (WSMO)[8] improves somewhat on the vague 'semantic' description of accepting instances of Person, and providing back instances of Person, by two means. Firstly, WSMO models an explicit *choreography*, which in the Web Service Modeling Language (WSML) is defined by a language fragment that indicates not just which classes (called concepts in WSML) are communicable, but also which *relations* (a n-ary generalisation of binary RDF properties) are. Unfortunately, indicating that instances of `foaf:age` are communicated does not tell the user whether the submitted user's age is returned, or whether their friends' ages are. In other words it could, in graph pattern descriptions, correspond either to the output shown in Table 2 or of: `?user foaf:age ?age; foaf:knows ?friend.`

Secondly, WSMO distinguishes *assumptions* from preconditions, and *effects* from postconditions, so that pre- and post-conditions may apply only to communicated data. Unfortunately WSMO implementations do not respect this difference. In fact IRS-III [1], a version of the Internet Reasoning Service that is one of the two reference platforms for WSMO, deliberately uses conditions in the assumption field to judge the applicability of services, given concrete input data (and not the exterior 'state of the World' that they are otherwise specified to model).

[7] `http://www.w3.org/Submission/OWL-S/`
[8] `http://www.w3.org/Submission/WSMO/`

The LOS and LIDS approaches to the use of graph patterns for service descriptions can be traced back to an extension of the OWL-S, in the work described in [8]. Here, though, this was cast slightly differently: as pre- and post-conditions. It is our belief that simply replacing the entire class-based annotation of messages with patterns is ultimately more comprehensible and useful than trying to reconcile an annotation with these conditions, especially since they have unclear semantics that are not directly related to communication. It is our hope, furthermore, that this approach will continue to grow in popularity due to the more useful descriptions that can be formed, while sticking to the 'lightweight' semantic languages — RDF(S) and SPARQL — that have been at least partly responsible for the rapid growth of Linked Data. For this reason we introduce a discovery mechanism that aids in the use of such service descriptions.

3 Approach

In our approach to discover Linked Services with graph pattern-based I/O descriptions we allow service requests to be formulated as *service templates* with the the same syntax than the service descriptions. So a service template is also a pair of SPARQL graph patterns: one representing the set of all possible *input* RDF graphs an agent (human or machine) can provide for the invocation of a service and one representing a template of *output* RDF graphs such an agent expects to be delivered from the service. Therefore service templates encapsulate all necessary request information needed, including the expected relation between the input and output.

Therefore the question of whether a given service description matches a service template correlates to the problem of graph pattern containment. The input graph pattern of a service description must be contained in the service template's pattern; every graph that satisfies the template input graph pattern must also satisfy the service description's input graph pattern. Intuitively this is to say that the input an agent can provide fulfils the needs of the service to be invoked. Note that this also allows for the agent to talk about additional data he can provide for service invocation even though a matching service does not require them.

Matching the output graph patterns works in an analogous way. The output graph pattern of a service description contains the output graph pattern of a template; i.e., every graph that satisfies the service description output graph pattern also satisfies the template output graph pattern. So the required containment relation of the output patterns is dual to that of the input graph patterns. Intuitively again this means a service output has to provide enough to satisfy the request, but can provide more.

The matching based on graph pattern containment subsists in two binary decisions (one for the input and one for the output), answering if a service description completely matches a service template. In addition to this, in order to provide a more flexible discovery approach, we allow for the ranking of service descriptions against service templates by providing a number of continuously-valued matching metrics.

To achieve this, we introduce two additional metrics:

- The *predicate subset ratio* (psr) measures to what degree the set of predicates used in one pattern are subsumed within the set used in another.
- The *resource subset ratio* (rsr) measures to what degree the set of named resources, in subject or object position, used in one pattern are subsumed with those of another pattern.

Intuitively these metrics indicate to what degree a service description and a service template are using the same vocabulary. If a service description does not match a template in terms of pattern containment completely, they allow to test whether they at least use some of the same resources and predicates (and to what degree). Therefore they provide a mechanism to discover services, which are close to a given template, but are not necessarily completely matching.

Similarly to the pattern containment we have to distinguish between the metrics for input and output. Since a template input graph pattern can offer more data than actually needed by a described service without endangering their compatibility, the subset ratios for the input patterns have to measure, to what degree the resources (respectively predicates) in the service descriptions are used in the service template. For the subset ratios of the output patterns this works the other was around, because a described service can offer more output than required be the template. So in this case the subset ratios have to measure, to what degree the resources (respectively predicates) in the template are used in the service description. The Equations (1)-(4) show how the metrics are calculated.

$$psr_{input} = \frac{\#\left(\{predicates\ in\ template\} \cap \{predicates\ in\ service\ description\}\right)}{\#\left\{predicates\ in\ service\ description\right\}} \tag{1}$$

$$psr_{output} = \frac{\#\left(\{predicates\ in\ template\} \cap \{predicates\ in\ service\ description\}\right)}{\#\left\{predicates\ in\ template\right\}} \tag{2}$$

$$rsr_{input} = \frac{\#\left(\{resources\ in\ template\} \cap \{resources\ in\ service\ description\}\right)}{\#\left\{predicates\ in\ service\ description\right\}} \tag{3}$$

$$rsr_{output} = \frac{\#\left(\{resources\ in\ template\} \cap \{resources\ in\ service\ description\}\right)}{\#\left\{resources\ in\ template\right\}} \tag{4}$$

Note, that if a graph pattern is contained in another one (i.e., the binary decision is positive), the subset ratios must necessarily result in a metric of 1.0.

As an example again consider an agent looking for a service that takes information about a person represented in the FOAF vocabulary as input, and provides the name and the age of the friends of this specified person as output. Further consider two potentially useful service descriptions. The first is for a service that expects input exactly as proposed by the template, but only provides the age of the person with the `foaf:age` predicate. The second is for a service that provides the name, age and the OpenID as output with the predicates `foaf:name`, `foaf:age` and `foaf:openid` respectively, but additionally requires a userID from a specific social network (from which the service would actually obtain its data). In this case the input graph pattern of the template does not promise all the data needed to invoke the service. On the other hand, only this service covers all the data the template requires in the output (as well as some additional, not needed).

The discovery process is depicted in Figure 1. In this example we would find that the agent who submitted the template can invoke the service described with the first service description but he only gets a subset of the output data he demands for. But taking into account that the $psr_{output} = 0.75$ and the $rsr_{output} = 1.0$, this service is still close and could be presented to the agent, for example with an additional notification, that some of the requested output will not be there.

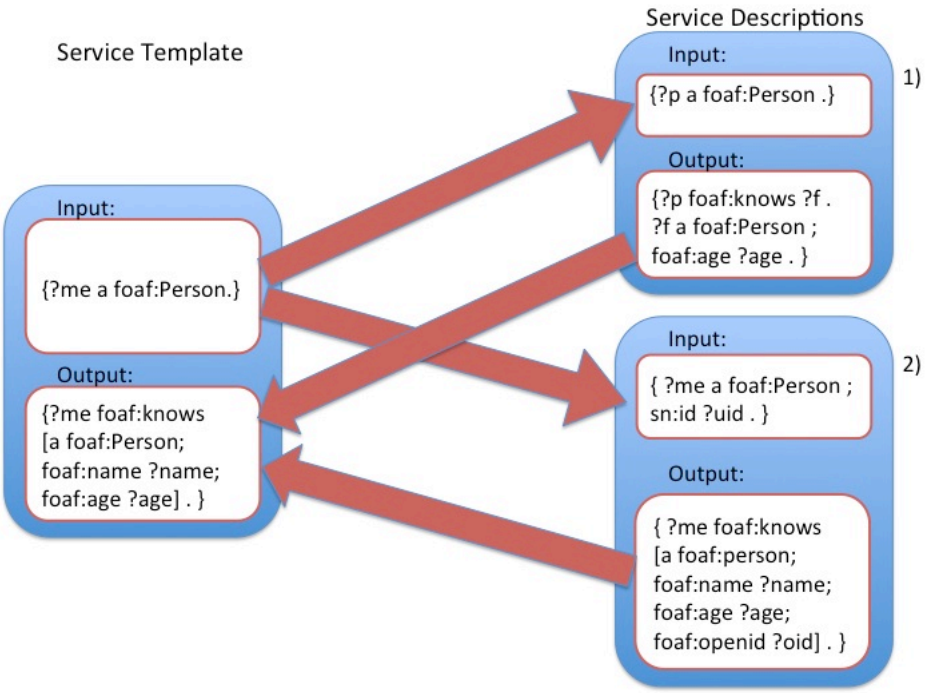
The second service description on the other hand represents a service that provides all the output data the agent is looking for. Here it is the input in the template that does not promise all the necessary data, from the service's point of view. Again, however, by looking at the subset ratios ($psr_{input} = 0.5$ and $rsr_{input} = 1.0$) it becomes apparent that this service could be useful for the agent, since it is very close to the template. In this case the agent could be specifically asked to provide the missing input for the service invocation.

To realize the described discovery approach and to address scalability we propose a discovery cloud, which provides as service description repository as well as template repository. This distribution cloud offers a RESTful [2] repository API to manage the service descriptions and to allow agents to submit new templates and retrieve the calculated metrics, or rather the discovered services ranked according to these.

Now the described semantic discovery process is applied at the following stages:

- When a new service template is uploaded to the discovery cloud, semantic discovery is used against every service description.
- When a new service description is uploaded to the discovery cloud every service template will be checked against this new service.

The insight pursued in our implementation, described in the following section, is that both of these problems can be structured as a MapReduce problem, with a map over the other type of resource, followed by a simple reduce. An important consideration, as with any MapReduce problem, is the locality of data; i.e., that the computation is reasonably well isolated from the communication of large amounts of data.



	Service Description 1)		Service Description 2)	
	input	output	input	output
pattern containment	YES	NO	NO	YES
psr	1.0	0.75	0.5	1.0
rsr	1.0	1.0	1.0	1.0

Fig. 1. Example for the discovery process

4 Implementation

In order to realise this approach for the discovery of Linked Services we implemented a system, that stores service descriptions and templates, both of which can be considered to consist of a URI as unique identifier and two SPARQL graph patterns that describe service input and output.

Work already exists on the provision of RESTful repositories for service descriptions, and provision of these according to Linked Data principles [4], [7]. Our implementation applies the same approach also to templates. In practical terms, when an RDF representation of a service description or template is HTTP POSTed to the repository, a URI is returned by which it is identified as a resource. In the standard REST manner, the resource can be updated by PUTting a new representation to the same URI, and it can be removed by a DELETE.

By managing templates as persistent resources, discovery against the template becomes an on-going task. Each retrieval of the list of services matching a template may be different due to two forms of dynamicity. First new service descriptions may be added to the repository, or indeed removed, between retrievals. Secondly, dynamic aspects of the service may affect ranking of each list; for instance the Quality of Service provided by the described service, as this is monitored over time.

When an agent (human or machine) would like to find services for a given task, then, a template can be POSTed. Every submitted template is stored and matched with all currently stored service descriptions. We use Jena[9], a Java framework for building Semantic Web applications, and ARQ[10], a query engine for Jena, to implement the matching mechanism. To determine if a service description matches a template we calculate, if the graph representing the input in the service description is 'contained' in the graph representing the input in the template; equivalent to checking whether the promised input is enough to invoke the described service. Analogously it is determined if the graph pattern representing the output of the template is contained in the graph representing the output in the service description. This is equivalent to checking, if the output of the described service is enough to satisfy the demands of the agent.

To determine these containment relations, the patterns for input and output (of templates and service descriptions respectively) are parsed and used as "WHERE"- clause of ARQ SPARQL ASK queries. Additionally the variables in both patterns are substituted by generated resources, resulting in graphs, which are skolemized versions of the original patterns. To test if a graph pattern A is contained a graph pattern B, we execute the ASK query of A over the skolemized version of B. The query will result in a positive answer, iff B contains A.

To allow for a sophisticated ranking, instead of just a binary discovery decision, we additionally calculate two other metrics: The *predicate subset ratio*, which measures the fraction of predicates used in the input graph pattern of the service description, that are also used in the input graph pattern in the template (and vice versa for output graph patterns). The *resource subset ratio* analogously measures the fraction of resources in subject or object position of the triple patterns in one graph that are used in the other.

[9] `http://openjena.org`
[10] `http://jena.sourceforge.net/ARQ`

These two metrics provide a continuum of matches be expressing to what degree the template and the service description use the same vocabulary. To calculate the two ratios for input and output respectively ARQ SPARQL SELECT queries are executed over the skolemized graphs to extract the set of predicates (resources) used in the graph patterns. Each set of metrics, generated in this way, for every combination of template and service descriptions is tagged with an identifier consisting of a combination of respective service description URI and template URI.

To allow service descriptions to be updated, or to populate the system with new service descriptions, an analogous process is employed. A submitted (via HTTP POST) service description is stored in the system and matched with all service templates. The resulting metrics are also tagged with an identifier and complement the already existing results. Thus, every combination of template and service description has a set of results that is persistently saved and can be retrieved from the system via HTTP GET.

If the system is populated with several thousands of service descriptions, the amount of calculations for determining all the metrics can be quite high. However, to provide for scalability of our approach we use Apache Hadoop[11], the open-source implementation of Google MapReduce. The Hadoop software is designed for distributed computation by dividing computation jobs into smaller sub-tasks, which can be executed in parallel on different nodes in a cluster of machines (map function). The results of these sub-tasks are retrieved and combined to achieve the overall goal of the original computation job (reduce function). For this purpose Hadoop implements a distributed file system (HDFS), which spans over an arbitrary number of computers. Data is stored on blocks of this file system; these blocks are distributed randomly over all nodes in the cluster. If the input data of a computing job is spanning over several blocks, a sub-task for every block is created and executed on the node where the block resides. Additionally the blocks can be replicated several times to provide a safe mechanism against failure of nodes.

To deploy our system we use OpenCirrus[12], a collaboration of several organizations to provide a research testbed for cloud-based systems. The Karlsruhe Institute for Technologies cluster within OpenCirrus makes use of OpenNebula[13] toolkit, an open source software used by OpenCirrus to build an "infrastructure as a service"-cloud (IaaS). This environment allows us to easily create and configure virtual machines that act as independent computers. We use these machines to set up a Hadoop cluster. This implies, that our cluster runs on top of a cloud, further abstracting from actual physical hosts.

We store the service descriptions, templates and matching results on a distributed HDFS storage, hosted on several virtual machines, as described. When a template, or service description, is submitted, Hadoop calculates the matching metrics by transferring and executing the code, that implements the matching mechanism together with the submitted template, to the nodes where the service descriptions (templates) are stored, rather than moving the data to the code. Furthermore Hadoop is "rack-aware", which means it always tries to use nodes close to each other (e.g. blades on the same rack in a datacenter) to reduce network traffic in the cluster.

[11] `http://hadoop.apache.org`
[12] `http://opencirrus.org`
[13] `http://www.opennebula.org`

Since our Hadoop system runs on virtual machines, whose communication is balanced by the OpenNebula Toolkit, we chose a flat structure, acting as if all nodes are hosted on the same rack. Finally Hadoop tries to balance the workload of the nodes, taking into account that some nodes contain the same data blocks due to the described replication mechanism.

After calculation of the metrics, the map function assigns a timestamp and the identifier to every set of metrics and passes the generated results to the reduce function. In our case the reduce function just gathers all the results and saves them persistently on the distributed file system. Since our system also allows for updating templates and service descriptions by re-submitting a new version of them with the same identifier, we have to run a second house-keeping MapReduce job. This second job compares the newly generated results with the results that are already stored. If a submitted service description is tagged with the same URI than a preexisting service description (i.e., an update is intended), some of the generated results will also have the same identifiers. In this case the older results are deleted, which can be checked by using the mentioned timestamps.

5 Evaluation

To evaluate our discovery system, especially in terms of scalability and in the absence of a large number of existing real service descriptions (since these number only in the double-figures so far, we have developed a generator. This create random pairs of related SPARQL graph patterns within boundaries, set by certain parameters, described below. These graph patterns can be interpreted as input and output of a service description or a template, because these are essentially equivalent.

For evaluation we generated 10 000 tuples used as service descriptions. Both patterns in these tuples are composed with a random number between 5 and 50 of triple patterns. The triple patterns for every respective pair are generated with resources in subject or object position, randomly drawn out of a local resource pool consisting of 10 to 50 different (URI-identified) resources. These local resources are randomly drawn out of a global pool of 500 resources. The predicates in the triple patterns of the tuples are also randomly drawn out of 3 to 25 different predicates in a local predicate pool. And again this local pool is randomly chosen out of a global pool of 250 predicates. So the difference between the local and global pools is, that the global pools of resources and predicates are used for all tuples, whereas the local predicates and resources are only consistent for both of the graph patterns within a tuple. This approach is chosen to establish a credible relationship between input and output.

Additionally the generator uses variables, rather than resources, in subject or object position with a probability of 0.3 in each case. A variable is used in predicate position with a probability of 0.2. In every tuple between 2 and 10 different variables are used. Since variables are already locally valid within one tuple, no global variable pool to draw from is needed. Additionally we generated a tuple of graph patterns used as template with the same parameters.

We populated the system with the service description using different setups with one, two, five, eight and ten worknodes in the Hadoop cluster, deployed on virtual machines of the OpenCirrus test bed. With every setup one additional virtual machine is needed to act as namenode for the Hadoop cluster. The namenode is used for the coordination of the distributed computation tasks, but does no computation itself. The distributed HDFS storage was configured with a block size of 1MB with a replication of factor 3 for every used block. The 10 000 service descriptions corresponded to 8.16MB of data and were therefore stored over 3 x 9 blocks on the cluster.

Then the matching process for the generated template over all service descriptions was triggered on every setup. We measured the execution time needed for the matching itself (i.e., first MapReduce job) and the overall execution, which includes the time needed for the second MapReduce job to combine the newly calculated with the preexisting metric sets. To provide for comparable results regarding the overall time, we did not prepopulate the system with results. Therefore the second MapReduce job used every time only the 10 000 newly calculated metric sets as input (i.e., one for every combination of service description and template).

worknodes	execution	time (sec)	mean (sec)	standard deviation (sec)	standard error (sec)
1	1.	477.3			
	2.	463.3	470.3	9.9	7.0
2	1.	283.7			
	2.	277	280.4	4.7	3.3
5	1.	169.7			
	2.	156	162.9	9.6	6.8
8	1.	155.3			
	2.	167.1	161.2	8.2	5.9
10	1.	134			
	2.	121.7	127.8	8.7	6.2

Table 3. measurements of overall execution time

worknodes	execution	time (sec)	mean (sec)	standard deviation (sec)	standard error (sec)
1	1.	394.3			
	2.	395.8	395	1	0.7
2	1.	223.6			
	2.	219.3	221.5	3	2.1
5	1.	120.6			
	2.	124	122.4	2.4	1.7
8	1.	121.7			
	2.	117.2	119.5	3.2	2.3
10	1.	81.4			
	2.	82.1	81.8	0.5	0.4

Table 4. measurements of exclusive matching time

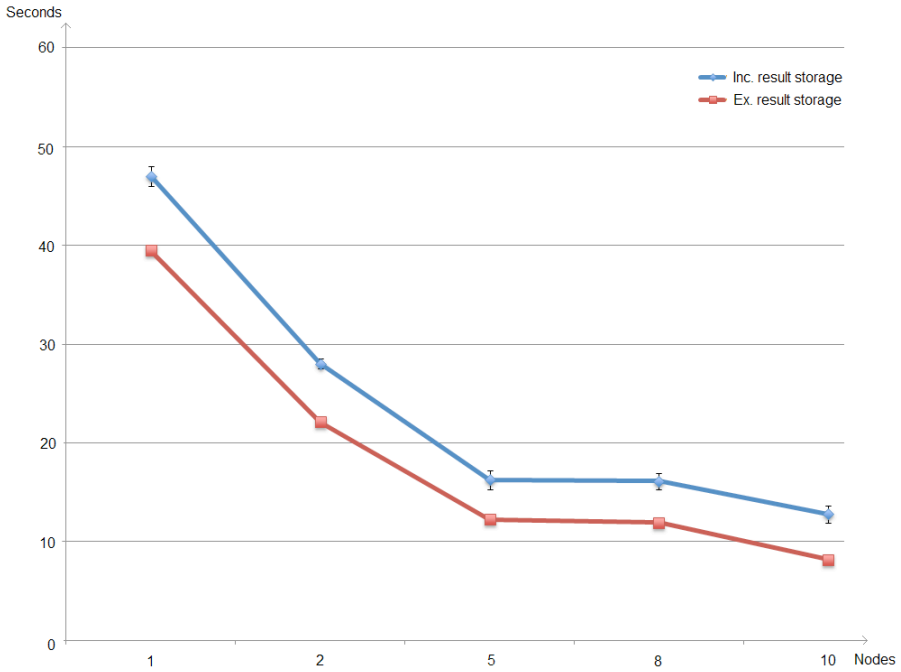


Fig. 2. graphical representation of measurements

To account for fluctuations in network traffic we measured the matching on each setup twice. The results are shown in Table 3 and Table 4 with a graphical representation in Figure 2. The calculation of the metrics alone took between 81.4 sec, on ten worknodes, and 395.8 sec, on one worknode. The overall execution time was measured between 121.7 sec using ten worknodes, and 477.3 sec on one worknode. It can be seen, that the system scales well by adding additional worknodes between one to five nodes. Between five and eight nodes the execution time stagnates almost completely. By using ten worknodes in the Hadoop cluster the measured times are further decreased compared to the setup with eight worknodes, although the improvement is less significant than in the area between one and five worknodes.

The behavior between one and five worknodes is easily accounted for by the possibility to execute more computations simultaneously. The more nodes are on the system, the more sub-tasks can be launched at the same time. By employing up to eight worknodes above five no further decrease in execution time is achieved because the Hadoop system was able at these settings to balance the workload with fewer nodes. Since the maximum number of map tasks executed by the system is determined by the amount of block the input data fills on the HDFS storage. So in our case not more than nine map tasks can be launched (eight, and one with a small input size of 0.19MB). Therefore on settings with one to five worknodes almost every worknode has to execute at least two map tasks. This provides the namenode with more posibilities to distribute the map tasks among the worknodes.

Such a distribution takes into account the fact that the worknodes contain non-disjunctive subsets of the overall input data set. The tasks can therefore be assigned in such a way that all employed worknodes contribute an equal amount of work to the calculation of the metrics. By using, for example, eight nodes, the namenode loses this possibility to some extent, since it always prefers parallel computation over load balancing (i.e., it will not wait for a worknode to finish if another worknode is already available). This also explains why a further, though diminishing, decrease of execution time is achieved by using more than eight nodes. In this case the namenode can (and must) decide not to use one of the nodes (and assigning the insignificant small map job to another). In this situation the least useful worknode is chosen to be disregarded by the namenode.

The effect of losing the possibility to balance the workload between the nodes can easily be avoided by choosing a smaller blocksize that allows for more map tasks than available nodes. But for our evaluation this observation also shows, that the improvement in terms of execution speed can not only be attributed to the increase of computation resources (i.e., adding additional CPUs and memory with every worknode), but also to the strategic distribution and execution of matching sub-tasks.

By comparing the results of the overall execution time and the matching time without housekeeping, similar observations can be made. The time needed to execute this second MapReduce job decreases due to the employment of a second worknode compared to a setup with only one node, but no further improvement can be achieved by adding additional nodes. The inputs for this second job are the calculated metrics, for every combination of service descriptions and template. They amounts to 2.68MB of data. So only 3 MapJobs can be start simultaneously.

The standard deviation and standard error of the individual results are also shown in the tables, and represented in the figure (as bars). For the overall execution time the standard deviation ranges between 4.7 sec and 9.9 sec, which results in a standard error between 3.3 sec and 7 sec. For the exclusive matching process the standard deviation is measured between 0.5 sec and 3.2 sec, which results in a standard error between 0.4 sec and 2.3 sec. Those values clearly indicate the stability of the system. The difference between the overall execution time and the exclusive matching time can be explained by the second MapReduce job that is included in the overall time, since most fluctuations in our measurements are due to the overhead time, that is needed to start a new MapReduce job.

Our results are not only valid for the matching a template over service descriptions, but also for populating the discovery system with a new service description, because they are syntactically equivalent and the process to submit a new service description is symmetrical to the process of submitting a new template. In other words the 10 000 used graph pattern tuples could have just as well been interpreted as templates, that are already stored on the system and one new service description (i.e., the former template) is submitted.

6 Conclusions and Future Work

In this paper we have: motivated the use of SPARQL graph patterns for the description of Linked Services; given an overview of one approach to discovery where both services and templates, representing data requirements from services, are described in this way; detailed an interface to such a discovery approach extending the existing notion of provision of RESTful and Linked Data-compliant registries; detailed a distributed implementation based on Hadoop; and described an evaluation of the scalability of such an implementation based on realistic parameters.

In future work we will expand on our coverage of SPARQL and RDF(S) in two major ways. Firstly, while the discovery approach detailed here concentrates on the conjunctive graph patterns, to which Linked Data Services restrict themselves, the Linked Open Services approach has motivated the use of disjunction. Consider, for example, a social network that allows its users to hide their full date of birth but to expose their birthdays. This means that for some friends they would be included in the results for a birthdays-based API call, but their age would not be included. In this case we might model the output as follows:

```
{?user foaf:knows ?friend.
 ?friend sn:id ?fid.
 OPTIONAL {?friend foaf:age ?age}.}
```

Similarly we have considered, related to the geospatial services considered in our previous work [5], that a service might be flexible in the use of vocabularies encoding input. While our existing services have used the Basic Geo Vocabulary[14] (usually given the prefix `wgs84`, `wgs84_pos`, or `wgs84_loc`), the Geo OWL ontology[15] follows GeoRSS in using Geography Markup Language (GML)-style objects (declared in a namespace usually given the prefix `gml`[16]) containing complex GML-defined literals. A service that accepts, as input, point in either of these encodings could be described as follows:

```
{{?point a wgs84:Point; wgs84:lat ?lat; wgs84:long ?long.}
 UNION
 {?point a gml:Point; gml:pos ?pos.}}
```

Similarly it could be a promising avenue to consider the use of FILTER expressions within the SPARQL graph patterns, indeed we could restrict, given dates of birth, the returned people in the running example to actually have their birthdays at the time of request by these means. This, however, seems like a better of use true rule-based post-conditions (or effects, in WSMO terminology) because it is not necessarily a property of the data communicated and the strength, demonstrated in this paper, of graph-based patterns is that they directly capture the information communicated by a service.

[14] `http://www.w3.org/2003/01/geo/`

[15] `http://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/W3C_XGR_Geo_files/geo_2007.owl`

[16] `http://www.opengis.net/gml`

The provision for inference in the containment of one graph pattern by another is also part of our immediate future work. In our running example, the specification of an instance of `foaf:Person`, and likewise `Point`, is redundant since this is the domain of required predicates. There may be many such ways in which a pattern that is not directly contained may necessarily infer, in all matching patterns, the matching of another pattern. Inference also affects our continuous metrics in useful and interesting ways. This leads on to one final notable piece of on-going work: an attempt to establish the most effective means to combine matching metrics to define a useful ranking of services with respect to a template.

Acknowledgement: The work is supported by the EU FP7 projects SOA4All (IP 215219), and PlanetData (NoE 257641). We thank our colleagues from these projects for valuable discussions on the topics of this paper.

References

1. Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., Pedrinaci, C.: IRS-III: A broker for semantic web services based applications. In: Proceedings of the 5th International Semantic Web Conference (ISWC2006). Athens, Georgia, USA (Nov 2006)
2. Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis, University of California, Irvine (2000)
3. Krummenacher, R., Norton, B., Marte, A.: Towards Linked Open Services. In: 3rd Future Internet Symposium (September 2010)
4. Norton, B., Kerrigan, M., Marte, A.: On the use of transformation and linked data principles in a generic repository for semantic web services. In: Proceedings of the 1st Workshop on Ontology Repositories and Editors for the Semantic Web (ORES-2010). No. 596, CEUR-WS (2010)
5. Norton, B., Krummenacher, R.: Geospatial linked open services. In: Proceedings of Workshop Towards Digital Earth (DE-2010). No. 640, CEUR-WS (2010)
6. Pedrinaci, C., Domingue, J., Krummenacher, R.: Services and the Web of Data: An Unexploited Symbiosis. In: AAAI Spring Symposium (March 2010)
7. Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J., Domingue, J.: iServe: a linked services publishing platform. In: Proceedings of the 1st Workshop on Ontology Repositories and Editors for the Semantic Web (ORES-2010). No. 596, CEUR-WS (2010)
8. Sbodio, M., Moulin, C.: SPARQL as an Expression Language for OWL-S. In: Workshop on OWL-S: Experiences and Directions at 4th European Semantic Web Conference (June 2007)
9. Speiser, S., Harth, A.: Taking the lids off data silos. In: Proceedings of the 6th International Conference on Semantic Systems (iSemantics). ACM International Conference Proceeding Series (2010)

Ontology-based User-defined Rules and Context-aware Service Composition System

Victoria Beltran[1], Knarig Arabshian[2], and Henning Schulzrinne[3]

[1] Dept of Telematics, Universitat Politècnica de Catalonia/Fundació I2Cat, Barcelona, Spain
[2] Alcatel-Lucent Bell Labs, New Jersey, USA
[3] Dept of Computer Science, Columbia University, New York, USA

Abstract. The World Wide Web is becoming increasingly personalized as users provide more of their information on the Web. Thus, Web service functionality is becoming reliant on user profile information and context in order to provide user-specific data. In this paper, we discuss enhancements to SECE (Sense Everything, Control Everything), a platform for context-aware service composition based on user-defined rules. We have enhanced SECE to interpret ontology descriptions of services. With this enhancement, SECE can now create user-defined rules based on the ontology description of the service and interoperate within any service domain that has an ontology description. Additionally, it can use an ontology-based service discovery system like GloServ as its service discovery back-end in order to issue more complex queries for service discovery and composition. This paper discusses the design and implementation of these improvements.

Keywords: context-aware systems, ontologies, semantic web, rule-based systems, service discovery, service composition, web services

1 Introduction

In recent years, the World Wide Web has been advancing towards greater personalization. Services on the Web such as, social networking, e-commerce or search sites, store user information in order to profile the user and target specific products or ads of interest. Since web service functionality is increasingly relying on user information, a user's context is becoming more crucial towards creating a personalized set of services within the Web.

As these types of services proliferate, a framework is needed where multiple services can be discovered and composed for a particular user within a certain context. With this in mind, we have developed SECE (Sense Everything, Control Everything), a platform for context-aware service composition based on user-defined rules. The contributions to SECE are two-fold: a user-friendly rule language and the design and implementation of a context-aware service composition framework.

SECE differs from other rule-based systems in that it provides an interface for creating rules in natural English-like language commands. The main drawback

of rule-based systems is that the rule languages involve complex formulaic or XML descriptions. Lay people are not as inclined to use these systems as the learning curve for these languages may be steep. Thus, we have defined a formal rule language which resembles English. With a simplified English-like interface to creating rules, users will be more prone to incorporate rule-based systems into their lives, making context-aware computing a seamless part of everyday life.

Additionally, SECE provides a platform for context-aware service composition for a number of services, such as, presence, telecommunication, sensors and location-aware services. Users can subscribe to various services by formulating simple rules that create a composition of services. The rules trigger event-based service discovery and composition depending on the user's context, such as her location, time, and communication requests. Traditional rule-based systems are mostly designed to handle a single service domain. SECE, on the other hand, interacts with a few service domains. For more information on the SECE architecture and rule language, we encourage the readers to refer to the following paper [1].

In this paper, we discuss enhancements to both aspects of SECE: its rule language and back-end architecture. Whereas previously SECE had a hard-coded rule language for a limited number of event-based service domains, we have now improved SECE to use the Web Ontology Language (OWL) description of a service domain to dynamically create a rule language for that service domain. Additionally, SECE's architectural platform has been modified to integrate with a back-end ontology-based global service discovery system, GloServ, to access any type of service domain within the GloServ directory [2] [3]. GloServ classifies services in an ontology and provides ontology descriptions of different service domains. It also has an ontology-based query interface for service discovery and composition.

With these improvements, SECE can now be generalized to include all types of service domains, described in an ontology, as well as issue more complex ontology-based queries for service discovery and composition. Having the ability to adapt a rule language to new service domains makes SECE into a powerful front-end context-aware system. Additionally, by using GloServ as its back-end, SECE can now interoperate with any type of service that has an OWL description, broadening its scope drastically. We envision that SECE will enable services to seamlessly integrate into people's lives. A person can now create rules with ease and be notified of services at the right time and place. This will create a profound impact in how people interact with services. There will now be a closer connection between a person and services available, establishing a personalized network of services.

The organization of this paper is as follows: Section 2 describes current work in the field of context-aware computing and service composition; Section 3 gives an overview of the original SECE architecture and functionality; we discuss the enhancements to SECE and its implementation in Section 4; Section 5 discusses future work; finally, Section 6 summarizes the main contributions of this paper.

2 Related Work

Several solutions for user created services have been proposed; some of these solutions are compared to SECE in Figure 1. The second column indicates the user language for defining events and conditions that trigger action scripts. The third column indicates the language for action scripts. The fourth column shows the kinds of communication services that the users can use. The following columns show the types of information handled by the systems. CPL [4], LESS [5], SPL [6], VisuCom [7] and DiaSpec [8] are attempts to allow end users to create services, but they are all limited to controlling call routing. Also, CPL and LESS use XML and, hence, even simple services require long programs. Moreover, XML-based languages are difficult to read and write for non-technical end-users. DiaSpec is very low level. Writing a specification in DiaSpec and then developing a service using the generated framework is definitely not suitable for non-technical end users. The authors of DiaSpec extended [9] their initial work to support services beyond telephony, which include sensors and actuators. However, it is still only suitable for advanced developers. SPL is a scripting language which is suitable for end-users but only for telephony events. VisuCom has the same functionality as SPL, but allows users to create services visually via GUI components.

Systems	User rules	User actions	Communications	Time	Location	Presence	Sensors	Web services	Actuators
SECE	**Natural-language-like rules**	**Tcl scripts**	**Call, email, IM**	✔	**User & buddies**	**Rich**	✔	✔	✔
CPL	XML tree	Fixed XML actions	Call	✖	✖	✖	✖	✖	✖
LESS	XML tree	XML actions	Call	✔	✖	Basic	✖	✖	X10, vcr
SPL	script	Signaling actions	Call	✖	✖	✖	✖	✖	✖
VisuCom	Graphical UI	Signaling actions	Call	✖	✖	✖	✖	✖	✖
CybreMinder	Form based	Reminder	✖	✔	✔	✖	✔	✖	✖
Task.fm	Time rule	Reminder	✖	✔	✖	✖	✖	✖	✖
DiaSpec	Java	Java	✔✖	✖✔	✖✔	✖✔	✖✔	✖✔	✖✔

Fig. 1. Comparison to related work

CybreMinder [10] is a context-aware tool which allows users to setup email, SMS, print out and on-screen reminders based not only on time but also location and presence status of other users. It uses local sensors to detect a user's location. It does not take any actions, but rather displays reminders to the end user. Also it is not as powerful as scripting-based systems due to its form-based nature. Task.fm [11] is a similar SMS and email remainder system which uses natural language to describe time instants when email or SMS reminders will be sent. However, Task.fm only supports time-based rules and does not include information from sensors. This tool does not take actions other than reminding users via SMS, email or phone call.

Regarding composition of web services, SWORD [12] was one of the first prototypes. However, this offers a quite limited composition that is not automatic

and its scripting language is targeted at developers. Ezweb [13] is a graphical tool by which users can connect web services manually. However, this does not provide automatic web service discovery or a language for composing services. Moreover, service composition is not context-aware and proactive. Yahoo Pipes [14] is other graphical tool for web service composition. However, it presents the same limitations as Ezweb and its graphical interface is not really easy-to-use and intuitive, which makes it very difficult for non-technical users. We only found a prototype described in a research paper [15] that offers event-based web service composition. This means that service composition is triggered by events, such as changes in the user's context, instead of end users. However, this work does not provide any language or tool for specifying the web service compositions and events that trigger them. The authors seem to implement low-level compositions that may be personalized according to user preferences. Thus, this does not offer end users control of service composition. Moreover, this prototype seems not to be available in the Internet.

To the best of our knowledge, there is no implemented platform for allowing end users to compose services of different kind based on events. The current solutions are not proactive because the end-user is who triggers the composite services or only provides template-based compositions (i.e., the user is not who defines the compositions). There is neither a platform for event-based web service discovery. The composition tools that take user context into account, only consider a limited set of context. The graphical interfaces of the studied tools are quite limited and not flexible for non-technical users. The scripting languages provided by some tools are neither suitable for non-technical users and only support a limited set of context information. Moreover, none of the studied tools proactively discover web services based on the user preferences.

3 SECE

SECE is a rule-based context-aware system that connects services, that may have otherwise been disconnected, to create a personalized environment of services for a user. It has two fully-integrated components: user-defined rules in a natural English-like formal language and a supporting software architecture. Users are not required to continually interact with the system in order to query for or compose a set of services. They need to only define rules of what they want to accomplish and SECE does the rest by keeping track of the user's context, as well as information from external entities such as sensors, buddies, or social events in order to notify the user about a service. It accomplishes this by communicating with several third party applications and web services such as Google services (e.g., GMail, GContacts and GCalendar), Social Media services (e.g., Facebook or Twitter), VoIP proxy servers, presence servers, sensors and actuators. Figure 2 gives an overview of the overall SECE architecture and how it interacts with its environment. We will discuss these two components of SECE in this section.

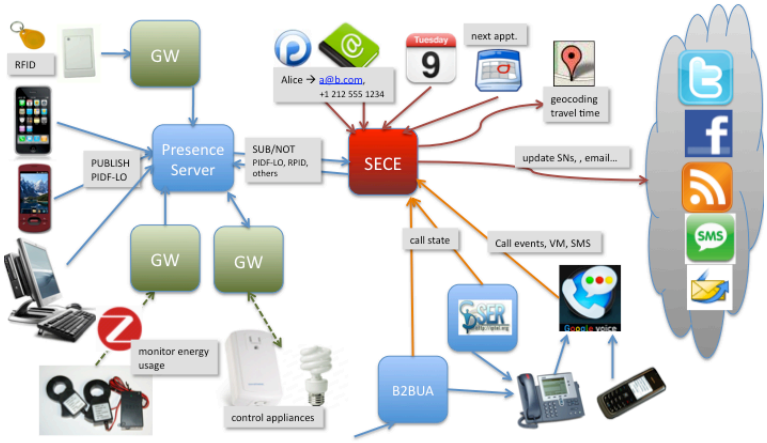


Fig. 2. SECE and its external components

3.1 SECE Architecture

As Figure 2 depicts, SECE is a web service that interacts with other web services, namely Google Services and Social Media services such as Twitter, Flickr and Facebook. The rules that are running on SECE and the rule actions that will potentially be executed determine the services with which SECE needs to interact. Thus, based on the kinds of rule that the user wishes to create and the actions that she wishes to compose, the user will need to configure the proper third-party services in her SECE account. Section 3.2 explains the SECE rules and actions, and their required services in more detail.

We are developing two services that tightly collaborate with SECE: the presence server and the VoIP proxy server. The presence server is built on the Mobicents Presence Service [16], which is compliant with SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) [17]. It is responsible for collecting and aggregating context from different sources and sending it to SECE. It accomplishes this by receiving presence publications from context sources that contain the latest information about a user and, in turn, notifying SECE of the context changes. In the SECE framework, context sources include user devices' presence applications and gateways that control sensor networks, energy consumption and user location via RFID. To use the presence service, the end user needs to create an account from the SECE website in order to obtain the SECE presence server's access information. Thus, the user can configure the SIMPLE-compliant presence applications (e.g. SIP Communicator and Pidgin) that run on her mobile devices or desktop computers to use the SECE presence server. In the future, the presence server will interact with the home gateway for obtaining information from sensor networks and changing the state of actuators.

The VoIP proxy server is a SIP (Session Initiation Protocol) Express Router (SER) [18] extended to interact with SECE for handling users' SIP communi-

cations. This server and SECE implement an efficient binary protocol that lets SER inform SECE of a SIP event and lets SECE notify SER of the action to take for this event. Basically, SER informs SECE of users' incoming and outgoing calls and instant messages (IM). If an event of this kind matches a rule, the rule is triggered and, therefore, decides to either forward, reject, or modify the call by invoking an action. Then, SECE will let SER know about the action to take. As the presence service, the user needs to create a SER account through her SECE account for using the VoIP proxy service. The user also needs to set her SIP-compliant multimedia applications to use the SECE VoIP proxy server as outbound/inbound SIP proxy. A first prototype of SECE has already been developed as a web service and is being tested by members of the Internet Real Time (IRT) group at Columbia University. For a more detailed description of the SECE architecture, we refer the readers to the following paper [1].

3.2 SECE rules

The SECE language supports five types of rules: time, calendar, location, context and communication. As a formal language, it states the valid combinations of keywords and variables for each kind of event and provides a set of commands, such as "sms", "email", "tweet" or "call". SECE rules and actions interact with different third party services based on their subscribed events and functions. Thus, SECE users need to learn about the services needed by the rule types and actions that they wish to use and configure their SECE accounts for such services. SECE will provide online documentation that gives users information about each rule's and action's syntax and required services. This documentation will also contain example rules to help users build rules for specific events and goals, and get familiarized with SECE rules. Figure 3 summarizes the required and optional services for the SECE rules and some actions.

Any SECE rule has the structure "*event* { *actions* }". *Event* defines the conditions that need to be satisfied to execute the *actions* that are delimited by braces. The SECE language for describing events is a formal language similar to English that has been designed to be easy to use and remember by end-users. This language is generated by an ANTLR grammar [19]. We use the Tcl language [20] as the syntax for the rule actions. This choice is due to Tcl's extensibility that allows adding new commands to its core with relative ease. Tcl provides a command that receives the name, arguments and code of a new command as parameters, constructs the corresponding Tcl command and incorporates it into the Tcl interpreter. Below, we describe the types of SECE rules and their involved services. In order to clearly display the structure of the rule and action language, the variables that are set by the user are highlighted in bold and the language keywords are italicized.

Time rules: Below are two types of rules: single time events and recurrent time events. The former starts with the keyword *on* and the latter starts with the keyword *every*. Both are fully-compliant with the Internet Calendar (ICal)

	GContact	GCalendar	GVoice	Twitter	PS	SER	GMail	GMaps	Flickr
RULE TYPES									
Time		Optional							
Calendar		Required							
Context					Optional				
Communication	Optional		Required for sms, voicemail			Required for SIP call, IM	Required for email		
Location								Required	
ACTIONS									
email	Optional						Optional		
tweet				Required					
flickr									Required
sms	Optional		Required						
call	Optional		Required for phone number			Required for SIP address			
status					Required				
forward	Optional					Required			
schedule		Required							
homelights					Required				

Fig. 3. Third party services of SECE rules and some actions

standard [21]. The *on*, *until*, *except* and *including* keywords are always followed by a date expression that can have different formats (e.g., "December 31, 2011", "31st day of December, 2011" and "12/31/2011") or can be an entry in the user's GCalendar. In the first example below, the user defined an entry named "Anne's birthday" in her 2011 GCalendar.

```
on Anne's birthday, 2011 at 12:00 in Europe/Zurich {
    sms Anne "Happy Birthday!!! John";
}
every week on WE at 6:00 PM from 1/1/11 until May 10, 2011
except 3rd WE of Feb, 2011 including first day of June, 2011 {
    email irt-list "reminder: weekly meeting today at 6:00 PM";
}
```

Calendar rules: These rules specify events that are defined in the user's GCalendar and always start with the keyword *when*. Thus, the user needs to configure his GCalendar in his SECE account before entering rules of this kind.

```
when 30 minutes before "weekly meeting" {
      email [event participants] "The weekly meeting will start in 30 minutes";
      if {{ ! my location within 3 miles of campus } {
          email [status bob.email] "I'm away" "Please, head the conference room and
          prepare everything for the weekly meeting. Not sure if I will be on time.";
      }
}
```

Location rules: A location rule starts with the keyword *me*, if it is about the user that is entering the rule, or an identifier of one of his friends such as a nickname, email and SIP address. Five types of location information are supported: geospatial coordinates, civic information, well-known places, user-specified places and user locations. Different location-related operators can be used, such as *near*, *within*, *in*, *outside of* or *moved*. Below we show a location rule using the *near* operator. *Within* means that the user is within a radius of the reference point. *Near* means the same but the radius is a default

distance that the user defines in his SECE account. *Outside of* and *in* means that the user is outside of and inside the reference point, which must be represented as a polygonal structure. We are working on a location database that allows users to predefine polygonal locations through a GMaps-based graphical interface. *Moved* means that the user moved the given distance from where he was located when the rule was entered or triggered for the last time.

```
Bob near "Columbia University" {
    if{ my status is idle } { call bob; }
}
```

Context rules: These specify the action to execute when some context information changes, such as presence or sensor state. These rules always start with the keyword *if*. If the rule is about the user that is entering the rule, this keyword if followed by *my*. Otherwise, the *if* keyword is followed by the friend's identifier. Below, we show an example of a context rule about a friend.

```
if Bob's status is available { alarm me; }
```

Communication rules: These specify the actions to execute in response to incoming, outgoing or missed communication requests. A request rule can start with the keyword *incoming*, *outgoing* or *missed*, followed by the type of event. The following rule is an example of incoming call handling.

```
incoming call to me.phone.work {
    if { [ my location is not office] } {
        autoanswer audio no_office.au;
        email me "[incoming caller] tried to reach you on your work phone at
        [incoming time]";
    }
}
```

4 Enhancing SECE Toward Ontology-based User-defined Rules for Automatic Service Discovery

As it stands, SECE has no way of automatically discovering a new type of service, generating a rule language for it and incorporating it in its system. The set of services that are supported in SECE are hard-coded. Thus, we have enhanced SECE to support ontology-based user-defined rules for automatic service discovery. The simple but illustrative example below emails the user whenever a new restaurant that satisfies the given conditions is found.

```
Any japanese restaurant that is cheaper than 20$ and whose location contains Manhattan {
    email me "new restaurant found" "Details: [event description]";
}
```

We have incorporated GloServ, an ontology-based service discovery system, within SECE's back-end architecture. GloServ provides an API whereby service ontology descriptions, for a number of domains, can be downloaded and queried for with an ontology query. GloServ uses the OWL DL ontology to describe its services. Thus, SECE can access these OWL specifications in order to dynamically define rules for the specific service domain. Users are made aware of these services by a front-end application to SECE that displays the discoverable services' descriptions. For each service domain, SECE will provide documentation on how to create rules. Currently, users will still need to learn how the rules are constructed, however, for the future, we plan on building a GUI that will use the ontology description to aid the user in constructing the rules. This section will describe the design and implementation of these enhancements.

4.1 SECE Architecture

Design Figure 4 outlines the main interactions between SECE, GloServ, front-end applications and web services. We assume that end users are connected to front-end applications, which detatches users from SECE and offers more flexibility. Front-end applications retrieve user data from SECE and allow users to create their rules probably by means of more fancy graphical interfaces, suggestions and user preferences, for example. From the moment at which a web service rule is entered in SECE on, SECE will periodically communicate with GloServ for discovering the web services that match the rule. A GloServ request specifies the web service of interest as a SPARQL query [22] and matched services' profiles, if any, are sent to SECE into a GloServ response. If a new web service matches a rule, SECE executes the rule's body.

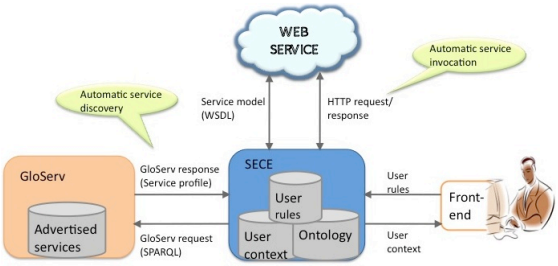


Fig. 4. SECE, GloServ, front-end applications and web services

SECE has a layered architecture, as Figure 5 shows. For details of each of the components, we encourage the reader to refer to the original SECE paper [1]. We will discuss the components that have been added to the enhanced SECE architecture in this section.

The new components that have been added to the SECE architecture are: 1) *WBRL* rule, which implements the web service rules; 2) Jena Ontology Model,

which contains the necessary ontologies' schemes; 3) GloServ Context Mediator, which periodically pulls GloServ for checking out new web services of interest. .

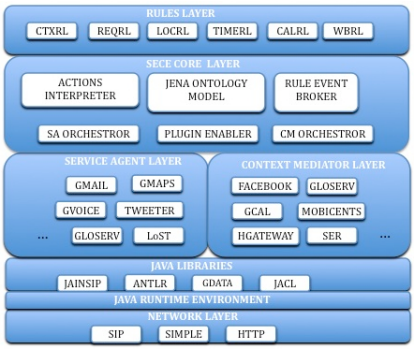


Fig. 5. SECE architecture

Implementation SECE stores the OWL specifications of web services in an ontology database that is built upon the Jena Framework [23]. When a web service rule is entered into SECE, it has to go through the following steps: 1) parse the rule (i.e., syntactic checking); 2) verify that the described kind of web service exists (i.e., semantic checking); 3) subscribe to the described web service event; and 4) take the rule's actions whenever this event occurs. Figure 6 outlines the main interactions for creating a web service subscription.

The SECE core coordinates the software components in SECE. First, the SECE parser checks that the input rule is consistent with the SECE language, which is generated by an ANTLR grammar [19]. As a result, the parser creates a *WSRule* object that encapsulates information about the rule, namely a web service event and the actions that will be taken if this event occurs. The web service event is defined by the service name and optionally a set of property constraints in the form of *(propertyName, operator, value)*. If the rule parsing is successful, the SECE core verifies that the rule's web service description corresponds to a web service's ontology. To do it, this interacts with the SECE Ontology Model (i.e., *SECEOntModel* in Figure 6). The SECE Ontology Model encapsulates the Jena database that contains the web services' ontologies and provides convenient functions for searching and retrieving information about them. A web service description is semantically correct if there exists a web service's ontology that describes a service that is named as the described web service and can be associated with the described properties and constraints. Thus, SECE will ask the SECE Ontology Model for the namespace URI of the web service and its properties. If this web service does not correspond to any ontology, the SECE ontology Model returns null values. This means that the rule's web service event is semantically incorrect, which results in aborting rule creation and warning

the user. Otherwise, the rule's web service event is semantically correct and the SECE core proceeds to create the corresponding subscription (i.e., *WSSubs* in Figure 6).

The SECE core then retrieves an event monitor from the Event Monitor Broker (*OntEM* and *EMBroker* in Figure 6). An event monitor is the agent that watches a particular service and generates an event whenever a new instance of this service is discovered. The Event Monitor Broker maintains a list of the event monitors that are actually monitoring a web service. Thus, if an event monitor for the web service event already exists, the Event Monitor Broker returns it. Otherwise, the Event Monitor Broker creates a new one, appends it to the list of monitors and returns it. Then, the SECE core associates the event subscription with the event monitor and starts the subscription.

Starting and pausing an event subscription makes it subscribe and unsubscribe to the associated event monitor, respectively. When an event monitor receives a subscription request and there are no other subscribers, it creates the corresponding SPARQL query that describes the web service event. This also starts up a recursive timer to query the GloServ Context Mediator (i.e., *GloServCM* in Figure 6) at fixed intervals with the SPARQL query. If this query results in any matched service, the event monitor creates an OntEvent object that describes the discovered service and notifies the subscriber of this event. Note that the outbound messages between GloServCM and GloServ are omitted in Figure 6 because of lack of space. When an event monitor is associated with more than one subscriber, the SPARQL query represents the least restrictive subscription. When a web service matches this subscription, the event monitor checks out whether the service matches any of the other subscriptions. Figure 6 only shows this check on the web service subscription *wss* through the *matchedServ* method. Furthermore, the event monitor maintains a cache of discovered events. When a new subscription is created, this cache is checked out and the matching web services are notified.

4.2 SECE Ontology-based Sublanguage

SECE provides a simple and generic ontology-based language for end-users to define web service rules. In line with SECE's philosophy, this language looks like natural English and is easy to learn. Its basic structure is "any *service* whose *prop rel value*" given that *service* is a web service class, *prop* is one of this service class' properties and *rel* and *value* represent a restriction on the property. *Rel* is a relational operator that depends on the property's type: *contains* and *is* for strings and $=, <, >, \leq$ and $\geq$ for numbers.

Multiple property constraints can be added by the *and* and *or* boolean operators as for example "any shopping offer whose type contains "ski boots" and whose price is cheaper than 150$". Equality on numeric properties can be expressed by the verb *has* followed by a number and the property name as in "any happy hour and inexpensive bar that has 20 free seats". Users can place property values before the class name when the property works as adjective. In the previous example, the *bar* class has the boolean properties *happyHour* and

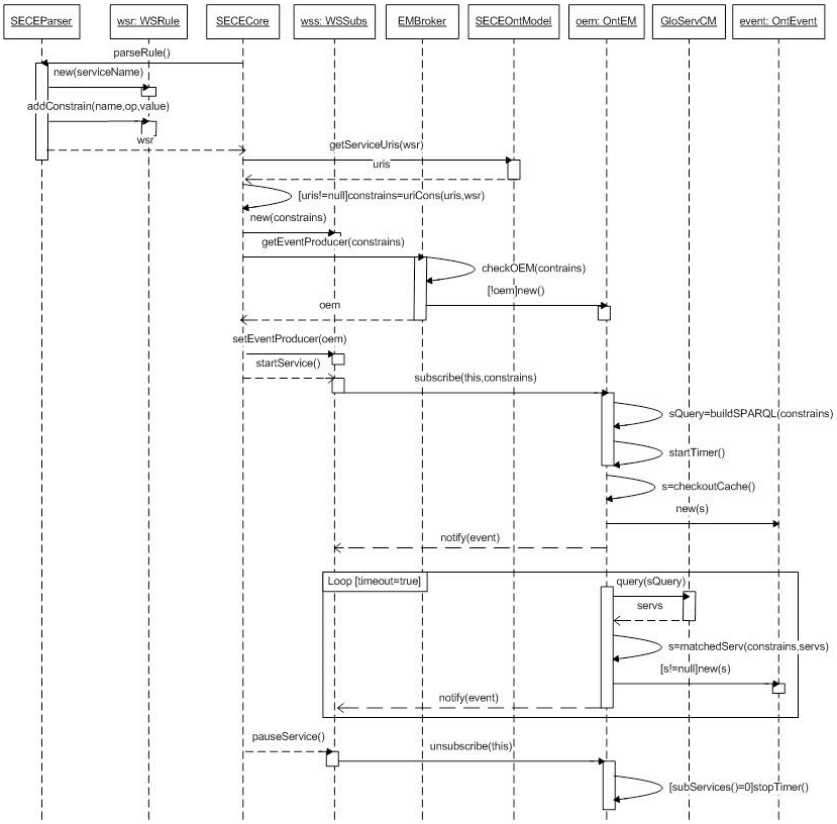


Fig. 6. Sequence diagram from entering a web service rule to querying GloServ

inexpensive. Boolean constraints can also be expressed by the operators *that has (no)* and *that is (not)* as in "any restaurant that has delivery", "any restaurant that is open 24 hours" and "any cultural exhibition that is free and is not crowded".

Boolean constraints can be applied to class properties or types, which depends on the ontology's structure and is transparent for end-users. An example of boolean property is the above-mentioned *delivery* property whose domain is the *restaurant* class. Boolean constraints on class types restrict inherited types as for example "any restaurant that is southamerican" subscribes to restaurants that are subclasses of the *southamericanRestaurant* class.

5 Future Work: Event-Based Context-aware Web Service Composition System

Integrating web service rules into SECE brings out exciting possibilities in the Semantic Web. This permits end-users to define and personalize context-aware

web service discovery, invocation and composition based on a variety of events. SECE provides a set of actions for users to build up compositions. Some actions interact with web services, such as *tweet*, *publish* and *email*; other actions send protocol-specific requests, such as *call* (i.e., SIP INVITE); and others are supportive routines. The set of web services with which SECE communicate is static and the communication is hard-coded.

Therefore, SECE compositions are static in the sense that, once a composition is created, it will not change. We are planning to incorporate dynamic compositions to SECE through automatic web service discovery and composition. Two new SECE actions will add this functionality: *find* and *plan* for discovery and composition, respectively. An example rule is shown below, in which the *plan* and *find* commands are pseudo-code because they have not been implemented yet. In this example, whenever a new flight is found, other web services are discovered (i.e., hostels, car rentals and restaurants) and composed (i.e., trip planning). Note that the *plan* action could invoke *find* to discovery web services that are necessary for the composition. As the discovered web services and the communication with them can be different each time the composition is executed, we say that this composition is dynamic.

Any **domestic flight** that is cheaper than **200$** and whose **date** is after **June 1, 2011** {
    p=plan flight with hostel and car rental;
    r=find good restaurants according to $p;
    email **me "new plan found" "Details: $p $r"**;
    sms **me "New Plan discovered. See email inbox for details!"**;
}

With these two new actions, SECE could perform semantic web service discovery and composition that does not need user interaction to be executed; it is automatically triggered by events. In addition, this would also allow combining static and dynamic composition. For example, the rule above provides dynamic composition through the *plan* and *find* actions and static composition through the *email* and *sms* actions. As the Semantic Web is not widely adopted yet, hybrids platforms like SECE are necessary to offer users flexible and powerful composition tools. Table 1 indicates the types of composition that SECE already supports (white column) and will support in the future (gray columns). Rows define the events that trigger the compositions and columns the types of web service communication in the compositions.

Table 1. Types of SECE composition

	Semantic service communication	**Hard-coded service communication**	**Both kinds of communication**
Web service events	Dynamic composition triggered by discovered web services	Static composition triggered by discovered web services *(current contribution)*	Mixed composition triggered by discovered web services
Other events	Dynamic composition triggered by real-world events	Static composition triggered by real-world events *(typical SECE composition)*	Mixed composition triggered by real-world events

For dynamic compositions, SECE will interact with web services automatically, by retrieving their models and, according to their WSDL specifications, constructing HTTP requests.

6 Conclusions

The Semantic Web is investing a great deal of effort in developing standards for providing automatic web service discovery and composition. Although many authors have been interested in this exciting topic in the last decade, complete solutions do not yet exist. Thus, there is a strong need for general-purpose platforms for automatic web service discovery and composition that also provide intuitive and user-friendly interfaces that do not require engineering or technical skills. Besides template-based composition, end users should be able to orchestrate service composition.

To face all these needs, we present a context-aware, event-based platform for service discovery and composition by integrating two existing solutions: SECE and GloServ. SECE is a user-centric, context-aware platform for service composition that provides a natural-English-like language for creating event-based rules. GloServ is a scalable network for web service discovery. We implemented the communication between GloServ and SECE. We extended SECE with an ontology database that stores the web services' schemes that come from GloServ. We also developed a SECE sublanguage to subscribe to web services, which allows subscribing to web service discovery events by creating rules in a user-friendly language that looks like natural English. This makes SECE suitable for non-technical users. SECE also allows creating service compositions that can be triggered by discovered web services and real-world events such as context changes, location, or time. Modeling SECE rules ontologically can provide front-ends with the means of understanding and learning new SECE rules automatically. Thus, the combination of SECE and GloServ paves the way for future extensions.

Acknowledgments. Victoria Beltran was supported by the Spanish Government through the CICYT project TIC2009-11453 and the FPU grant AP2006-02846.

References

1. O. Boyaci, V. Beltran, and H. Schulzrinne, "Bridging communications and the physical world: Sense everything, control everything," in *Proceedings on the IEEE Globecom (UbiCoNet Workshop)*, December 2010.
2. K. Arabshian and H. Schulzrinne, "An ontology-based hierarchical peer-to-peer global service discovery system," *Journal of Ubiquitous Computing and Intelligence*, vol. 1, no. 2, p. 133.
3. K. Arabshian, C. Dickmann, and H. Schulzrinne, "Service composition in an ontology-based global service discovery system," tech. rep., Columbia University, New York, NY, September 2007.

4. J. Rosenberg, J. Lennox, and H. Schulzrinne, "Programming Internet telephony services," *Internet Computing, IEEE*, vol. 3, pp. 63–72, May/Jun 1999.
5. Xiaotao Wu and Henning Schulzrinne, "Programmable End System Services Using SIP," *Conference Record of the International Conference on Communications (ICC)*, May 2003.
6. L. Burgy, C. Consel, F. Latry, J. Lawall, N. Palix, and L. Reveillere, "Language Technology for Internet-Telephony Service Creation," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 4, pp. 1795–1800, June 2006.
7. F. Latry, J. Mercadal, and C. Consel, "Staging telephony service creation: a language approach," in *IPTComm '07: Proceedings of the 1st international conference on principles, systems and applications of IP telecommunications*, (New York, NY, USA), pp. 99–110, ACM, 2007.
8. W. Jouve, N. Palix, C. Consel, and P. Kadionik, "A SIP-Based Programming Framework for Advanced Telephony Applications," in *IPTComm* (H. Schulzrinne, R. State, and S. Niccolini, eds.), vol. 5310 of *Lecture Notes in Computer Science*, pp. 1–20, Springer, 2008.
9. D. Cassou, B. Bertran, N. Loriant, and C. Consel, "A generative programming approach to developing pervasive computing systems," in *GPCE '09: Proceedings of the eighth international conference on Generative programming and component engineering*, (New York, NY, USA), pp. 137–146, ACM, 2009.
10. A. K. Dey and G. D. Abowd, "CybreMinder: A Context-Aware System for Supporting Reminders," in *HUC '00: Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, (London, UK), pp. 172–186, Springer-Verlag, 2000.
11. "task.fm Free SMS and Email Reminders." `http://task.fm`.
12. S. Ponnekanti and A. Fox, "Sword: A developer toolkit for web service composition," in *Proc. of the Eleventh International World Wide Web Conference, Honolulu, HI*, 2002.
13. J. Soriano, D. Lizcano, J. Hierro, M. Reyes, C. Schroth, and T. Janner, "Enhancing user-service interaction through a global user-centric approach to SOA," in *Networking and Services, 2008. ICNS 2008. Fourth International Conference on*, pp. 194–203, IEEE, 2008.
14. "Yahoo pipes." `http://pipes.yahoo.com/pipes/`.
15. R. Kazhamiakin, P. Bertoli, M. Paolucci, M. Pistore, and M. Wagner, "Having Services "YourWay!": Towards User-Centric Composition of Mobile Services," *Lecture Notes in Computer Science*, vol. 5468/2009, pp. 94–106, 2009.
16. "Mobicents." `http://www.mobicents.org/`.
17. "SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE)." `http://datatracker.ietf.org/wg/simple/charter/`.
18. "About SIP Express Router." `http://www.iptel.org/ser/`.
19. T. Parr, *The Definitive ANTLR Reference: Building Domain-Specific Languages*. Pragmatic Bookshelf, 2007.
20. J. K. Ousterhout and K. Jones, *Tcl and the Tk Toolkit*. Upper Saddle River, NJ: Addison-Wesley, 2nd ed., 2009.
21. B. Desruisseaux, "Internet Calendaring and Scheduling Core Object Specification (iCalendar)." RFC 5545 (Proposed Standard), Sept. 2009. Updated by RFC 5546.
22. W3C, "SPARQL Query Language for RDF." Website, January 2008. `http://www.w3.org/TR/rdf-sparql-query/`.
23. "Jena - A Semantic Web Framework for Java." Website. `http://jena.sourceforge.net/index.html`.

Random Indexing for Finding Similar Nodes within Large RDF graphs

Danica Damljanovic[1], Johann Petrak[2], Mihai Lupu[3], Hamish Cunningham[1], Mats Carlsson[4], Gunnar Engstrom[4], and Bo Andersson[4]

[1] Department of Computer Science, University of Sheffield, United Kingdom
d.damljanovic@dcs.shef.ac.uk, h.cunningham@dcs.shef.ac.uk
[2] Austrian Reseach Institute for Artificial Intelligence, Vienna, Austria
johann.petrak@ofai.at
[3] Information Retrieval Facility (IRF), Vienna, Austria
m.lupu@ir-facility.org
[4] AstraZeneca, Lund, Sweden
Mats.Carlsson, Gunnar.Engstrom, Bo.H.Andersson@astrazeneca.com

Abstract. In this paper, we propose an approach for searching large RDF graphs, using advanced vector space models, and in particular, Random Indexing (RI). We first generate documents from an RDF Graph, and then index them using RI in order to generate a semantic index, which is then used to find similarities between URIs, literals, and RDF subgraphs. We have experimented with large RDF graphs in the domain of life sciences and engaged the domain experts in two stages: firstly, to generate a set of keywords of interest to them, and secondly to judge on the quality of the output of the Random Indexing method, which generated a set of similar terms (literals and URIs) for each keyword of interest.

Key words: random indexing, vectors space models, information retrieval, RDF graphs, ontologies

1 Introduction

Recent years have seen a massive increase of highly structured data being made available in the form of RDF triple representations. Both legacy data and new data have been made available in RDF triple format and this representation has also made it worthwhile and feasible to create mappings between RDF data that originates from different legacy sources, leading to potentially very large RDF repositories. Initiatives such as Linked Open Data[5] are working on creation, publication and interlinking of huge RDF graphs.

Traditionally, RDF spaces are being searched using an RDF query language such as SeRQL [2] or SPARQL [15]. These languages allow the formulation of fine-grained queries by their ability to match whole graphs and to create complex conditions on the variables to be bound in the query. This level of complexity

[5] http://linkeddata.org/

and flexibility is very useful in many situations, especially when the query is created automatically in the context of an application. However, for end-users who want to explore the knowledge represented in an RDF store, this level of detail is often more of a hindrance: querying the repository is not possible without a detailed knowledge of its structure and the names and semantics of all the properties and classes involved. This is especially the case for large and unknown data structures which may have thousands of classes and properties, for example Linked Life Data[6] (5 billion statements), or FactForge[7] (2 billion statements).

In this paper we investigate whether advanced Information Retrieval (IR) methods can bring a new dimension to the task of searching huge RDF graphs. We propose a complementary approach based on word space model, more concretely Random Indexing (RI) [14], for building a *semantic index* for a large RDF graph. Traditionally, a *semantic index* captures the similarity of *terms* based on their contextual distribution in a large document collection, and the similarity between *documents* based on the similarities of the terms contained within. By creating a semantic index for an RDF graph, we are able to determine contextual similarities between graph nodes (e.g., URIs and literals) and based on these, between arbitrary subgraphs. These similarities can be used for finding a ranked list of *similar* URIs/literals for any given input term (a literal or a URI), which can then be used for exploring the repository or enriching SPARQL queries.

We evaluate our approach on subsets of the Linked Life Data (LLD) repository – a large integrated repository which contains 5 billion RDF statements from various sources covering the biomedical domain, including UniProt[8], PubMed[9], EntrezGene[10] and many more[11]. Our evaluation is based on human judgment by clinical research scientists (from AstraZeneca pharmaceutical company) who were involved in two stages: firstly, to generate a set of keywords of interest to them, and secondly to judge on the quality of the output of the Random Indexing method, which generated a set of similar terms (literals and URIs) for each topic of interest.

2 Related work

A considerable amount of work has been done in the area of using Information Retrieval methods for the task of selecting and retrieving RDF triples. However, most of these approaches do not take advantage of the latent semantics included in an RDF Graph, as their primary intention is finding the RDF files on the Web relevant to the given keyword and/or a URI. These systems are semantic search engines such as Swoogle [9] or Sindice ([18]). They collect the Semantic

[6] `www.linkedlifedata.com`
[7] `http://factforge.net`
[8] `www.uniprot.org/`
[9] `http://www.ncbi.nlm.nih.gov/PubMed/`
[10] `www.ncbi.nlm.nih.gov/sites/entrez?db=gene`
[11] see the full list at: `www.linkedlifedata.com/sources`

Web resources from the Web and then index the keywords and URIs against the RDF files containing those keywords and URIs, using the inverted index scheme. These search engines use traditional weighting mechanisms such as TF-IDF, and in [11] the authors introduce the *ReConRank* algorithm, which adapts the well-known PageRank algorithm to Semantic Web data. This method ranks the nodes in a topical subgraph that is selected based on keyword matching from the RDF files. In other words, it ranks the results of a query based on the RDF links in the results. The subgraph that the algorithm identifies includes both the subject nodes related to the query, and also the context of the subject nodes (i.e. the provenances or sources of the subjects), in order to improve the quality of ranking.

In comparison to these approaches we use the neighbouring nodes as semantic context for each node in an RDF graph. The nodes and their contexts are used as virtual documents for Random Indexing.

In [16], the authors describe an approach for generating a virtual document for each URI reference in an RDF triple store (or, equivalently, each node in an RDF graph). The virtual document contains the local name and labels of the URI reference, other associated literals such as those in *rdfs:comment*, and the names of neighbouring nodes in the RDF graph. These virtual documents are then used for ontology matching and also for generating object recommendations for users of Falcons [3]. In comparison to our approach, their neighbouring operations involve only one-step neighbours without including properties. Our approach includes properties, and parts of the TBox, and also can operate on an arbitrarily large graph of neighbouring nodes.

Finally, to the best of our knowledge, none of the similar approaches investigate the usage of methods that can discover latent semantics, such as Random Indexing.

3 Semantic Index

Latent Semantic Analysis (LSA) [8] is one of the pioneer methods which has been used for finding synonyms. The assumption behind this and other statistical semantics methods is that words which appear in the similar context (with the same set of other words) are synonyms. Synonyms tend not to co-occur with one another directly, so indirect inference is required to draw associations between words which are used to express the same idea [4]. This method has been shown to approximate human performance in many cognitive tasks such as the Test of English as a Foreign Language (TOEFL) synonym test, the grading of content-based essays and the categorisation of groups of concepts (see [4]). However, one problem with this method is scalability: it starts by generating a $term*document$ matrix which grows with the number of terms and the number of documents and will thus become very large for large corpora. For finding the final LSA model, Singular Value Decomposition (SVD) and subsequent dimensionality reduction is commonly used. This technique requires the factorization of the term-document matrix which is computationally costly and does not scale well. Also, calculating

the LSA model is not easily end efficiently doable in an incremental or out-of memory fashion. The Random Indexing (RI) method [17] circumvents these problems by avoiding the need of matrix factorization in the first place.

RI can be seen as an approximation to LSA which is shown to be able to reach similar results (see [14] and [5]). RI can be incrementally updated and also, the $term * document$ matrix does not have to be loaded in memory at once – loading one row at the time is enough for computing context vectors. Instead of starting with the full term-document matrix and then reducing the dimensionality, RI starts by creating almost orthogonal random vectors (index vectors) for each document. This random vector is created by setting a certain number of randomly selected dimensions to either +1 or -1. Each term is represented by a vector (term vector) which is a combination of all index vectors of the document in which it appears. For an object consisting of multiple terms (e.g. a document or a search query with several terms), the vector of the object is the combination of the term vectors of its terms.

Random Indexing relies on the Johnson-Lindenstrauss lemma:

Lemma 1. *Given* $0 < \epsilon < 1$*, a set* X *of* m *points in* R^N*, and a number* $n > n_0 = O(\frac{log(m)}{\epsilon^2})$*, there exists a mapping* $f : R^N \rightarrow R^n$ *such that* $(1-\epsilon)||u-v|| \leq ||f(u) - f(v)|| \leq (1+\epsilon)||u-v||$*, for all* $u, v \in X$*.*

and particularly on the proof provided by Johnson and Lindenstrauss in their 1984 article [13], where they show that if one chooses at random a rank n orthogonal projection, then, with positive probability, the projection restricted to X will satisfy the condition in the Lemma. RI relies on the observation that, in a high dimensional space, a random set of vectors is always almost orthogonal.

In order to apply RI to an RDF graph we first generate a set of documents which represent this graph, by generating one *virtual document* for each URI in the graph (Section 3.1). Then, we generate a semantic index from the virtual documents (Section 3.2). This semantic index is then being searched in order to retrieve similar literals/URIs (Section 3.3).

3.1 Generating virtual documents

The task of deriving a set of documents from a huge RDF graph starts with generating a *representative subgraph* for each URI of interest. We shall refer to such an URI as a *representative URI*.

A representative subgraph represents the context of a URI i.e. the set of other URIs and literals directly or indirectly connected to that URI. For a representative URI S, the representative subgraph of order N is a set of all paths of triples $(S, P_1, O_1; O_1, P_2, O_2; \cdots; O_{N-1}, P_N, O_N)$. If O_N is not a literal we also include all triples O_N, P_{N+1}, L_J where L_J is a literal. In other words, we apply the Breadth-First-Search starting with the representative node, and extend this to the Depth Search which is defined by N. In addition, we include or exclude certain parts of the TBox: direct classes for instances are excluded

($P_N! = rdf : type$), while other annotation properties such as $rdfs : label$ are included. In the experiments reported in this paper, the representative subgraphs are of order 1 ($N = 1$).

We create *virtual documents* by including all paths from representative subgraphs where:

- all URIs of nodes or appearing inside literals are included unchanged;
- for literals we remove punctuation and stop words, and then lowercase the text; we also remove number literals, gene and protein sequences, complex names, and HTML tags.

3.2 Generating semantic index

There are several parameters which can influence the process of generating semantic index, or vectors using the RI method:

- **Seed length** Number of +1 and -1 entries in a sparse random vector.
- **Dimensionality** Dimension of the semantic vector space – predefined number of dimensions to use for the sparse random vectors.
- **Minimum term frequency** Minimum frequency of a term to get included in the index.

Our experiments study how variations of these parameters influence the quality of the results and how sensitive the method is to that variation.

3.3 Search

Once the semantic index has been created, it can be used to find similarities between URIs, literals, and RDF subgraphs. We use the cosine function to calculate the similarity between the input term (literal or URI) vector and the existing vectors in the generated vector space model. We can perform the following kinds of searches:

1. *finding similarities between two terms*: given a keyword, find similar literals and URIs; this can be used in several ways for example for refinement of SPARQL queries (see [7]); also, it can be used as an alternative way of browsing and finding URIs or literals related to a topic of interest (expressed through a keyword or a set of keywords)
2. *finding documents related to a specific term*: this task would be useful for suggesting a set of representative URIs related to a given keyword.
3. *finding documents related to a document*: this task would be useful for suggesting a set of representative URIs related to a set of URIs.
4. *finding terms related to the specific documents*: this can be used for describing a representative URIs through a set of literals and URIs.

While in the context of large RDF graphs such as LLD, we find all of these searches useful, in the experiments we present next, we focus on term-term search (Item 1) only. As the LLD dataset covers the life sciences domain, we have conducted a study with the clinicians from AstraZeneca, who are domain experts and understand the knowledge available in this large dataset.

4 Experiments

Our goal in using the Random Indexing method is to investigate whether it can offer an alternative way of searching large RDF spaces, by suggesting literals or URIs which are similar to the topic of interest. We conduct an evaluation experiment with clinical research scientists from AstraZeneca, with the aim to assess this.

4.1 Dataset

Linked Life Data is a dataset covering the life sciences domain, and the latest version 0.6 contains 5,052,047,661 statements in total (for a comparison, one year ago it contained 4,179,999,703 statements). Advanced IR methods based on Vector Space Model (VSM) are computationally expensive, and therefore, before we apply the Random Indexing method on the whole dataset, we evaluate it on two smaller subsets of LLD.

We have generated the two subsets as follows. For 1528 seed URIs (the URIs representing all MEDLINE articles from December 2009) we retrieve neighbouring subgraphs (of order 1) recursively until we reach certain predefined limit of statements, and we refer to these as LLD1 and LLD2. Table 1 shows the sizes of LLD1 and LLD2.

	LLD 1	LLD2
number of statements	595798	4573668
number of virtual documents	64644	473742
number of terms	417753	1713349

Table 1. Sizes of LLD1 and LLD2 datasets

4.2 Evaluation measures

In order to calculate the correctness of the retrieved terms, there are standard Information Retrieval measures such as *precision*, *recall* and *Mean Average Precision (MAP)*. Precision is defined as the number of relevant documents retrieved divided by the total number of documents retrieved and is usually calculated for certain number of retrieved documents (e.g., Precision@10, Precision@20). Recall is the number of relevant documents retrieved divided by the total number of existing relevant documents (which should have been retrieved).

Mean Average Precision (MAP) is by far one of the most popular measures in IR evaluation because, for each system and set of topics, it provides a single value to measure its performance [6]. Average Precision (AP) is computed for each topic by first calculating precision for each relevant document that is retrieved and then averaging these values. Mean Average Precision is then the

mean of these values for all keywords. Furthermore, by the nature of the averaging process, MAP is more sensitive to ranking than *precision* at a specific point, favouring systems which return more relevant documents at the top of the list than at the bottom, whereas *precision* does not make this distinction as long as the results are within the cut-off range.

As our task is to retrieve most relevant literals and URIs first, we used MAP@10. Recall is extremely difficult to measure due to the number of terms in our datasets (see Table 1). In addition, our task is to help domain experts explore large RDF graphs, which is similar to Web search in the sense that there is a vast amount of terms to be searched through, and also a significant number which is relevant for each input term. Hence, for these kinds of tasks, users care more about precision than about recall. Indeed, they care most about the top ranked results, which is exactly what is captured by MAP.

Relevance of retrieved terms was evaluated by two clinical research scientists. All scientists looked at all retrieved terms. *Relevant* were considered only those terms which *both* scientists marked as relevant. In order to measure agreement between scientists on this particular task, we measured the Inter Annotator Agreement (IAA) between the two clinicians based on the words which both of them marked as relevant/irrelevant.

IAA has been used mainly in the classification tasks, where two or more annotators are given a set of instances and are asked to classify those instances into some pre-defined categories. The two commonly used IAA measures are *observed agreement* and *Kappa (κ)* [12].

Observed agreement is the portion of the instances on which the annotators agree. For our case, with the two annotators and two categories (relevant and irrelevant), it is defined as

$$A_o = \frac{a+d}{a+b+c+d} \tag{1}$$

where a refers to the number of terms *both annotators agreed as relevant*, d refers to the number of terms *both agreed as irrelevant*, b refers to the number of terms *annotator 1 marked as relevant, and annotator 2 as irrelevant*, c refer to the number of terms *annotator 1 marked as irrelevant, and annotator 2 as relevant.*

A certain amount of agreement is expected by chance which is not captured by the observed agreement. The Kappa measure is a chance-corrected agreement. **Kappa** is defined as the observed agreements A_o minus the agreement expected by chance A_e and is normalized as a number between -1 and 1.

$$k = \frac{A_o - A_e}{1 - A_e} \tag{2}$$

$k = 1$ means perfect agreement, $k = 0$ means the agreement is equal to chance, $k = -1$ means 'perfect' disagreement.

There are two different methods for estimating A_e: in **Cohen's Kappa**, each annotator has a personal distribution, based on his distribution of categories. In **Siegel & Castellans Kappa**, there is one distribution for all annotators,

derived from the total proportion of categories assigned to all annotators (see [10] for more details and for the comparison of the two). We used Cohen's Kappa in our experiments.

4.3 Experimental setup

We have performed our experiment through the following steps:

1. **Extracting topics of interest** represented as query terms which are present in both LLD1 and LLD2. In order to avoid exposing the scientists to learning SPARQL, we have formed a team of one computer scientist and one clinical research scientist. The computer scientist was executing the SPARQL query and browsing through the links and URIs, while the clinical research scientist was only looking at the abstracts which the computer scientist selected. As a result, we obtained 18 keywords which all appeared in both LLD1 and LLD2 datasets. We split this set into two halves as shown in Table 2 , and then perform the following two steps in two iterations: first, *Group 1* is used for *training* the model, and *Group 2* for *testing* it. In the second iteration, the two sets are swapped.

Group 1	**Group 2**
acetylcholinesterase	Posttraumatic Stress Disorder
synergistic effect	trial
cholinergic signaling	bladder cancer
PTSD	Adverse events
antagonist	trauma
efficacy	antioxidant
clinical trial	magnesium
cognitive	cystectomy
lung	5-HT receptors

Table 2. Topics of interest divided into two groups for training/testing the Random Indexing method

2. **Training the model**: we generated RI models for several variations of the following RI parameters for both LLD1 and LLD2:
   - vector dimension: 500, 1000, 1500, 1800, 2500
   - seed length: 10, 50, 100, 300, 500, 1000
   - term frequency: 1, 2, 5, 8, 10

   This resulted in 290 runs (145 per dataset[12]). We then searched for similar words for each topic of interest from the *training* set, and presented them to clinicians who accessed the relevance. The combinations for parameters which lead to the best results (measured through MAP) were considered as the optimal setting for testing the method in the next step.
3. **Testing the model**: for the models generated using the optimal parameters retrieved in the previous step, we retrieved 10 similar words for each topic

[12] 5 runs are missing from this count, corresponding to the situation where the seed size is 1000, and the vector dimensionality is 500, which is impossible

of interest from the *testing* set and calculated MAP. The correctness of the retrieved terms was assessed by clinical research scientists to whom we gave the terms in the form of a survey (see below).

Human assessment The retrieved keywords for each topic of interest in both *training* and *testing* sets were assessed by humans. We merged the results from all searches into one pool, and gave this list to the scientists in the form of a survey. When the similar term was a URI, we have extracted the label from LLD and showed it in brackets. This is to ensure that the scientists can concentrate on meaning of these rather than looking and searching LLD in order to find the label. An example task looked similar to this:

```
----------------------------------------------------------------
Is 'trauma' related to (delete URIs/words which are not related):
----------------------------------------------------------------
arteriopathy
back-projection
barotraumas
gunshot
http://linkedlifedata.com/resource/umls/id/C0003048 (Animal
                                                     Experimentation)
http://linkedlifedata.com/resource/umls/id/C0004601 (Back Injuries)
http://linkedlifedata.com/resource/umls/id/C0005604 (Birth trauma)
............
```

The most difficult task when designing this experiment was to define the meaning of *relevant*. Relevant, in this context, is any word related to the given keyword. This is a quite broad definition, which has, as it has been reported by clinical research scientists who were involved in this experiment, posed a number of difficulties due to many different levels of relevance. One of them stated that it would not be easy to repeat the same tasks and mark the same words as relevant if they had to repeat the same task again. We consider those that are not deleted as relevant. Only those words which have been marked as relevant twice (by two different clinicians) were eventually used when evaluating our results.

4.4 Results

In this section we first look into the results of training the model and finding the best parameters with two separate groups independently. Then, we look at the results of testing the RI method using these best parameters.

Training the model We expect to see variations of MAP, for different values of dimensionality, seed length, and minimum term frequency parameters. Our goal is to find the combination of parameters for which MAP is highest, so as to use those in the testing phase.

Figure 1 shows the distribution of MAP across all cases, and for each group used for training. It seems that the keywords from *Group 1* were more challenging for the method, as MAP values are much lower on average. However, as we can

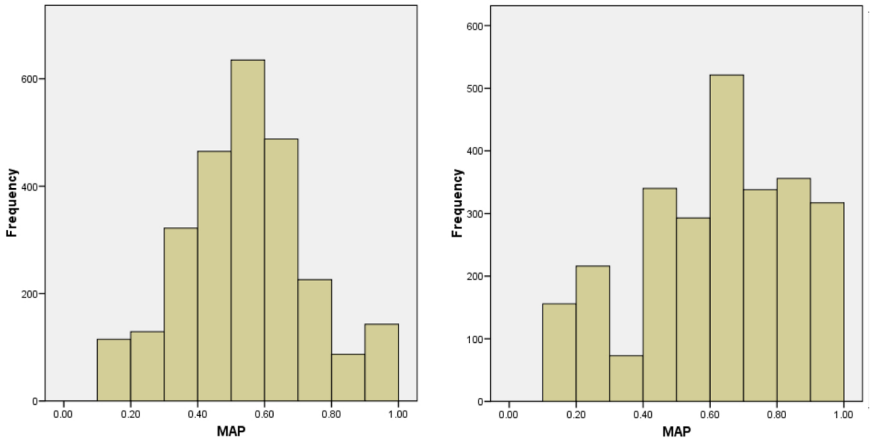


Fig. 1. The distribution of the Mean Average Precision for all combinations of parameters, for *Group 1* (left) and *Group 2* (right) used as training sets

see in Table 3 results for *Group 1* were better with LLD2 in comparison to LLD1, while for *Group 2* results were better with LLD1. The reason is a high difference in MAP for keywords: *5-HT receptors*, *trauma* and *trial*. All other keywords from Group 1 performed similarly for both datasets. However, looking closely into results of *Group 2* and the differences of MAP per keyword, there is a fluctuation with one half performing better with LLD1, while the other half performing better with LLD2 (see scattergram in Figure 2).

	Dataset	Number of runs	Mean	Std. Deviation
Group 1	1	1305	0.50	0.16
	2	1305	0.59	0.19
	1&2	2610	0.54	0.18
Group 2	1	1305	0.65	0.22
	2	1305	0.59	0.24
	1&2	2610	0.62	0.23

Table 3. The dispersion values for the distribution of MAP across two datasets

Looking closely into the effect of parameter variations, considering LLD1 and LLD2 groups independently, the results reveal that variations of the parameters did not have major influence on MAP. Detailed diagrams outlining the influence of the variation of all three parameters are shown in Figures 3, 4, and 5.

Although parameter variations seem not to have significant influence on MAP, there are certain patterns which are visible. Namely, the best minimum frequency parameter is in the bottom range (1 for *Group 2* and 2 for *Group 1*) for the smaller dataset, while for the larger, it seems to be in the top (10 for both groups). This might be an explanation for MAP being lower for the larger dataset: more data causes more noise which seems to be filtered nicely using the minimum frequency parameter.

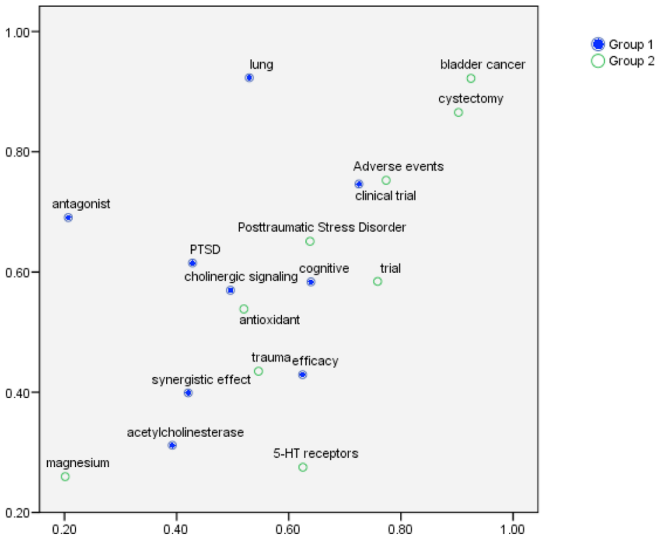


Fig. 2. Correlation of MAP for LLD1 (X axis) and LLD2 (Y axis) for all keywords

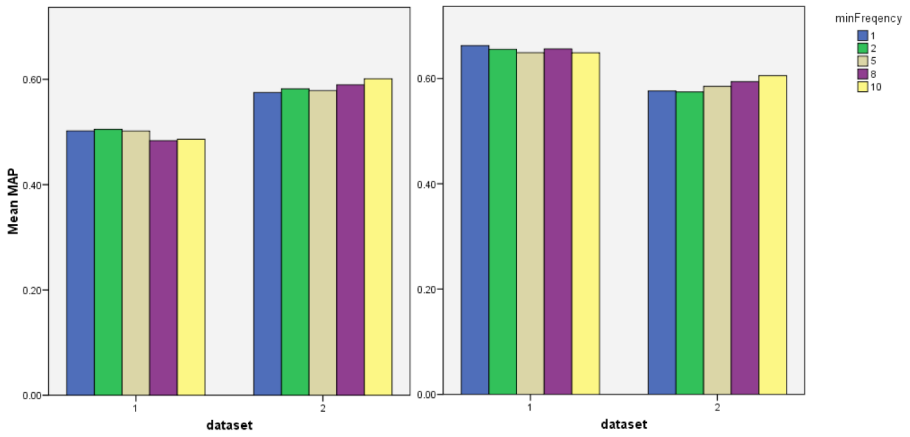


Fig. 3. The effect of the variation of minimal term frequency on *MAP*, across two datasets, for *Group 1* (left) and *Group 2* (right) used as training sets. The distribution of MAP across all categories of minimum term frequency is the same (independent samples Kruskal-Wallis test, p=0.44 and p=0.444 for LLD1 and LLD2 respectively, Group 1; p=0.808 and p=0.784 for LLD1 and LLD2, Group 2).

With regards to dimensionality, its variation has more influence on MAP with the smaller dataset, than with the larger one. This indicates that the value span which we chose for dimensionality parameter for LLD2 needs to be expanded in order to have any effect on results. However, MAP values are within reasonable range.

46

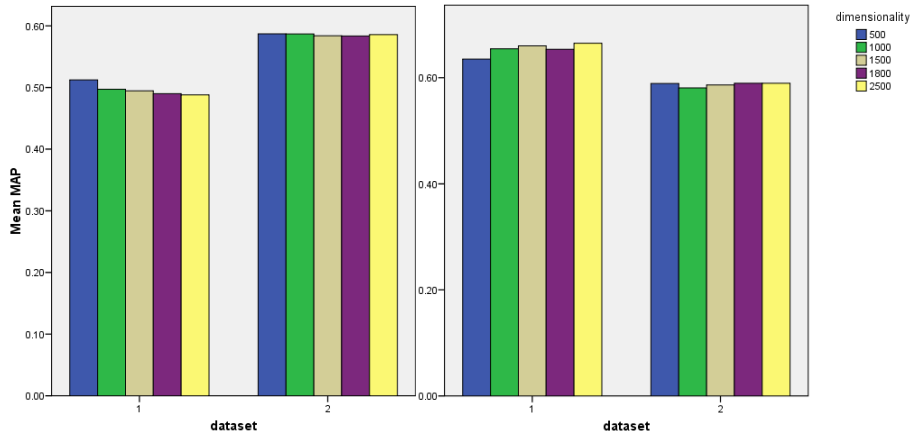


Fig. 4. The effect of the variation of dimensionality on *MAP*, across two datasets, for *Group 1* (left) and *Group 2* (right) used as training sets. The distribution of MAP across all categories of dimensionality is the same (independent samples Kruskal-Wallis test, p=0.676 and p=1.0 for LLD1 and LLD2 respectively, Group 1; p=0.587 and p=0.996 for LLD1 and LLD2, Group 2).

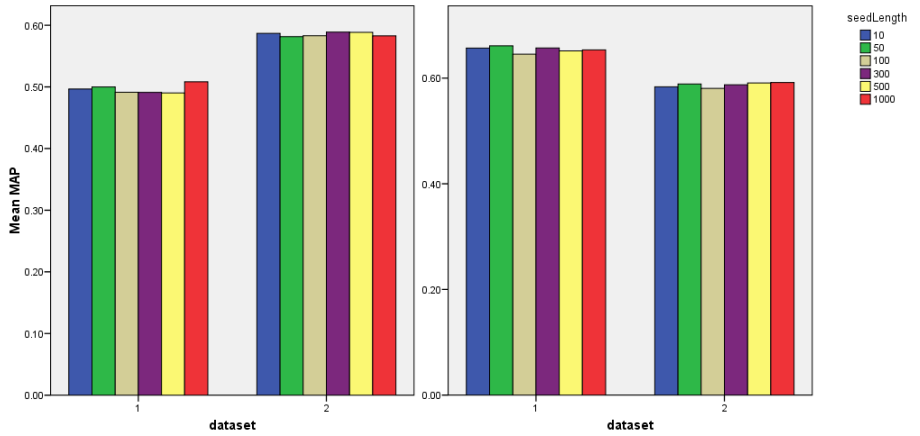


Fig. 5. The effect of the variation of seed length on *MAP*, across two datasets, for *Group 1* (left) and *Group 2* (right) used as training sets. The distribution of MAP across all categories of minimum term frequency is the same (independent samples Kruskal-Wallis test, p=0.931 and 0.997 for LLD1 and LLD2 respectively, Group 1; p=0.961 and 0.998 for LLD1 and LLD2, Group 2).

The variation of seed length parameter value seems not to cause any significant changes to MAP across both datasets, and hence, we consider the lowest value of this parameter as the optimal one, due to the fact that the the computational resources required to build and search the semantic space are proportional to the value of seed length. Table 4 outlines optimal parameters: those that we chose to use in the testing phase.

	Group 1		**Group 2**	
Dataset	LLD1	LLD2	LLD1	LLD2
Min frequency	2	10	1	10
Seed length	10	10	10	10
Dimensionality	500	500	1500	500
MAP	0.55	0.61	0.65	0.61

Table 4. Optimal parameters chosen for *Group 1* and *Group 2* used as training sets

Finally, the size of the dataset had a significant influence on MAP (Mann-Whitney U Test, $p < 0.0001$) for both Group 1 and 2 meaning that the larger set (LLD2) resulted in producing the higher value of MAP for Group 1, while for Group 2 the results were better with the smaller dataset (LLD1).

Testing the model In what follows we explore whether the model built using the optimal parameters just presented can be used to effectively test the model. In our context, testing the model means evaluating the set of related terms (literals and URIs) returned by our method for the set of testing keywords given as input.

We ran the search method using *Group 2* as a *testing* set against the RI model trained with *Group 1*, and then *Group 1* as a *testing* set against the RI model trained with *Group 2*. Results are shown in Table 5. The RI method results in as good or better MAP for *Group 2* in comparison to MAP for the best trained model (*Group 1* column in Table 4), while for *Group 1* the resulting MAP for LLD2 is as good as that of the best trained model (*Group 2* column in Table 4), while for LLD1 it is lower for 0.15. This is due to the distribution of keywords in *Group 1*, due to which MAP for the RI model with optimal parameters is only 0.05 higher (0.55).

In the testing phase, MAP across both groups reached 0.565 and 0.61 for LLD1, and LLD2 respectively.

	Group 2		**Group 1**	
Dataset	LLD1	LLD2	LLD1	LLD2
Min frequency	2	10	1	10
Seed length	10	10	10	10
Dimensionality	500	500	1500	500
MAP	0.63	0.61	0.5	0.61

Table 5. Testing the Random Indexing method using *Group 2* and *Group 1* as *testing* sets

Also important to observe is the fact that when the data corpus increases (e.g. LLD2 vs LLD1) the method becomes very stable, and observed MAP values in the training process are reproduced in the subsequent test phase. Arguably this is due to the small difference in MAP across parameters, but it still shows that RI is a stable method even in this unusual use-case we are dealing with.

Human assessment. In order to assess overall difficulty of the tasks which we solve using the RI method, we calculated Inter-annotator agreement, and indeed *Observed agreement* and *Cohen's Kappa agreement* (see Section 4.2). The observed agreement across all keywords was 0.81, and the Cohen's Kappa was 0.61 which indicates that the given task of selecting relevant keywords for a topic of interest was indeed difficult for domain experts.

The code and datasets from the described experiment, including generated virtual documents and semantic spaces, can be downloaded from the LarKC Wiki[13].

Performance The parameter values affect not only the quality of results but also the required resources and the indexing time. Increasing the value of dimensionality and seed length almost exponentially increases the time to generate the semantic space (from 0.67 minutes for 500 dimensions to 3 minutes for 2500, LLD1; from 3.78 minutes for 500 to 11.5 minutes for 2500 dimensions, LLD2). The higher the value for *seed length* and *dimensionality*, the higher the requirements for the computational resources and RAM in particular[14]. Application of RI to the whole LLD dataset poses the scalability issues related to the size of our corpus. While indexing is a one-off operation (that takes $\tilde{1}6$ hours on MDC computer with 256G RAM), the search for 'lung' after the space is generated takes 14 minutes. Therefore, in our related work reported elsewhere ([1]) we looked at the parallelisation of the RI search algorithm in order to make exploring large RDF graphs using the contextual similarities of the comprising nodes applicable in real time applications.

5 Conclusion and future work

We described the application of the Random Indexing method for the task of searching large and unknown RDF graphs. We tested our method on the subsets of the Linked Life Data, by training it using the variation of parameters, and then involving domain experts to judge on the relevance of retrieved terms. None of the parameters had a significant influence on MAP, apart from the size of the dataset. However, the values of MAP reaching 0.565 and 0.61 for LLD1, and LLD2 datasets respectively, indicate that the generation of virtual documents as described in this paper and generating the semantic index using the RI method has promising results. The reason for the stability of the RI method might have been the span of the parameters which we used, and hence in our future work we will expand the variation span and also repeat the runs across the same parameter variations in order to increase the significance of results.

Acknowledgments We would like to thank creators of SemanticVectors[15] library which is used in the experiments reported in this paper. This research has been supported by the EU-funded LarKC[16] (FP7-215535) project.

References

1. Assel, M., Cheptsov, A., Czink, B., Damljanovic, D., Quesada, J.: MPI Realization of High Performance Search for Querying Large RDF Graphs using Statistical

[13] `http://wiki.larkc.eu/LarkcProject/statisticalSemantics`

[14] The experiments are conducted on the MDC super-computer: 2 IBM x3950M2, 32 Cores (4 quad core Intel Xeon@2.93GHz per node), 256 Gbytes of main memory, production cluster for Java software and serial code.

[15] `http://code.google.com/p/semanticvectors/`

[16] `http://www.larkc.eu/`

Semantics . In: Proceedings of the 1st Workshop on High-Performance Computing for the Semantic Web, Collocated with the 8th Extended Semantic Web Conference (ESWC 2011). Heraklion, Greece (June 2011)
2. Broekstra, J., Kampman, A.: Serql: A second generation rdf query language. In: In Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval. pp. 13–14 (2003)
3. Cheng, G., Ge, W., Qu, Y.: Falcons: Searching and Browsing Entities on the Semantic Web. In: Proceedings of WWW2008. pp. 1101–1102 (2008)
4. Cohen, T., Schvaneveldt, R., Widdows, D.: Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections. Journal of Biomedical Informatics (2009)
5. Cohen, T.: Exploring medline space with random indexing and pathfinder networks. AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium pp. 126–130 (2008)
6. Croft, B., Metzler, D., Strohman, T.: Search Engines: Information Retrieval in Practice. Addison Wesley, 1 edn. (February 2009)
7. Damljanovic, D., Petrak, J., Cunningham, H.: Random Indexing for Searching Large RDF Graphs. In: Poster Session at the Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010). Lecture Notes in Computer Science, Springer-Verlag, Heraklion, Greece (June 2010)
8. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407 (1990)
9. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: Proceedings of the 13th ACM international conference on Information and knowledge management. pp. 652–659. ACM, New York, NY, USA (2004)
10. Eugenio, B.D., Glass, M.: The kappa statistic: a second look. Computational Linguistics 1(30) (2004), (squib)
11. Hogan, A., Harth, A., Decker, S.: Reconrank: A scalable ranking method for semantic web data with context. In: Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006). Athens, GA, USA (2006)
12. Hripcsak, G., Heitjan, D.: Measuring agreement in medical informatics reliability studies. Journal of Biomedical Informatics 35, 99–110 (2002)
13. Johnson, W.B., Lindenstrauss, J.: Extensions to lipschiz mapping into hilbert space. Contemporary Mathematics 26 (1984)
14. Karlgren, J., Sahlgren, M.: From words to understanding. In: Uesaka, Y., Kanerva, P., Asoh, H. (eds.) Foundations of Real-World Intelligence, pp. 294–308. Stanford: CSLI Publications (2001)
15. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation - 15 January 2008, W3C (2008)
16. Qu, Y., Hu, W., Cheng, G.: Constructing virtual documents for ontology matching. In: Proceedings of WWW2006. pp. 23–31 (2006)
17. Sahlgren, M.: An introduction to random indexing. In: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005. Citeseer (2005)
18. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Proceedings of the 6th International Semantic Web Conference. Busan, Korea (2007)

An organizational environment for in silico experiments in molecular biology

Yuan Lin[1], Marie-Angélique Laporte[1,3], Lucile Soler[4], Isabelle Mougenot[1,2], and Thérèse Libourel[1,2]

[1] LIRMM, UMR5506 CNRS-UM2, 161, rue Ada, 34095 Montpellier, Cedex 5, France
firstname.lastname@lirmm.fr
[2] UMR ESPACE DEV IRD-UM2, 500 rue J.F. Breton, 34093 Montpellier, Cedex 5, France
firstname.lastname@univ-montp2.fr
[3] Centre d'Ecologie Fonctionnelle et Evolutive, UMR5175 CNRS, 1919, route de Mende, 34293 Montpellier, Cedex 5, France
firstname.lastname@cefe.cnrs.fr
[4] CIRAD-PERSYST, Campus International de Baillarguet, 34398 Montpellier cedex 5, France
firstname.lastname@cirad.fr

Abstract. Molecular biologists, just like geneticists, make use of various experimental mechanisms and devices to conduct research and to validate or invalidate their theories or initial hypotheses. Mechanisms powered by information technology, called in silico, put data and analysis tools at the centre of the experiments, and are thus different from in vivo, ex vivo and in vitro mechanisms.
Multiple resources (data sources as well as analysis tools) are widely available and, very often, allow various modes of operation, requiring certain expertise for their optimal use. This is especially true when drawing up complex analysis scenarios based on the sequential use of appropriate processing tools. To facilitate the construction of these experimentation mechanisms, we propose a scientific workflow infrastructure which uses an organizational environment to allow abstract planning of the experimentation, followed by its concretization. The concretization phase includes a verification of the conformity of the planned process chains composition to avoid any error during execution.

Keywords: Scientific workflow, analysis pipeline, specification language, validation aspects of service composition.

1 Introduction

Life sciences often rely on the chaining of data and application resources to express the experimentation process. Valuable resources for biology, while available in ever-increasing quantities, remain, for the most part, cost-expensive and time-consuming to acquire and thus their reuse becomes almost a necessity.

To design these complex experiments, scientists often need to locate suitable resources and then to organize or reorganize them. In addition, each experiment deserves to be saved so that it can be re-executed several times, either in various different configurations or with diverse test data. In such a context, the use of a scientific workflow proves to be an invaluable help. Several dedicated software applications for this purpose now exist, most notably in the financial sector, and research in the field is relatively advanced. A first study [7] presented our approach based on the concept of the scientific workflow environment. Its objective is to help the user to:

- design experimentation process chains (in as abstract a manner as possible),
- better organize resources (data and processes) which will be elements in the concretization of these process chains,
- capitalize on the existing by constructing new processes from previously devised experimentation plans.

This article develops our research advances in terms of resource organization and semi-automatic verification of validity of workflows designed within a prototype.

This article is structured as follows: section 2 presents a brief state of the art, section 3 proposes an architecture for implementing a scientific workflow and section 4 provides a glimpse of the organization brought about. Section 5 covers the proposed verification of conformity, section 6 illustrates with an example the validation of conformity of a concrete process chain, and section 7 presents perspectives in progress.

2 State of the art

A study was conducted based on characteristics we deemed relevant [8]:

- The existence of a meta level for describing and creating process chains. In fact, the generic aspect conferred by meta-modelling appears to be fundamental for all of us.
- Taking the experimental aspect into account. The unique characteristics of scientific data and processes should show through at the formalism level.

We present here only two representative projects, Kepler[1] and Taverna[6], which gain a certain amount of popularity among workflow scientists.

2.1 KEPLER

KEPLER [5] is a complete scientific workflow environment based on the Ptolemy II platform of the University of Berkeley. As far as process chains are concerned,

[5] http://kepler-project.org/

KEPLER adopts a human organization metaphor. It is Actor-Based and considers all components of a process chain as actors. Actors (services) are accessed via a structure corresponding to the business ontology of the concerned domain.

The workflow is represented using a graphical language in the form of a graph linking *ports* (input/output parameters) of *actors* via *channels*. One or more actors in charge, *Directors*, plan tasks for other actors of the organization; they do so based on the available ontology. The execution plan of a process chain (or a portion of a process chain) is therefore created by a *Director* of the system. Any necessary adaptations are achieved by intermediary *sender* and *receiver* programs, which ensure the compatibility of data transferred over a channel. The process chain is saved in the form of MoML (Modelling Markup Language) files. (MoML is an XML-based language.) At the environment-interface level, a specific zoom feature is associated with the concept of an *opaque actor* (cf. figure 1). An *opaque actor* appearing in a process chain can be opened, thus revealing its constituent details.

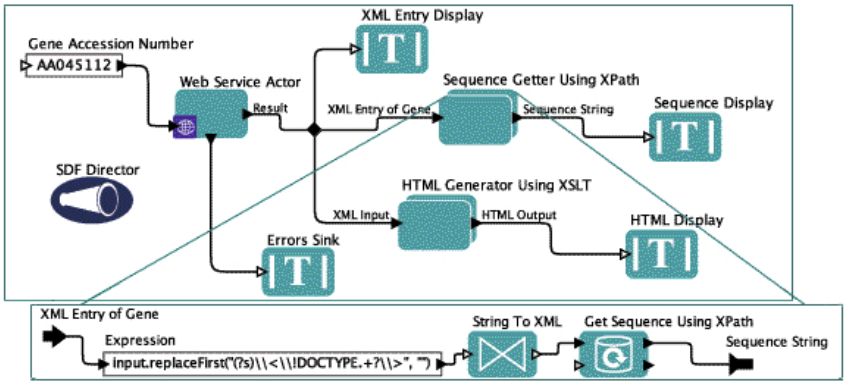


Fig. 1. Overview of a process chain in the KEPLER environment

2.2 Taverna

Taverna is a workflow project created by the ^{my}Grid team in England and used mainly in the life sciences. A workflow in Taverna is considered as a process graph in which processes are connected by data links or control links. Processes used are essentially web services (which can be supplemented by local libraries, manuscript scripts, etc.). During process composition, the user manually couples input/output parameters of web services or invokes *shim services*, specific adaptors existing from couplings constructed and tested for experiments. In addition, the process chain is saved in the form of a SCUFL (Simple Conceptual Unified Flow Language) file. (SCUFL is an XML-based language.)

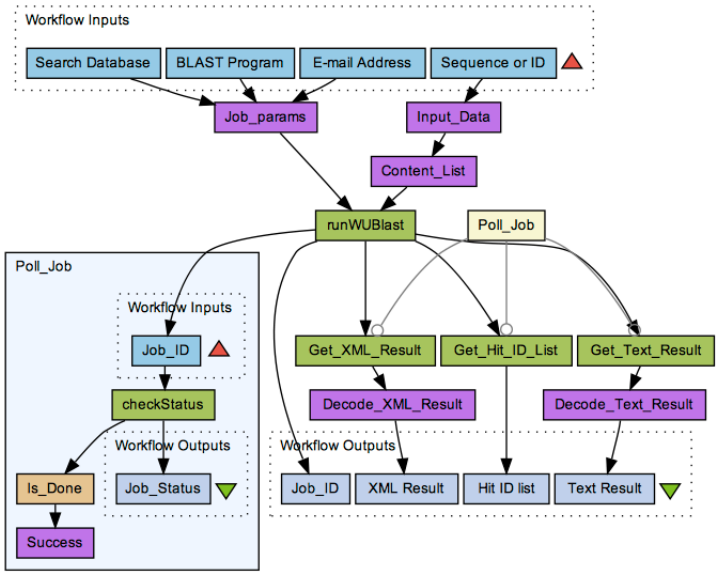


Fig. 2. A concrete workflow in Taverna (taken from the *myExperiment* Taverna sharing site)

2.3 Other related works of interest

The Taverna and Kepler projects both provide generic models for instantiation and composition of services. Additionally, some other approaches are also highly relevant to scientific workflow management:

- The project BioMoby [17], as a first attempt to assist process chaining by using scientific resources, which are described and classified in the *MOBY Central*.
- PISE ans its revised system Mobyle [18] that provides a web environment (a Web Portal) to define and execute bioinformatics analyses. Registered analysis programs are pre-classified in a hierarchy, as well as some frequently-used workflows. Experts can easily find them by using the search function panel that is integrated in the web site.
- The project ProtocolDB [19] proposed to model scientific workflows at two different layers (design protocol/ implementation protocol). An implementation protocol for a given design protocol is realized by mapping design tasks to different implementation tasks (scientific resources like database queries/ tools), and by connecting them together.
- In [20–22], scientific workflow modeling is supported by resource discovery approaches.

In this manuscript we focus mainly on scientific workflows and the way they are modeled and implemented. Our proposal introduces an additional level of

abstraction, whose purpose is to describe the business domain prior to creating the process chains. This additional modelling level is predicted to facilitate the construction of process chains by allowing biologists to use their expertise of their domain, but without requiring them to have expert and often precise knowledge of the underlying resources and their locations. It also plays the role of a prescription model, to which instantiation and service composition models have to conform.

3 Workflow architecture

Our efforts have been guided by the business point of view, that of the experimenters. Designing an experimental protocol corresponds to general model with three stages: 1) *Definition*: abstract definition of a process chain corresponding to an experimentation sequence (planning the experiments), 2) *Instantiation*: a more specific definition after identifying the various elements of the chain (data/processes), 3) *Execution*: customized execution (according to strategies corresponding to the requirements).

Based on this experimental life cycle, and inspired by the architectural styles proposed by OMG [11], we propose the following 3-level architectural vision (cf. figure 3):

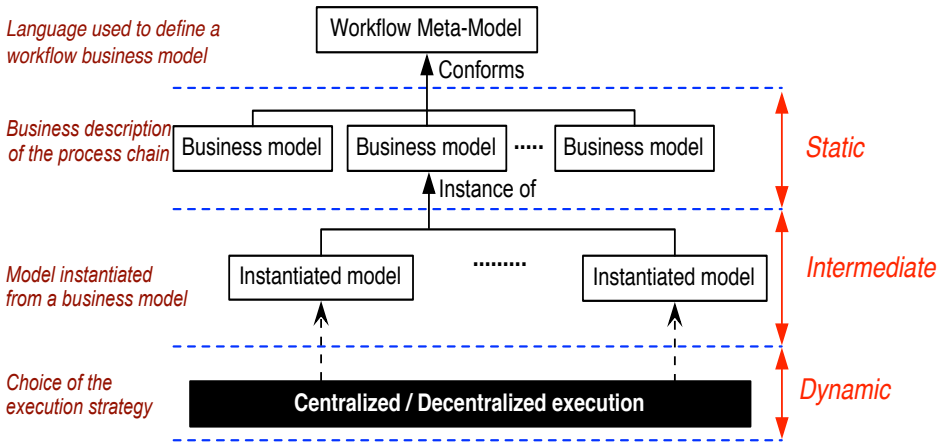


Fig. 3. 3-level architecture of a workflow component

The *static* level concerns the design phase. It is a matter of constructing (abstract) business-process models using a simple language. The *intermediate* level represents an instantiation and pre-verification phase. Using the business

process model, the user constructs the real process chain by selecting and locating the processes and data most appropriate to the planned experimentation. The pre-verification is semi-automatized (cf. section 4). The *dynamic* level concerns the actual execution phase. It takes place based on the various strategies defined by both the user and the operational configurations.

The *static* level has been studied in some detail in our [7, 8]. We have analyzed various language standards such as UML (activity diagram) [9] and SPEM [10], as also various existing projects such as BioSide [5], Meta-model WDO-It! [12] and CIMFlow [4]. Following this study, we proposed a simple but complete language. It is based on a language defined by a meta-model whose abstract elements, *tasks* or *processes*, are connected by unidirectional links and by the intermediary of *ports*. To facilitate the manipulation of abstract process chains, a corresponding graphical language was created within a prototype (cf. the top part of the figure 4). By using this workflow definition language, a simple example is modelled and shown in the lower part of the figure 4 [6].

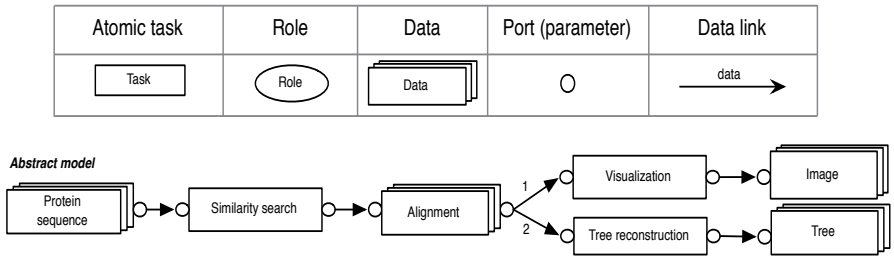


Fig. 4. Some essential elements of our graphical language and a simple example

We currently focus on the *intermediate* level, which consists of two essential stages:

- instantiation of the abstract model with existing resources (data/processes);
- validation of the concrete model instantiated from the organizational environment.

4 Organizational environment

To carry out the experimental protocols, the abstract model *instantiation* stage consists of finding and reusing existing resources. To facilitate this search, we base ourselves on the concept of *organizational environment*. This environment relies on the description of resources (data and processes) in the form of metadata

[6] This example is also used in the later sections, we will explain it in detail during the following sections.

(expressed in XML schema format). The resource descriptions are hierarchized in resource categories and in concrete resources. As shown in figure 5, it consists of:

- an organization relating to processes. It manages the hierarchy of descriptions of process categories and of concrete processes. The concept of *Converter* corresponds to the concept of a specific process responsible for adapting data between different formats of the same data category.
- an organization relating to data. It manages a hierarchy of descriptions of data categories, of concrete data and of the various associated data formats[7].

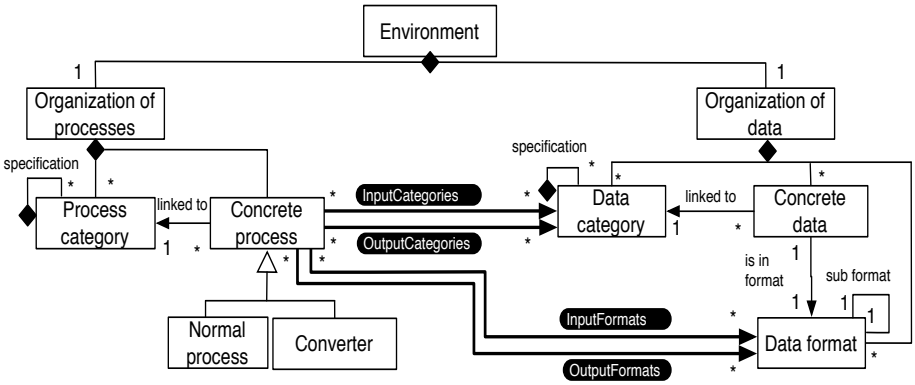


Fig. 5. Organizational environment

To illustrate this concept of the environment, we take an example from the world of molecular biology (cf. figure 6). The upper part of each hierarchy (processes and data) represent a set of categories (shown as ovals) sorted according to the generalization/specialization relationship. The descriptions of concrete resources (data or processes) are then associated to their category.

The description of a concrete data describes its format, whereas that of a concrete process corresponds to its *signature*, which we formalize thus:

Definition 1. *Formalized signature of a concrete process*

Name (Input parameter list) : (Output parameter list), where each parameter is described by the doublet (Data category : data format).

A set of data formats (Fasta, xml, MultiFasta, Clustal, Newick, Jpeg) is also presented. Figure 6 is therefore complemented by the description of signatures of some example concrete processes:

[7] Remark: It should be noted that several data categories can share the same format.

Blastp(ProteinSeq:Fasta) : (SeqPairs:xml)
ClustalW(ProteinDataBank:MultiFasta) : (MultipleAlignment:Clustal)
InteractiveSelection(SeqPairs:xml) : (ProteinDataBank:MultiFasta)
Logo(MultipleAlignment:Clustal) : (Image:jpeg)
PhyML(MultipleAlignment:Clustal) : (PhylogeneticTree:Newick)

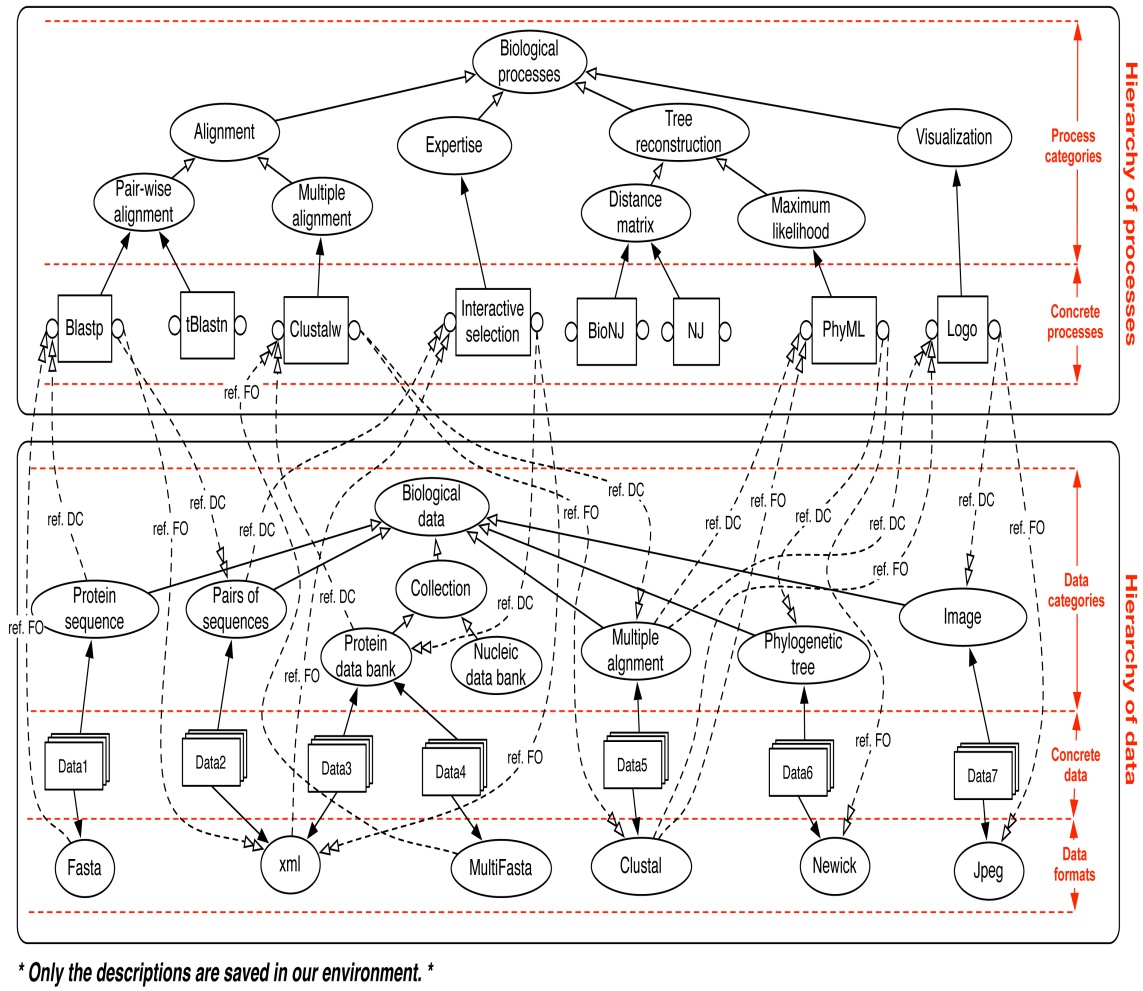


Fig. 6. Illustration of an organizational environment in a biological context

5 Conformities

5.1 The problem

As already mentioned, the second important stage of the *intermediate* level consists of validating the concrete model instantiated from the abstract model.

Let us take an example described by using the workflow language, corresponding to an abstract process chain model that a biologist designs with the intention of characterizing a protein sequence which interests him in the context of his putative functional domains.

At the concrete level, the idea is to begin by using the *Blast* similarity-search tool to compare the protein sequence under consideration with a data bank of protein sequences and to thus identify segments with high similarity shared both by the protein sequence under consideration and by various sequences in the sequence data bank. These similar segments indicate the possible presence of functional domains. The biologist then continues his study by reusing the results output from the *Blast* tool [2], either to construct a phylogenetic tree and retrace the evolutionary history of the sequence via the *PhyML* tool [3] or to display the preserved positions common to all the similar segments via the *Logo* tool [13]. This simplified example of a process chain in molecular biology allows us to highlight the difficulties encountered by the biologist in using the results output by one tool as input to another tool. The difficulties relate, at the same time, to the nature of the data (here characterized as data category), to the format of this data, and, finally, to the biologists expertise. In the example, we make willing use of the discrepancy which arises between the Blast tool, which outputs a collection of simple alignments, and the PhyML and Logo tools, which require multiple alignments to run. In fact, Blast leads to multiple discrepancies two-by-two, involving the sequence under consideration and one of the sequences from the sequence data bank which is similar to it; whereas PhyML and Logo use the shared similarity by a set of sequences which includes the sequence under consideration. This example highlights what we will subsequently term *semantic incompatibility*.

In its upper part, the figure 7 shows the abstract process chain and in the lower the concrete chain obtained after locating data descriptions *S1* and adapted processes *Blastp* and *PhyML*. The problem which we designate as one of *validation of the instantiated (concrete) model* consists of verifying the *compatibility* of each *composition*. A *composition* corresponds to the link between an output parameter p1 of a process T and an input parameter p2 of the process following T; we denote it *(p1 → p2)*.

5.2 Identifying situations of compatibility

Verification is undertaken by analyzing the signatures of linked processes. To do so, we have to take two important aspects into account:

– The *syntactic* aspect : relating to the data formats used by the parameters.

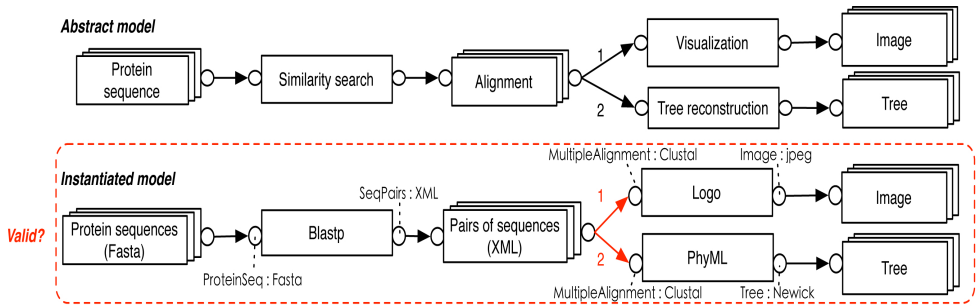


Fig. 7. Problem at hand

– The *semantic* aspect : relating to the processs functionality. It not only depends on the processs name but also on the signification of the input/output parameters.

For two processes *T1(dc1:fo1) : (dc2:fo2, dc3:fo3)* and *T2(dc4:fo4) : (dc5:fo5)*, let us suppose that there exists a composition, denoted *p1→p2*, between the *p1 (dc3:fo3)* output parameter of process *T1* and the *p2 (dc4:fo4)* input parameter of process *T2*.

Syntactic and semantic compatibilities are defined as follows:

Definition 2. *Syntactic compatibility*

p1 → *p2* is syntactically compatible if *(fo3 = fo4)* ∨ *(fo3 is a sub-format of fo4)*, denoted $p1 \overset{Syn}{\rightarrow} p2$. Two parameters are syntactically compatible if they use the same data format or if they use an output format which is a sub-format of the input format. Else $p1 \overset{Syn}{\not\rightarrow} p2$.

Definition 3. *Semantic compatibility*

p1 → *p2* is semantically compatible if *(dc3 = dc4)* ∨ *(dc3 is a sub-category of dc4)*, denoted $p1 \overset{Sem}{\rightarrow} p2$. Two parameters are semantically compatible if they use the same category, or if they use an output category which is a sub-category of the input category. Else $p1 \overset{Sem}{\not\rightarrow} p2$.

The verification of a compositions compatibility is thus done at two levels: syntactic and semantic. Three types of situations can arise:

– Situation 1 $(p1 \overset{Sem}{\rightarrow} p2) \wedge (p1 \overset{Syn}{\rightarrow} p2)$: *p1* and *p2* are compatible at the semantic and syntactic levels. This is the ideal situation in our context; we designate it as valid.
– Situation 2 $(p1 \overset{Sem}{\rightarrow} p2) \wedge (p1 \overset{Syn}{\not\rightarrow} p2)$: *p1* and *p2* are compatible at the semantic level but not at the syntactic level. The composition is syntactically adaptable. An adaptation between the two data formats will be necessary (cf. converters).

– Situation 3 $p1 \overset{Sem}{\nrightarrow} p2$: The two parameters are not semantically compatible. In such a case, it is pointless to proceed to verify their syntactic compatibility (in fact, for us, two parameters with different significations cannot be paired). The composition is semantically adaptable.

From these definitions, we develop our proposed approach for resolving the incompatibilities.

6 Validation of the experimental chain

Of the three compatibility situations identified, the latter two require an adaptation stage before going on to the execution phase. It is a matter of finding one or more intermediate processes which can overcome the compositions incompatibility. For situations 2 and 3, two types of adaptations are proposed:

– *semantic* adaptation (for situation 3). The incompatibility of situation 3 represents the case where the two parameters of a composition use incompatible data categories. The adaptation here consists of finding a possible intermediate process chain between these two categories.
– *syntactic* adaptation (for situation 2). In situation 2, where the composition is already semantically compatible, the problem can be expressed as a divergence between the data formats used by the two connected parameters. All that is required is to find *converters* to convert one data format into the other.

These adaptations are based on the organizational environment. The search for intermediate processes can be equated to a search for itineraries between two incompatible data categories or formats. We will illustrate this using the example and the organizational environment constructed earlier (cf. figure 6).

Let us consider again the previous example. The verification conducted on the instantiation of the abstract model detects a semantic incompatibility in the composition between Blastp and Logo or between Blastp and PhyML due to difference in categories *Pairs of sequences* and *Multiple Alignment* (*Incompatibility situation 3*). The (semantic) adaptation will be applied; it consists of finding in what we call the (semantic) resource graph the path allowing the conversion of categories.

The construction of the (semantic) resource graph consists of extracting, from the organizational environment, the descriptions of processes and of data categories referenced by their parameters. Such a (semantic) resource graph generated from the environment described in the figure 6 is shown in the figure 8.

A graph traversal algorithm is used to find all the possible paths between the two concerned data categories (*Pairs of sequences* and *Multiple Alignment*). A single path is found in the graph: *Pairs of sequences* → *InteractiveSelection* → *ProteinDataBank* → *ClustalW* → *Multiple Alignment*. The two processes, *InteractiveSelection* and *ClustalW*, will therefore be added to the incompatible chain (cf. figure 9).

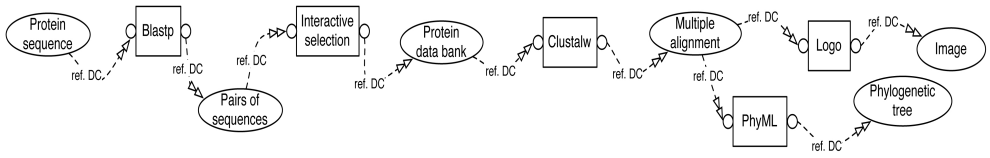


Fig. 8. (Semantic) resource graph generated from the organizational environment of the figure 6

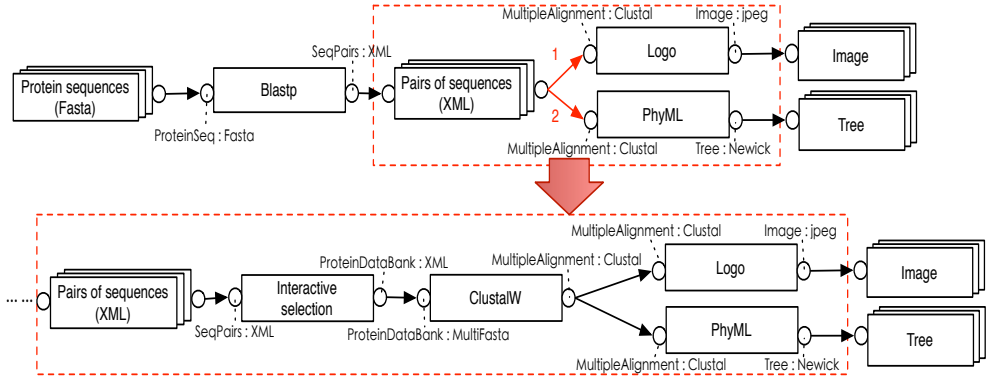


Fig. 9. Semantic adaptation

Once this adaptation is done, there still remains the existing syntactic incompatibility of the composition between the *InteractiveSelection* and *ClustalW* processes because even though *InteractiveSelection* outputs the same data category that is accepted for input by *ClustalW*, their data formats are different (*xml* and *MultiFasta*). Syntactic adaptation consists of finding specific *converters*, or compositions of *converters*, necessary for these conversions. We will not cover this stage in detail; it is simply enough to understand that converters (or their composition) can be added to obtain the required validity.

7 Conclusion and perspectives

A prototype (http://www.lirmm.fr/ lin/project/) illustrating the key aspects of our approach for designing and validating scientific process chains is currently being developed. This prototype serves as a basis for an inductive experimental approach using data of BAC and EST nucleic sequences as well as physical and genetic maps for identifying and characterizing genetic markers relating to sex of the Nile tilapia (Oreochromis niloticus). Over a longer term, we intend to integrate the current prototype into a platform with a search engine based on resource descriptions to be able to undertake the execution using real re-

sources, after requisite validation of experimentation chain. It will eventually also use open-source controlled vocabularies such as PFO (Protein Feature Ontology)[14], SO (Sequence Ontology)[15], and GO (Gene Ontology)[16] to enrich data categories by additional representations and thus extend the descriptive capacities of the organizational environment.

References

1. I. Altintas, B. Ludäscher, S. Klasky, and M. A. Vouk. S04 - *introduction to scientific workflow management and the kepler system*. In SC, page 205, 2006.
2. S. Altschul, W. Gish, W. Miller, E. Myers and D. Lipman. *Basic local alignment search tool*. In Journal of Molecular Biology, vol 215, pages 403-410, 1990.
3. S. Guindon and O. Gascuel. *A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood*, in Systematic Biology, vol 52, pages 696-704, 2003.
4. L. Haibin, F. Yushun, *CIMFlow: A Workflow Management System Based on Integration Platform Environment*. In Proceedings of 7th IEEE International Conference on Emerging Technologies and Factory Automation. Barcelona : ETFA, 1999: 187-193.
5. M. Hallard & al. *Bioside : faciliter l'acceès des biologistes aux ressources bioinformatiques*, JOBIM, Montreéal 2004, p 64.
6. D. Hull, K. Wolstencroft, R. Stevens, C. A. Goble, M. R. Pocock, P. Li, and T. Oinn. *Taverna: a tool for building and running workflows of services*. Nucleic Acids Research, 34(Web-Server-Issue):729732, 2006.
7. T. Libourel, Y. Lin, I. Mougenot, C. Pierkot, JC. Desconnets, *A Platform Dedicated to Share and Mutualize Environmental Applications*. Proceedings of 12th International Conference on Enterprise Information Systems, Madere, 2010.
8. Y. Lin, T. Libourel, I. Mougenot, *A Workflow Language for the Experimental Sciences*, Proceedings of 11th International Conference on Enterprise Information Systems, Milan, 2009.
9. Object Management Group (OMG), *OMG Unified Modeling LanguageTM (OMG UML), Infrastructure Version 2.3*. OMG Document Number: formal/2010-05-03.
10. Object Management Group (OMG), *SPEM - Software & Systems Process Engineering Meta-Model Specification, Version 2.0*. OMG Document Number: formal/2008-04-01.
11. Object Management Group (OMG), *Meta Object Facility (MOF) Core Specification OMG Available Specification Version 2.0*, OMG Document Number: formal/06-01-01.
12. P. Pinheiro da Silva, L. Salayandia, A.Q. Gates, *WDO-It! A Tool for Building Scientific Workflows from Ontologies* (2007). Departmental Technical Reports (CS). Paper 201.
13. T. D. Schneider and R. M. Stephens, *Sequence Logos: A New Way to Display Consensus Sequences*. In Nucleic Acids Res., vol 18, pages 6097-6100, 1990.
14. G.A. Reeves, K.Eilbeck, M.Magrane, C.O'Donovan, L.Montecchi-Palazzi, M.A. Harris, S.E. Orchard, R.C. Jimenez, A.Prlic, T. J. P. Hubbard, H.Hermjakob, J.M. Thornton. *The Protein Feature Ontology: a tool for the unification of protein feature annotations*. In Bioinformatics, vol 24, pages 2767-2772, 2008.
15. K.Eilbeck, S.E Lewis, C.J Mungall, M.Yandell, L.Stein, R.Durbin, M.Ashburner. *The Sequence Ontology: a tool for the unification of genome annotations*. In Genome Biology, vol 6, pages R44, 2005.

16. M.Ashburner, C.A. Ball, J.A. Blake, D.Botstein, H.Butler, J. Michael Cherry, A.P. Davis, K.Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L.Issel-Tarver, A.Kasarskis, S.Lewis, J.C. Matese, J. E. Richardson, M.Ringwald, G.M. Rubin, G.Sherlock, *Gene ontology: tool for the unification of biology. The Gene Ontology Consortium.* In Nature Genetics, vol 25, pages 25-29, 2000.
17. Michael DiBernardo, Rachel Pottinger, Mark Wilkinson: *Semi-automatic web service composition for the life sciences using the BioMoby semantic web framework.* Journal of Biomedical Informatics 41(5): 837-847 (2008).
18. Bertrand Néron, Hervé Ménager, Corinne Maufrais, Nicolas Joly, Julien Maupetit, Sébastien Letort, Sébastien Carrère, Pierre Tufféry, Catherine Letondal: *Mobyle: a new full web bioinformatics framework.* Bioinformatics 25(22): 3005-3011 (2009).
19. Michel Kinsy, Zoé Lacroix, Christophe Legendre, Piotr Wlodarczyk, Nadia Yacoubi Ayadi: *ProtocolDB: Storing Scientific Protocols with a Domain Ontology.* WISE Workshops 2007: 17-28
20. Zoé Lacroix: Resource Discovery, Second International Workshop, RED 2009, Lyon, France, August 28, 2009. Revised Papers Springer 2010.
21. Zoé Lacroix, Cartik R. Kothari, Peter Mork, Rami Rifaieh, Mark Wilkinson, Juliana Freire, Sarah Cohen Boulakia: *Biological Resource Discovery.* Encyclopedia of Database Systems 2009: 220-223.
22. Nadia Yacoubi Ayadi, Zoé Lacroix, Maria-Esther Vidal: *A Deductive Approach for Resource Interoperability and Well-Defined Workflows.* OTM Workshops 2008: 998-1009.

A Directory of Heterogeneous Services[1]

Zijie Cong, Alberto Fernández, Carlos A. Soto

CETINIA, University Rey Juan Carlos, Móstoles, Spain
zijie@ia.urjc.es, alberto.fernandez@urjc.es, casotob@ia.urjc.es

Abstract.
This paper presents a directory of heterogeneous web services, which addresses the issue of service discovery involving heterogeneous description languages such as OWL-S, SAWSDL, WSDL and plain text. Service descriptions are mapped into a unified description model, which captures various important elements in different service description approaches. Our directory then performs service registration, automatic discovery and manual browsing utilizing these unified models. A preliminary evaluation shows a satisfying result.

Keywords: service directory, service discovery, matchmaking, semantic web services, service oriented architecture.

1 Introduction

In Service-Oriented Architectures, web services can be described in various models, from highly expressive semantic web service description languages such as OWL-S and WSMO to plain text. The possibility and capability of automatic service discovery is limited by the diversity of service description models.

A directory of heterogeneous web services is presented in this paper, which addresses the issue of service discovery involving various service description models. Common approaches use the same description language for both advertisements and requests.

Services description in different description languages are mapped into a unified model, which dedicates to service matchmaking purpose, before registration. This unified model captures many important features of existing description languages, such as the semantic I/Os, category information and syntactic description. It is independent of the original service description language, thus it can be modified and expanded with minimal effort while avoiding the complication of mapping a less expressive description language, such as keywords, to a highly expressive description language with additional information requirement. A matchmaking algorithm is

[1] Work partially supported by the Spanish Ministry of Science and Innovation through grants TIN2009-13839-C03-02 and CSD2007-0022 (CONSOLIDER-INGENIO 2010)

performed over this model, thus providing heterogeneous service discovery capabilities.

The rest of the paper is organized as follows: In section 2, we describe the general structure of the directory, and the mapping from existing service description languages to a unified model. In section 3, the matchmaking process is explained in detail, and the implementation and preliminary evaluation of some components is shown in section 4. The related works and conclusion are then presented in section 5 and 6, respectively.

2 Service Directory Architecture

The architecture of our service directory is depicted in Fig. 1. There are two types of agents that interact with the directory, the one who offers the service (*Service Provider*) and the consumer of services (*Service Requester*). As we will see in section 4, they can access the directory through a REST service or a human-oriented web interface.

Service providers register services in the directory providing the following information:

- *Service Description*: the service description specified by the provider is essential because it will contain all the information related to the service offered (it can include the service category). In our framework we allow several service description models. They include semantic models (OWL-S [16], WSMO [3]), syntactic models (WSDL [4]), hybrid (SAWSDL [6]), as well as other lighter approaches (*keyword-*, *cloud-*, and *text*-based service descriptions).
- *Grounding*: the service provider must attach the information required to access the service by a client (for example a WSDL file).
- *Category* (optional): the category of the service can be explicitly defined in this section according to the NAICS [18] classification. As we will see later, service category is complemented with information provided in the *service description* section, such as explicit annotation (e.g. in some versions of OWL-S) or extracted from a textual description.

Service descriptions and category are combined and converted into a common format (*AT-GCM*) and stored in a *Service Registry*. The common format (section 2.1) comprises the relevant characteristics of the original models, from a service matchmaking point of view. The *Mapping to AT-GCM* module generates the AT-GCM version of the service from the service description and the category.

The *AT-GCM*, the *Grounding*, and the original *Service Description* provided by the Service Provider are stored as an entry in the *service registry* database.

When client agents (*service requesters*) want to use the service directory for finding a service, they send the necessary information (*Query Description*) to obtain a list of matching services (sorted list by their degree of match with the query). Query descriptions are specified using one of the available description languages. Note that our framework is able to return services described in a different language to the query. For instance, it may return an OWL-S service while the query is specified using WSDL.

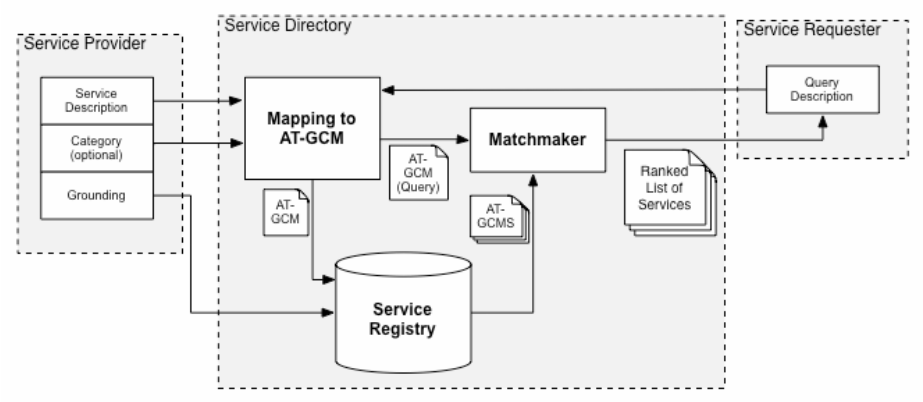


Fig. 1. Service Directory Architecture

When the service directory receives a query description, the query is transformed into the ATM-GCM format (*Mapping to AT-GCM*) and passed to the *Matchmaker*. Then, the matchmaker compares the query against the AT-GCM versions of the services stored in the database and returns a ranked list of services to the client. This process is detailed in section 3.

2.1 A unified model for representing service descriptions

Setting out from existing conceptual comparisons between semantic web service descriptions ([11, 12, 20, 22], and considering lighter approaches too, we obtained a *General Common Model* (*AT-GCM*[2]) with the following elements: *inputs, outputs, preconditions, effects, keywords, textual description, category* and *tag cloud*.

Detailed description about the model and the mappings from original models to the *AT-GCM* can be found in [2]. Here we summarise that description.

Definition 1. Let $\mathcal{N}$ be a set of concepts of domain ontologies, a *general common model* (*AT-GCM*) for service discovery is a tuple $<\mathcal{I}_{GCM}, \mathcal{O}_{GCM}, \mathcal{P}_{GCM}, \mathcal{E}_{GCM}, \mathcal{K}_{GCM}, \mathcal{C}_{GCM}, \mathcal{T}_{GCM}, \mathcal{TC}_{GCM}>$, where:

- $\mathcal{I}_{GCM} = <I_{syn}, I_{sem}>$ is the set of syntactic ($I_{syn} \in \{a, .., z\}^*$) and semantic ($I_{sem} \subseteq \mathcal{N}$) inputs of the service.
- $\mathcal{O}_{GCM} = <O_{syn}, O_{sem}>$ is the set of syntactic ($O_{syn} \in \{a, ..., z\}^*$) and semantic ($O_{sem} \subseteq \mathcal{N}$) outputs.
- $\mathcal{P}_{GCM}$ is the set of preconditions. $\mathcal{P}_{GCM} \subseteq \mathcal{N}$
- $\mathcal{E}_{GCM}$ is the set of effects. $\mathcal{E}_{GCM} \subseteq \mathcal{N}$
- $\mathcal{K}_{GCM} = <\mathcal{K}_{syn}, \mathcal{K}_{sem}>$ is the pair of sets of syntactic and semantic keywords, where $\mathcal{K}_{syn} \subseteq \{a, \ldots, z\}^*$, $\mathcal{K}_{sem} \subseteq \mathcal{N}$.

[2] *AT* stands for *Agreement Technologies*, meaning agreement among different service description models. It is also the name of one of our funding projects (CSD2007-0022).

- $\mathcal{C}_{GCM}$ is a set of categories of the service, described semantically ($C_{sem} \subseteq \mathcal{N}$) (e.g. NAICS or UNSPSC).
- $\mathcal{T}_{GCM}$ is a textual description of the service.
- $\mathcal{TC}_{GCM}$ is a tag cloud. $\mathcal{TC}_{GCM} = \{<t, n> | t \in \{a, \ldots, z\}^*, n \in \mathbb{N}\}$.

Table 1 shows how the different elements of the *AT-GCM* can be obtained from each source service description model. The first column specifies the element of the *AT-GCM*, while each cell contains the value mapped from the model shown in the first row.

There are many straightforward mappings that consist of simple associations between parameters in both models. For instance, in OWLS/WSMO $\mathcal{I}_{GCM}=<\emptyset$, pt($\mathcal{I}$)> because they only provide semantically described inputs $\mathcal{I}$ ($\mathcal{I}_{sem}$), where pt($\mathcal{I}$)={ t | t=parameterType(i) $\forall i \in \mathcal{I}$}.

However, some fields (e.g. tag-clouds, keywords) may not be explicitly described by a given model but they can be obtained from the rest of the description.

Table 1. Service(S)-to- *AT-GCM* mapping

GCM	**OWL-S / WSMO**	**SAWSDL**	**WSDL**	**Keyword (tag)**	**Tag Cloud**	**Text**
$\mathcal{I}_{GCM}$	$<\emptyset$, pt($\mathcal{I}$)>	$<\mathcal{I}_{syn}, \mathcal{I}_{sem}>$	$<\mathcal{I}, \emptyset>$	$<\emptyset, \emptyset>$	$<\emptyset, \emptyset>$	$<\emptyset, \emptyset>$
$\mathcal{O}_{GCM}$	$<\emptyset$, pt($\mathcal{O}$)>	$<\mathcal{O}_{syn}, \mathcal{O}_{sem}>$	$<\mathcal{O}, \emptyset>$	$<\emptyset, \emptyset>$	$<\emptyset, \emptyset>$	$<\emptyset, \emptyset>$
$\mathcal{P}_{GCM}$	$\mathcal{P}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{E}_{GCM}$	$\mathcal{E}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{C}_{GCM}$	$\mathcal{C}$	$Cat(\mathcal{T})$	$Cat(\mathcal{T})$	$Cat(\mathcal{T})$	$Cat(\mathcal{T})$	$Cat(\mathcal{T})$
$\mathcal{T}_{GCM}$	$\mathcal{T}$	$\mathcal{T}$	$\mathcal{T}$	$\emptyset$	$\emptyset$	S
$\mathcal{TC}_{GCM}$	$\Delta(\mathcal{T}) \cup \mathcal{N}(\mathcal{I}) \cup \mathcal{N}(\mathcal{O})$	$\Delta(\mathcal{T}) \cup \mathcal{I}_{syn} \cup \mathcal{O}_{syn}$	$\Delta(\mathcal{T}) \cup \mathcal{I} \cup \mathcal{O}$	$\{<t, 1> \mid t \in \mathcal{K}_{syn}\}$	S	$\Delta(S)$
$\mathcal{K}_{GCM}$	$<\tau(\Delta(\mathcal{T})) \cup \mathcal{N}(\mathcal{I}) \cup \mathcal{N}(\mathcal{O})$, pt($\mathcal{I}$)$\cup$ pt($\mathcal{O}$)>	$<\tau(\Delta(\mathcal{T})) \cup \mathcal{I}_{syn} \cup \mathcal{O}_{syn}$, $\mathcal{N}(\mathcal{I}_{sem}) \cup \mathcal{N}(\mathcal{O}_{sem})>$	$<\tau(\Delta(\mathcal{T})) \cup \mathcal{I}_{syn} \cup \mathcal{O}_{syn}$, $\phi>$	$\mathcal{K}$	$<\tau(S), \emptyset>$	$<\tau(\Delta(S)), \emptyset>$

Fig. 2 sumarises the characteristics of the *AT-GCM* that can be obtained from each original service description model.

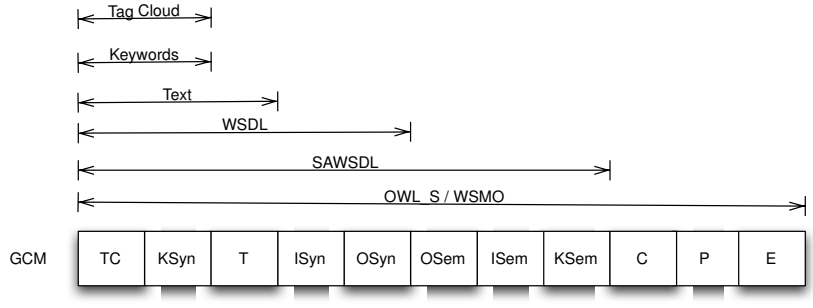


Fig. 2 AT-GCM characteristics covered by service description models

2.2 Model Expansion

Useful information about services may not always be explicitly defined by the providers in their service descriptions. Such information could, however, be discovered from other elements in the description and/or by using external resources. In this section, we briefly introduce the expansion of *AT-GCM* using existing elements and external resources.

A complete schema is shown in Fig. 3.

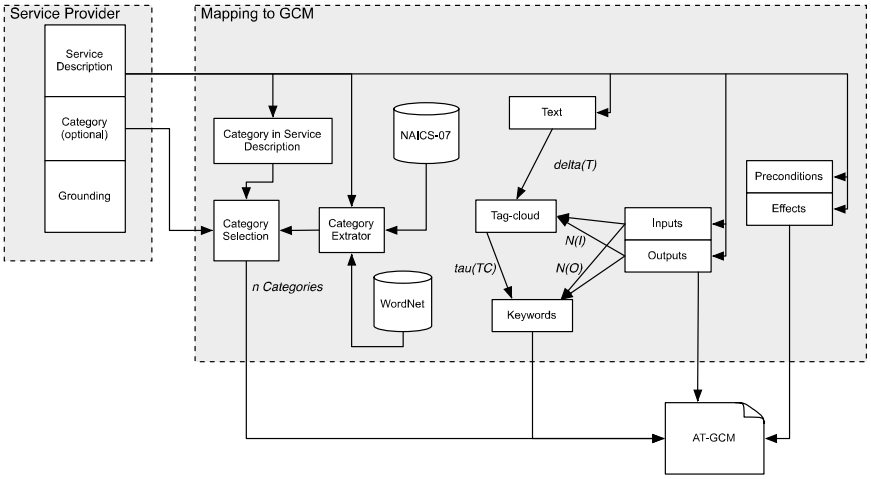


Fig. 3 Mapping to AT-GCM

2.2.1 Extracting tag-clouds and keywords from text

Although, as illustrated in Fig. 2, most service description languages include neither syntactic keywords nor tag-cloud, these two elements can be extracted from other parts of description such as text, inputs and outputs.

Function $\Delta(\mathcal{T})$ (Table 1) extracts the k most relevant keywords from $\mathcal{T}$. The relevance of each word in textual information is their TF-IDF weights [24] calculated using other textual information of services registered in our directory.

Before computing the TF-IDF weight of the word, a set of stop-words is filtered out from the text to accelerate the process. As nouns and verbs are more semantically significant than other parts of speech, words falling into the rest of lexical categories are also filtered out. This process is done using WordNet [17].

WordNet is a lexical database for English language. It groups English words into sets of synonyms called *synsets*, with various semantic relations between these synsets. These semantic relations include *hyponym, hypernym, domain, cause, member, holonym, meronym similar, antonym, instance etc.* With these semantic relations, WordNet can be considered as an ontology.

We also use WordNet to lemmatization words. Comparing to other popular stemming algorithms such as Porter's [23] stemming algorithm, WordNet significantly reduces *over-stemming* errors, which could lead to false positive results.

In addition, the set of input concept names $\mathcal{N}(\mathcal{I})$ and output concept names $\mathcal{N}(\mathcal{O})$ in semantic descriptions (OWL-S, WSMO, SAWSDL) are considered for the cloud with non-character symbols removed and converted to lowercase. In the case of keyword-based service descriptions (where no text is included), a plain cloud is created with frequency 1 for every keyword in the description.

Syntactic keywords can be easily obtained from tag clouds (either original or calculated with Δ), by simply adopting the words in the cloud (function $\tau(TC)$, being TC a tag-cloud).

The set of input and output concept parameter types ($pt(\mathcal{I})$ and $pt(\mathcal{O})$) are also adopted as **semantic keywords**.

2.2.2 Category Discovery

Our directory is organized using service's category information based on the North American Industry Classification System (NAICS). Services need to provide at least one NAICS category to be registered in our directory.

Among all service description languages considered by our directory, only OWL-S provides a mechanism to include NAICS category information in the service description, but also commonly ignored by service providers.

To associate an appropriate category with the service, we first extract keywords related to each category from NAICS 2007 Index file. During each service registration, if no category information is provided by the service provider nor defined in the service description, *category extractor* calculates the similarity between keywords extracted from service description and keywords of each NAICS 2007 category to find the most suitable categories for the service.

The similarity is measured by mapping each keyword from both NAICS categories and service description to WordNet synsets, and the similarity is defined as:

$$\frac{|K_S \cap k_c|}{|k_c|}$$

where K_S denotes the keywords extracted from service description S, and k_c denotes sets of keywords of each NAICS 2007 category c.

3 Service Matchmaking

Service matchmaking is an essential part of our service directory. The similarity between two service descriptions (request and advertisement) is based on the similarities of each pair of corresponding elements in their AT-GCMs. Only elements existing in both descriptions are considered, the rest are ignored.

We further classify the elements in AT-CGM into three categories: semantic elements, syntactical elements and category information. Each type of element is associated with an ontology, and a generic ontological similarity algorithm is applied to calculate the similarity between each pair of corresponding elements of service request (S_R) and advertisement (S_A).

- Semantic elements are associated directly with their original ontologies used in the service description.
- Syntactic information is associated with external lexical databases such as WordNet, which can also be considered as an ontology.
- The category of a service is often an element in certain classification systems, such elements are usually organized in a hierarchy, which can be considered as an ontology also.

Table 2 summarizes the AT-GCM components in each category and the associated ontology:

Table 2 Categorizing AT-GCM components

Category	Component	Ontology
Semantic Elements	I_{sem}, O_{sem}, K_{sem}	[From service description]
Syntactic Elements	K_{syn}, I_{syn}, O_{syn}, TC	WordNet
Category Information	C	NAICS-07

Fig. 4 illustrates the complete matching schema.

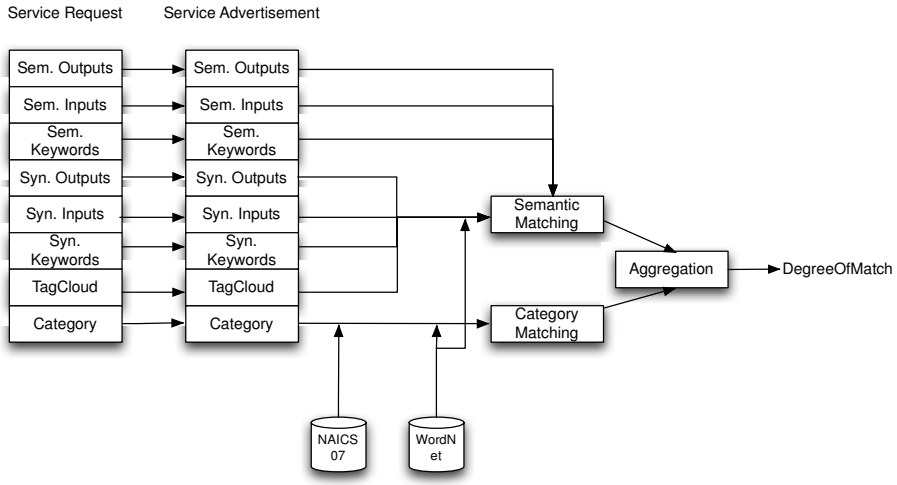


Fig. 4. Service Matchmaking based on AT-GCM

3.1 Semantic Elements Matching

Semantic elements in AT-GCMs include semantic inputs, semantic outputs and semantic keywords. For instance, in an AT-GCM obtained from an OWL-S description, the semantic elements are I_{sem}, O_{sem} and $K_{sem}=(I_{sem} \cup O_{sem})$.

The matching process of semantic concepts in web services takes one concept from service request (C_R) and service advertisement (C_A) and returns their degree of match.

The degree of match between these semantic concepts is based on their subsumption relation in the ontology. In this paper, we adopt the four degrees of match proposed by Paolucci et al. in [19]: *exact (C_A=C_R), plug-in (C_R subsumes C_A), subsumes (C_A subsumes C_R)* and *fail (otherwise)*.

To obtain a numerical similarity between two concepts, we further calculate the length of the *shortest ancestral path* between these two concepts, which was introduced by Y. Li et al. in [15]:

$$sim(C_1, C_2) = \begin{cases} 1 & \text{if } C_1 = C_2 \\ e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{otherwise} \end{cases}$$

where $\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of the shortest path length (l) between the two concepts and the depth (h) of the least common subsumer in the concept hierarchy, respectively.

We combine this function with the four degrees of match commented above into a unique numerical real value between 0 and 1, being *exact = 1, plug-in $\in$ (0.5,1), subsumes $\in$ (0,0.5)* and *fail* = 0:

$$conceptMatch(C_R, C_A) = \begin{cases} 1, & \text{if } C_R = C_A \\ \frac{1}{2} + \frac{1}{2} sim(C_A, C_R), & \text{if } C_R \text{ subsumes } C_A \\ \frac{1}{2} sim(C_R, C_A), & \text{if } C_A \text{ subsumes } C_R \\ 0, & otherwise \end{cases}$$

3.1.1 Semantic Outputs/Inputs

In line with Paolucci's proposal in [19], a semantic output matches if and only if for each output of the request there is a matching output in the service description, i.e. the service provides all the outputs required.

For two sets of semantic outputs, O^R_{sem} and O^A_{sem}, the similarity between these two outputs is calculated using function:

$$OSemMatch(O^R_{sem}, O^A_{sem}) = \begin{cases} 1, & \text{if } |O^R_{sem}| = 0 \\ \underset{o^R \in O^R_{sem}}{\text{Min}} \underset{o^A \in O^A_{sem}}{\text{Max}} (conceptMatch(o^R, o^A)), & \text{otherwise} \end{cases}$$

In function *OSemMatch* O^R denotes the semantic outputs from service request. Therefore, if the service request requires no outputs ($|O^R_{sem}|=0$), it returns 1, exact match, regardless of the outputs produced by service advertisement O^A_{sem}. Otherwise, the semantic match is obtained by taking, for each output in the request, the best

match against the ones in the advertisement. The worst case (minimum value) is then chosen to combine the best matches.

For semantic inputs, an analogous approach is followed, but with the order of the request and advertisement reversed.

3.1.2 Semantic Keywords

For semantic keywords from service request, K^R_{sem} (R) and from service advertisement, K^A_{sem} (A) the degree of match between two sets of semantic keywords is calculated using measure proposed in [5]:

$$KM(R,A) = \frac{\sum_{r\in R}\overrightarrow{r}}{|\sum_{r\in R}\overrightarrow{r}|}\cdot\frac{\sum_{a\in A}\overrightarrow{a}}{|\sum_{a\in A}\overrightarrow{a}|}$$

with $r = (sim(r,r_1), sim(r,r_2)\ldots, sim(r,a_1), sim(r,a_2)\ldots)$, and *a* analogously.

Alternative semantic similarity measures can be used, such as the measure described by Hau et al. in [7].

3.2 Syntactic Elements Matching

Syntactic elements in AT-GCM include syntactic keywords, tag-cloud, syntactic I/Os and text. To achieve uniformity and simplicity, we would like to adopt the similarity measures defined in the last section to suit the syntactic elements too.

However, these elements have no associated ontological concepts explicitly defined in the service description. Thus, these elements need to be mapped into concepts of a certain lexical database with subsumption relation defined, such as *WordNet*.

3.2.1 Syntactic Keywords

Syntactic keywords are first mapped to WordNet synsets, with *hypernym/hyponym* relations defined between synsets, we simply adopt function *KSemMatch* defined in the last section:

$$KSynMatch(R_{syn}, A_{syn})_{WordNet} = \frac{\sum_{r\in R}\delta_r\overrightarrow{r}}{|\sum_{r\in R}\delta_r\overrightarrow{r}|}\cdot\frac{\sum_{a\in A}\delta_a\overrightarrow{a}}{|\sum_{a\in A}\delta_a\overrightarrow{a}|}$$

where $K^R_{synsets}$ and $K^A_{synsets}$ denote WordNet synsets associated with keywords in the service request and service advertisement respectively, and δ denotes weight of a keyword, which is always 1 at current stage.

Similarity between tag-clouds is calculated in the same way with weights (frequencies):

$$TagMatch(R_{syn}, A_{syn})_{WordNet} = \frac{\sum_{r\in R}\delta_r\overrightarrow{r}}{|\sum_{r\in R}\delta_r\overrightarrow{r}|}\cdot\frac{\sum_{a\in A}\delta_a\overrightarrow{a}}{|\sum_{a\in A}\delta_a\overrightarrow{a}|}$$

where δ_r and δ_a denotes the frequency of term *r* (in *R*) and *a* (in *A*) respectively.

3.2.2 Syntactic Inputs/Outputs

Degree of match of WordNet synsets mapped from syntactic inputs and outputs are calculated in the same way as their semantic counterparts.

$$OSynMatch(O^R_{syn}, O^A_{syn})_{WordNet} = \begin{cases} 1, & \text{if } |O^R_{syn}| = 0 \\ \underset{o^R \in O^R_{syn}}{\text{Min}} \underset{o^A \in O^A_{syn}}{\text{Max}} (conceptMatch(o^R, o^A)), & \text{otherwise} \end{cases}$$

3.3 Category Matching

As stated in section 2, our directory uses NAICS 07 as services categorization standard. With 2341 categories in total, NAICS 07 standard organizes these categories in a 5-level hierarchy.

Each category is considered as a concept in this category taxonomy, the calculation of the similarity between two categories is done by using:

$$CatMatch(C_1, C_2) = sim_{NAICS-07}(C1, C2)$$

3.4 Aggregation Function

Finally, service matching must combine the similarity value for each of these fields.

sim_{Isyn}, sim_{Isem} $sim_{Osem,}$ $sim_{Osyn,}$ sim_{TC} , $sim_{Ksyn,}$ sim_{Ksem}, sim_C denote the similarity of syntactic/semantic inputs, syntactic/semantic outputs, tag-cloud, syntactic/semantic keywords and category respectively between a service request and a service advertisement. An aggregation function is a function that combines these similarity values.

For the moment, a general approach is taken: a weighted sum of each similarity, where the weighting parameters are the contribution of the corresponding components of the AT-GCM. The contribution of each component is calculated using a logistic function:

$$w(n_c) = \frac{1}{(1 + e^{(1 - \frac{n_c}{0.5\overline{N}})})}$$

where n_c denotes the number of elements in component C (for example, number of semantic outputs), and $\overline{N}$ denotes the average number of elements in both service models.

Function *w* is a logistic function, which makes the weights of the components with number of elements close to the average increase rapidly. Also, logistic function prevents the over-influence caused by components with excessive number of elements.

4 Implementation and Evaluation

The directory service implementation consists of a web server to perform various operations defined in section 2 (register and search services). The server may be accessible through a web interface implemented on the same server, or through REST operations to receive and respond to customer requests.

We used SQLite[3] database to facilitate the implementation in future distributions of the service directory.

The service directory receives search requests and responds to them through JSON [9] data exchange, including a list of descriptions of the matching services and their corresponding grounding so that they can be invoked if desired.

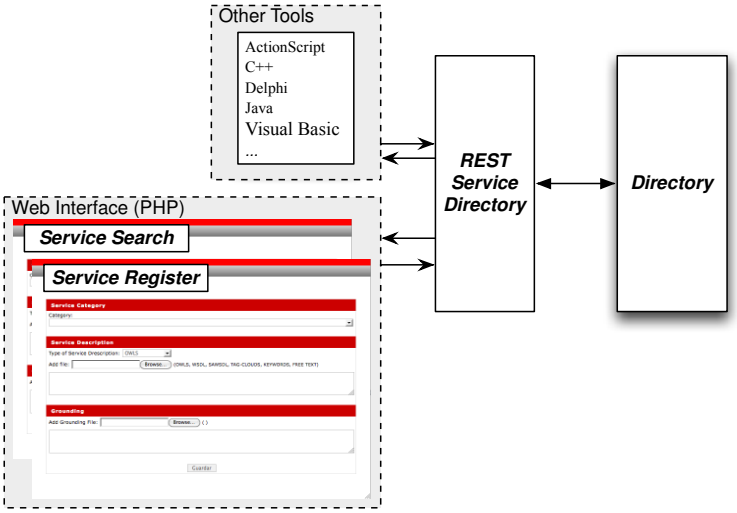


Fig. 5. Service Directory Interaction

The implemented Web Interface also uses REST to interact with the service directory. Fig. 5 shows the interaction of our proposed service directory with the Web Interface and other languages. When the directory receives a client request (GET) it carries out the operation using the specific parameters included in the request and answers using JSON objects. The client can use the received information to show it or invoke the services.

4.1 Evaluation

Based on OWLS-TC[4] 4.0, we performed two experiments to evaluate the precision of category extraction and syntactic keywords matching.

[3] http://www.sqlite.org/
[4] http://www.semwebcentral.org/projects/owls-tc/

As both experiments involve syntactic matching, the relevance is relatively subjective. Therefore, the precision of the results is calculated against human judgement.

Category Extraction

We selected 78 services from the OWLS-TC, and 5 NAICS-07 categories were extracted using techniques described in section 2.2. Then we manually evaluated how many extracted categories were acceptable (agree with human judgement). The measure is essentially a *precision at 5*:

$$precision = \frac{|C_{extracted} \cap C_{human}|}{|C_{extracted}|}$$

where $|C_{extracted}| = 5$.

In comparison, we also performed an experiment in category extraction without WordNet, i.e, character-wise matching was performed over stemmed keywords from service description and category index.

The results showed an average precision of 0.698 from our approach and 0.2734 from using pure syntactic matching.

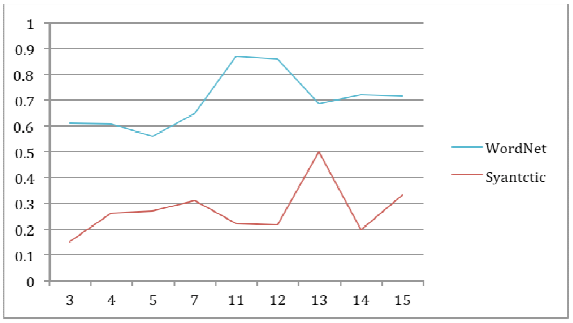


Fig. 6. Precision and number of keywords extracted

In general, the precision of extraction with WordNet is higher than pure syntactic matching. However, Fig. 6 shows that as the number of keywords increases, WordNet approach's precision decrease.

This could be due to the fact that the number of WordNet synsets associated with keywords increases rapidly hence overgeneralized the *domain* of the service An other possible cause could be that the experiment was performed with a relatively small amout of samples, thus noises are very obivious, for example, only one service has 13 keywords extracted and its value cloud be an exceptional extreme value.

Syntactic Keywords Matching

We selected 8 service requests from the OWLS-TC's *Request and Relevance Sets,* and using relevance information provided by OWLS-TC as the benchmark, our syntactic matching algorithm has an average precision of 80.5%.
Again, this results could be not reliable due to the small number of samples used. Therefore, further larger scale experiments will be one of our future works

5 Related Work

Some (not many) other efforts have been made trying to align or compare different service description approaches. As we mentioned in section 2.1, we set out from existing conceptual comparisons between popular semantic web service languages [11, 12, 20, 22] to obtain a general model description of services that facilitates their discovery.

Most of the current approaches to Semantic Web Services matching, particularly those based on OWL-S, are based on subsumption reasoning on concepts included in the descriptions (e.g. [14, 19]). Klusch et. al [10] present a hybrid matchmaker that complements logic based reasoning with approximate matching techniques from Information Retrieval. In this sense we propose a hybrid approach, which combines subsumption checking, concepts similarity, and information retrieval. However, we focus on the integration of several different service description.

The directory service using a common model (AT-GCM) in the same direction as iServe [21] uses the minimum service model to address interoperability, the difference is that our board to consider Tag-Cloud, and keywords free text for use in the directory.

Ambite et al introduced a system (DEIMOS) for constructing semantic web service from online sources automatically in [1]. DEIMOS uses an existing semantic web service as a seed, by calculating the syntactic similarity and a brute-force invocation-observation learning process, DEIMOS semantically annotated an external source. Differently to our approach they use only inputs/outputs to characterise services. Also, they use the Local-As-View (LAV) [13] datalog rules to describe the sources. We use RDF instead, although this does not reduce expressivity against LAV, in fact DEIMOS generates an RDF graph from LAV descriptions.

In addition, A. Heß introduced a web service classification approach using machine-learning techniques in [8]. Even though the evaluation showed a remarkable accuracy, no information about computational efficiency was shown. As techniques such as Naïve-Bayes and SVM could be noticeably computationally expensive, this approach might not be entirely suitable for service discovery in a large, open environment.

6 Conclusion

In this paper we have dealt with the problem of service discovery in open systems. We proposed an architecture that considers the alignment of service description models, and the transformation of them into a unified common model. We do not only consider explicit information specified in structured service descriptions, but we enrich descriptions with additional information extracted using text processing. Although we provided with an alignment mechanism for a set of service description languages, other languages can be easily integrated into. In fact, if such new model fits into the proposed *AT-GCM* only the adequate mappings have to be specified.

Regarding computational aspects, note that the mapping of service advertisements to the *AT-GCM* can be done at registration time, so we only need to process the service request at run time (as well as the matchmaking algorithm).

We also proposed the combination of service matching and concept similarity into an integrated service-matching framework.

The implementation and a preliminary evaluation showed a satisfying result regarding category and keywords extraction. Further evaluations, such as F-measure and recall of extracted categories as well as precision/recall of service are part of our future plans.

7 References

1. Ambite, J.L., Darbha, S., Goel, A., Knoblock, C.A., Lerman, K., Parundekar, R., Russ, T.: Automatically constructing semantic web services from online sources. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 17–32. Springer, Heidelberg, 2009.
2. Balta, A., and A. Fernandez. Towards Service Matchmaking of Heterogeneous Service Descriptions. Worshop on Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE@AAMAS2010), Toronto, 2010.
3. Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., K¨onig-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., and Stollberg, M. Web Service Modeling Ontology (WSMO). W3C Member Submission, 2005. http://www.w3.org/Submission/WSMO/.
4. Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001
5. Ehrig M. Ontology Alignment: Bridging the Semantic Gap. Springer. 2007.
6. Farrell, J. and Lausen, H. Semantic Annotations for WSDL and XML Schema (SAWSDL). W3C Recommendation 28 August 2007. http://www.w3.org/TR/sawsdl/
7. Hau, J., Lee, W., Darlington, J.: A Semantic Similarity Measure for Semantic Web Services. In: Proceedings of the Workshop Towards Dynamic Business Integration co-located with the 14th International World Wide Web Conference (WWW), 2005.
8. Heß, A., Johnston, E., Kushmerick, N.: Machine Learning for Annotating Semantic Web Services. In: Semantic Web Services: Papers from the 2004 AAAI Spring Symposium Series. AAAI Press , 2004.
9. JSON, JavaScript Object Notation. http://www.json.org/, 2011
10. Klusch, M., Fries, B., and Sycara, K. OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 7, Issue 2, April 2009, Pages 121-133.

11. Kourtesis, D., and Paraskakis, I. Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. Springer Berlin / Heidelberg. Pages 614-628. 2008.
12. Lara, R., and Polleres, A. D4.2v0.1 Formal Mapping and Tool to OWL-S, WSMO working draft 17 december 2004. http://www.wsmo.org/2004/d4/d4.2/v0.1/
13. Levy, A.Y.: Logic-based techniques in data integration. In: Minker, J. (ed.) Logic-Based Artificial Intelligence. Kluwer Publishers, Dordrecht, 2000.
14. Li, L. and Horrocks, I. A software framework for matchmaking based on semantic web technology. Int. J. of Electronic Commerce, 8(4):39–60, 2004.
15. Li, Y, Bandar, Z.A. and McLean, D. An approach for measuring semantic similarity between words using multiple information sources. IEEE Trans- actions on knowledge and data engineering, pages 871–882, 2003.
16. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDemott, D., McIlraith, D., Narayanan, D., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K.. OWL-S: Semantic Markup for Web Services. W3C Member Submission, 2004. Available from http://www.w3.org/Submission/OWL-S/.
17. Miller, G.A. WordNet: a lexical database for English, Communications of the ACM, pages 39-41, 1995. Association for Computational Linguistics.
18. NAICS Association. NAICS code searching. http://www.naics.com/search.htm, 2004.
19. Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. Semantic Matching of Web Service Capabilities. In ISWC, pages 333–347. Springer Verlag, 2002.
20. Paolucci, M., Wagner M., and Martin M. Grounding OWL-S in SAWSDL. Springer Berlin / Heidelberg. Pages 416-421. 2007.
21. Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J. and Domingue, J. iServe: a Linked Services Publishing Platform, Workshop: Ontology Repositories and Editors for the Semantic Web at 7th Extended Semantic Web Conference, 2010.
22. Polleres, A., and Lara, R. A Conceptual Comparison between WSMO and OWL-S, WSMO Working Group working draft, 2005. http://www.wsmo.org/2004/d4/d4.1/v0.1/.
23. Porter, M. F. An algorithm for suffix stripping, Program, vol. 14, no. 3, 130-137, 1980.
24. Salton, G. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, 1989.

A Framework for Resource Annotation and Classification in Bioinformatics

Nadia Yacoubi Ayadi[†], Malika Charrad[†], Soumaya Amdouni[‡] and Mohamed Ben ahmed[†]

[†] National School of Computer Science,
University of Manouba, 2010 Tunisia
nadia.yacoubi@asu.edu, malika.charrad@riadi.rnu.tn,
mohamed.benahmed@riadi.rnu.tn
[‡] High Institute of Management, Bardo City, Tunisia

Abstract. Semantic annotation is commonly recognized as one of the cornerstones of the semantic Web. In the context of Web services, semantic annotations can support effective and efficient discovery of services, and guide their composition into workflows. Because semantic annotation is a time consuming and expensive task, (semi-)automatic approaches for semantic annotation extraction are required. In this paper, we propose a semi-automatic extraction approach of lightweight semantic annotations from textual description of Web services. In contrast with most of the existing semi-automatic approaches for semantic annotations of Web services which rely on a predefined domain ontology, we investigate the use of NLP techniques to derive service properties given a corpus of textual description of bioinformatics services. We evaluate the performance of the annotation extraction method and the importance of lightweight annotations to classify bioinformatics Web services in order to bootstrap the service discovery process. Our framework relies an unsupervised clustering approach based on a simultaneous clustering algorithm that enables to determine biclusters of Web services and semantic annotations highly correlated.

Keywords: Semantic Annotation, Semantic Web Service, Block Clustering, Bioinformatics

1 Introduction

During the last decade, semantic Web services (SWS) [20] technology have been proposed and investigated to support effective and efficient service discovery, composition and invocation by machines. Despite the appealing characteristics of semantic Web services principles, their uptake on a Web-scale has been significantly less prominent than initially anticipated [21]. In fact, research on semantic Web services has mostly focused on devising domain-independent Web service description ontologies such as OWL-S [19] and WSMO [22]. Semantic Annotations for WSDL (SAWSDL) [15] adopts a bottom-up approach by adding semantics to existing Web service standards through mapping syntactic definitions to

a set of ontological concepts. All of these approaches rely on a pre-determined domain ontology to explicit service semantics. Reasoning tasks performed with semantic Web service descriptions is mainly conditioned by the quality of this domain ontology [4]. The existence of a domain ontology to capture domain knowledge in an explicit and formal way is crucial. In several fields, many domain ontologies have been developed for several purposes. The complexity of reasoning tasks increases when semantic service descriptions are generated by means of several domain ontologies. In the bioinformatics field, the OBO foundary[1] lists around 60 ontologies for life sciences including molecular biology, anatomy, biochemistry, environment, neuroscience, etc. (for a survey, see [24]). None of these ontologies is suitable to annotate bioinformatics Web services; although, they are *rich in semantics* but not *enough generic* to capture high-level concepts and their semantic relationships.

In this paper, we propose a *bottom-up* approach to extract domain-dependant lightweight semantic annotation from textual description of Web services. Such annotations of Web services aims to capture static (i.e., domain concepts) and procedural knowledge (i.e., tasks) of a domain. Despite their importance, few domain ontologies exist for the purpose of Web services annotation, and thus, building such ontologies is a challenging task. Natural language documentations of Web services are short textual descriptions intended to close the "*semantic gap*" between low-level technical features of Web services (e.g., data types, port types, or data formats) and the high-level, meaning-bearing features a user is interested in and refers to when discovering a Web service. Hence, our semi-automatic approach combines different extraction patterns to generate lighweight annotations describing service properties such as inputs, outputs, or functionnalities. We notice that our extraction method provides a good starting point for ontology building.

Therefore, we rely on a simultaneous clustering algorithm, namely CROKI2 [13], to identify clusters (groups) of services that are described by a specific subset of highly correlated annotations. Simultaneous clustering step has two benefits. Firsly, clustering Web services based on semantic annotations would greatly boost the ability of Web services search engines to select suitable services given a discovery query. Secondly, it enables to detect implicit associations (relationships) between highly correlated annotations which is crucial in an ontology building process. In fact, the co-occurrence of a subset of annotations within a subset of Web services reflects implicit relationships that could be taxonomic or non taxonomic between these annotations. To the best of our knowledge, no approach was developed using block-clustering, however, most of the approaches enables either annotations clustering [16, 1] or services clustering [17, 12].

The paper is organized as follows. The section 2 reviews related work conducted in the fields of automatic annotation of Web services and block clustering. Section 3 presents our framework for semantic annotation and clustering of Web services. In the section 4, we present and discuss the results of our experimentations. Section 5 concludes the paper and outlines our future work.

[1] http://www.obofoundry.org/

2 Related Work

2.1 Semantic annotation learning for Semantic Web services

Converting an existing Web service into a semantic Web service requires significant effort and must be repeated for each new Web service. We review in this section research work that focus on learning semantic annotations by exploiting textual descriptions, WSDL files or even Web forms. Hess and al. proposes ASSAM (Automated Semantic Annotation with Machine Learning), a semi-automatic WSDL annotator application. ASSAM [14] relies on a pre-determined domain ontology and uses a machine learning algorithm to provide users with suggestions on how to describe the elements in the WSDL file. However, because of the intensive expert user intervention, applicability of such solution for large-scale annotation of web services could be impractical despite of the fact that these solutions tend to provide high-quality annotations. Sabou et al. [23] proposes an automatic extraction method based on Natural Language Processing (NLP). Experimentations was conducted in the bioinformatics field by learning an ontology from the documentation of Web services in the context of the ^{my}Grid project. The evaluation of the extracted ontology shows that the approach is a helpful tool to support process of building domain ontologies for Web services. Our approach relies on [23]'s approach by using also NLP processing techniques to generate semantic annotations of Web services.

Also, within the bioinformatics space, Afzal et al. [2] developed a text mining approach based on literature to learn *semantic profile* of bioinformatics resources. The approach identifies a set of semantic classes of descriptors that could be attached to a bioinformatics resource: *data*, *data resource*, *task*, and *algorithm*. The instances of these classes were collected by harvesting a corpus of scientific papers along with related sentences containing the resource name. However, the case study conducted in [2] shows that the coverage broad of the ^{my}Grid ontology used as annotation support is partially limited especially to capture functional service descriptions. The quality of extracted descriptors was only measured from the curator's perspective view which is not accurate in the semantic Web context where Web services are supposed to be discovered and composed by agents.

Ambite and al. [3] present an approach to automatically discover and create semantic Web services. The idea behind their approach is to start with a set of known sources and the corresponding semantic descriptions and then discover similar sources, extract the source data, build semantic descriptions of the sources, and then turn them into semantic Web services. Authors implemented the Deimos system and evaluated it across five domains. In contrast to our work, the goal of Deimos is to build a semantic description that is sufficiently detailed to support automatic retrieval and composition. Our work aims to generate lightweight annotations useful to classify Web services and bootstrap the service discovery process in the bioinformatics field.

2.2 Web service Clustering

With the expectable growth of the number of available Web services and service repositories, the need for mechanisms that enable the automatic organization and discovery of services becomes increasingly important. In this context, most of the existing research rely on a one-way clustering, either annotations clustering [16, 1] or services clustering [12, 17]. When clustering algorithms are used, each service in a given services cluster is described using all annotations. Similarly, each annotation in an annotation cluster characterizes all services. For instance, Based on their approach presented in [2], Afzal and al. propose in [1] to use lexical kernel metrics to identify semantically related networks of resources by computing similarity between annotations. However, the goal of our work is to identify groups of services that are more described by a specific subset of annotations which refers to find biclusters of services and annotations highly correlated in order to bootstrap the service discovery process. We rely on simultaneous clustering which is an approach enabling to find local pattern where a subset of subjects might be similar to each other based on only a subset of attributes. Simultaneous clustering, usually designated by biclustering, co-clustering or block clustering aims to find sub-matrices, which are subgroups of rows and subgroups of columns that exhibit a high correlation. A number of algorithms that perform simultaneous clustering on rows and columns of a matrix have been proposed to date. This type of algorithms has been proposed and used in many fields, such as bioinfomatics [18], Web mining [8] and text mining [6]. Table 1 outlines a comparison between one-way clustering and simultaneous clustering.

Table 1. Comparison between Clustering and Simultaneous clustering

Clustering	**Simultaneous Clustering**
- applied to either the rows or the columns of the data matrix **separately** ⇒ **global model**.	- performs clustering in the two dimensions **simultaneously** ⇒ **local model**.
- produce **clusters** of rows **or** clusters of columns.	seeks **blocks** of rows **and** columns that are interrelated.
- Each subject in a given subject cluster is defined using **all** the variables. Each variable in a variable cluster characterizes **all** subjects.	- Each subject in a bicluster is selected using **only** a subset of the variables and each variable in a bicluster is selected using **only** a subset of the subjects.
- Clusters are **exhaustive**	- The clusters on rows and columns **should not be exclusive** and/or **exhaustive**

3 General Framework

The proposed framework is comprised of two main steps. The first one aims to perform a semi-automatic semantic annotation extraction from Web services textual documentations. Semantic annotations enables to describe service properties

such as functionalities, inputs, outputs, and other domain-dependant features. One particluarity of textual Web service description is that they employ natural language in a specific way. In fact, such texts belong to what was defined as sublanguages [23]. A sublanguage is a specialized form of natural language which is used within a particular domain and characterized by a specialized vocabulary, semantic relations, and syntax (e.g., medical test report). The semantic annotation extraction step exploits the linguistic regularities of a sublanguage to identify semantic service properties. The second step of our approach consists on Web service clustering in terms of semantic annotations. This step allows to discover subgroups (biclusters) of Web services and subgroups of semantic annotations that exhibit a high correlation by applying the CROKI2 algorithm [13]. In following, we present in further details the two steps.

3.1 Semantic Annotation Extraction of Web services

The semantic annotation extraction phase allows to identify two types of knowledge: domain concepts and procedural knowledge describing services tasks. First, a morphosyntactic analysis of textual description of Web services is performed. In this step, a sentence splitter and a tokeniser components are used to extract sentences and basic linguistic entities. Then, a POS (Part-Of-Speech) Tagger is performed to associate to each word (token) a grammatical category and thus distinguish the morphology of various entities. For example, the sentence below, the tagger identify a verb (i.e., *compute*), three nouns (i.e., *structure*, *RNA*, *sequence*), an adjective (i.e., *secondary*), and a preposition (i.e., *for*).

```
compute (VB) Secondary (JJ) Structure (NN) for (Prep) RNA (NN) sequence (NN).
```

We distinguish different types of syntactic patterns depending on the semantic annotation type. Syntactic patterns describe selectional constraints that exploit sublanguages particularities. We distinguish syntactic patterns that allow to extract inputs and outputs of services, services tasks, and domain-dependant features which are strongly related to the bioinformatics domain:

1. **Identifying service tasks is crucial for the service discovery and composition issue.** We observed that, in majority of textual descriptions of Web services, verbs identify the functionnality performed by a Web service. In our work, we consider different classes of verbs which inform on the service task. For example, *VBRetrieval* is the class of verbs that indicates a retrieval process (e.g., *get, retrieve, fetch, search, find, return, query*). A frequently occuring pattern which involves this verbs class and the preposition *from* can be used to easily determine the output and the retrieved resource as described by the following selectional pattern:

$$VBRetrieval <Output> \ from <Source>.$$

Other verb classes were recognized, such as *VBExtraction* which is a class of verbs denoting an extraction process, *VBExtraction*={*extract, scan, identify, locate, analyse*}.

2. **Identifying inputs and outputs of Web services.** Inputs and outputs of Web services denote domain concepts which are generally depicted by nouns in the corpus. However, to get high-quality annotations, we create a list of biological terms comprised by a set of single word terms. When two or more biological concepts are used together, we interpret them as a single biological concept and update the list by adding it, i.e., *gene expression*, *transcription factors*, *protein structure*, *tertiary protein structure*, *amino acid sequence*, *chromosome segment*, etc. We define different heuristics that identify the roles of concepts (input or output) depending on the structure of the sentence. Some extraction patterns are presented in Table 2. Therefore, our extraction patterns identifies cases when several concepts are related via logical operators such as "*and*", "*or*". In this case, the same role is assigned to each concept.

Table 2. Examples of Extraction Patterns identifying inputs and outputs of Web services

Extraction Pattern
accepts|**consumes**|**takes input**|**requires**|**Operates On** % <InputService>
VBRetreival|**build** % <OutputService> **given**|**for** % <InputService>
% **Given** <InputService> %
% **returns** <OutputService> %
% <OutputService> **is returned** %
% **compares** <InputService> to <InputService> %
% **compares** <InputService> **against** %

3. **Identifying domain-dependant features.** We define a set of extraction patterns that focus on bioinformatics-dependant features. For example, we propose patterns to identify data formats (e.g., FASTA, GFF, GIF, etc.) related to inputs/outputs formats. An example of such patterns is described as follows: % **computes** <OutputService> **for** % <InputService> **described with** <dataFormat> %.

3.2 Web services Clustering

We propose to use a simultaneous clustering approach to classify Web services in terms of semantic annotations. Our approach aims to find biclusters of Web services and annotations by applying CROKI2 algorithm [13]. We propose an accelerated version of this algorithm in [7]. The general purpose of a block clustering algorithm is described as follows. Given the data matrix A, with set of rows $X = (X_1, ..., X_n)$ and set of columns $Y = (Y_1, ..., Y_n)$, a_{ij}, $1 \leq i \leq n$ and

$1 \leq j \leq n$ is the value in the data matrix A corresponding to row i and column j. Simultaneous clustering algorithms aim to identify a set of biclusters $B_k(I_k, J_k)$, where I_k is a subset of the rows X and J_k is a subset of the columns Y. I_k rows exhibit similar behavior across J_k columns, or vice versa and every bicluster B_k satisfies some criteria of homogeneity.

Croki2 algorithm. The Croki2 algorithm is applied to the contingency table composed of services and annotations to identify a row partition $P = (P_1, ..., P_K)$ composed of K clusters and a column partition $Q = (Q_1, ..., Q_L)$ composed of L clusters that maximizes $\mathcal{X}^2$ value of the new contingency table (P,Q) obtained by regrouping rows and columns in respectively K and L clusters. Croki2 consists in applying K-means algorithm on rows and on columns alternatively to construct a series of couples of partitions (P^n, Q^n) that optimizes Chi2 value of the new contingency table $T_1(P, Q)$ defined by this expression:

$$T_1(k, l) = \sum_{i \in P_k} \sum_{j \in Q_l} a_{ij}$$

$k \in [1, ..., K]$ and $l \in [1, ..., L]$.
Marginal frequencies in table T1 are :

$$f_{kl} = \sum_{i \in P_k} \sum_{j \in Q_l} f_{ij}$$

$$f_{k.} = \sum_{i \in P_k} f_{i.}$$

$$f_{.l} = \sum_{j \in Q_l} f_{.j}$$

Biclusters validity. The application of Croki2 algorithm leads to an exhaustive enumeration of biclusters. It is possible to select only biclusters satisfying certain criteria such as a user-specified bicluster size, bicluster homogeneity and bicluster relevancy [13].

– Homogeneity H is the inertia conserved by the bicluster divided by the initial inertia.

$$H = B_{kl}/T_{kl}$$

$$T_{kl} = \sum_{i \in P_k} \sum_{j \in Q_l} f_{i.} f_{.j} (f_{ij}/f_{i.} f_{.j} - 1)^2$$

and

$$B_{kl} = g_{k.} g_{.l} (g_{kl}/g_{k.} g_{.l} - 1)^2$$

The value of this ratio is between 0 and 1. A high value of this ratio indicates that the bicluster is homogenous.

– Relevancy R is the inertia conserved by the bicluster divided by the global inertia.

$$R = B_{kl}/B$$

$$B_{kl} = g_{k.}g_{.l}(g_{kl}/g_{k.}g_{.l} - 1)^2$$

$$B = \sum_{k,l} B_{kl}$$

This ratio indicates whether the bicluster is relevant.

4 Experimentations

4.1 Experimental Dataset

Our experimental corpus consists of 100 bioinformatics services descriptions from the biocatalogue[2], a new curated life science Web services repository. The development of Biocatalogue shows the dramatic increase of bioinformatics Web services and tools with 2053 services and 148 providers[3]. Biocatalogue allows users to discover Web services through keyword-based retrieval or category browsing. Annotations manually attached to Web services are either textual descriptions or lists of tags. Tagging Web services with a set of lexical tokens defined by users is not a perfect way to enable an efficient service discovery. Manual resource tagging is an error prone and time consuming task. Figure 1 shows the top-20 tags used on biocatalogue. In total, 951 tags were created by users to describe services. The use of tags to describe Web services raises several issues such as the ambiguity of their significance (e.g., `BioMoby` or `soaplab` in Figure 1), the variability of the spelling for several tags that may refer to the same concept. Finally, the lack of explicit knowledge representations in folksonomies (a set of tags) to express whenever the tag describes for example a service task, service input or output which prevents their use towards a significant resource discovery. In our work, Web services are semantically annotated based on their textual descriptions. Extracted semantic annotations enable to automatically construct a semantic service profile. In following, we evaluate respectively the annotation extraction module and the block clustering algorithm.

4.2 Annotation Extraction Performance

We designed an annotation extraction module using the GATE [10] framework. We used the ANNIE plugin (A Nearly-New IE system) which contains a tokeniser, a gazetteer (system of lexicons), a POS Tagger, a sentence Splitter, and a Named Entity (NE) transducer. The various extraction patterns described in section 3.1. were implemented using JAPE [11], a rich and flexible rule mechanism which is part of the GATE framework. The NE transducer applies JAPE

[2] http://www.biocatalogue.org
[3] Last Access on 22^{th} april 2011

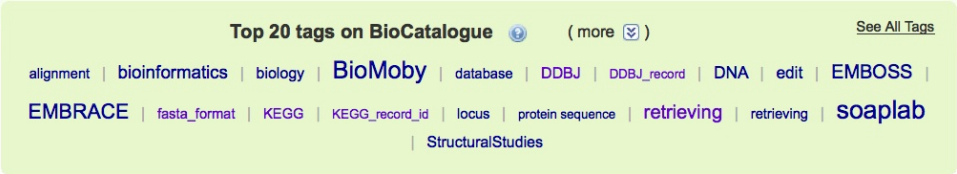


Fig. 1. Top-20 tags in Biocatalogue

rules to input service descriptions in order to generate semantic annotations. Indeed, JAPE (Java Annotation Patterns) engine provides finite state transduction over annotations based on regular expressions. A JAPE grammar consists of a set pattern/action rules. A JAPE rule has a Left-Hand-Side (LHS) and a Right-hand-Side (RHS). The LHS specifies the annotation pattern that may contain regular expression operators (e.g., *, ?, +). The RHS consists of annotation manipulation statements. Annotations matched on the LHS of a rule are referred to on RHS by means of labels that are attached to patten elements. The gazetteer lookup modules, part of the JAPE engine, enable to identify domain concepts in the textual description based on a set of lists of tokens. We have created different lexicons lists containing *bioconcepts*, *service tasks*, *dataformats* and *identifiers* (e.g., EntrezGene_ID, KEGG_ID). Figure 2 illustrates an example of JAPE rule for input service annotation.

We evaluate the results of our experimentations in terms of three metrics: *precision*, *recall* and *F-measure* as depicted in Table 3. The three metrics are calculated as follows.

$$Precision = \frac{Correct + 1/2Partial}{Correct + Spurious + 1/2Partial}$$

$$Recall = \frac{Correct + 1/2Partial}{Correct + Missing + 1/2Partial}$$

$$F - measure = \frac{(\beta^2 + 1)P * R}{\beta^2 R + P}$$

GATE provides an automatic tool for automatic evaluation, named *AnnotationDiff* to compare a set of annotations generated manually and the set of the annotations generated by our extraction method. To measure the performance of the extraction method, we manually identified semantic annotations from the service descriptions corpus. Then, using the *AnnotationDiff* Tool, we compared this set of annotations with the ones that were extracted through extraction patterns.

```
Phase: input
Input: Lookup Token
Options: control = appelt
Rule: input_service

(
({ Token.string == "accept" }
{ Lookup.majorType == bioconcept })
|
({ Token.string == "create" }
{ Lookup.majorType == bioconcept }
{ Token.string == "from" })
|
({ Token.string == "bind" }
{ Lookup.majorType == bioconcept }
{ Token.string == "with" })
|
({ Token.string == "compute" }
{ Lookup.majorType == bioconcept }
{ Token.string == "from" }|{ Token.string == "for" })
|
({ Token.string == "Alignment of" }
{ Lookup.majorType == bioconcept })
|
({ Token.string == "convert" }|{ Token.string == "translate" }
{ Lookup.majorType == dataformat })

)

:label
-->
:label.Input = {rule = "input_service"}
```

Fig. 2. An example of a JAPE rule

Table 3. Precision, Recall and F-measure

Annotation Type	Precision	Recall	F-measure
Service Name	1	0.83	0.90
Service Input	0.9	0.87	0.88
Service Output	0.9	0.87	0.88
Service Task	0.95	0.97	0.95

4.3 Block Clustering Evaluation

The application of Croki2 algorithm leads to an exhaustive enumeration of biclusters. The data used to evaluate the Croki2 algorithm consists on 98 services and 78 annotations only. The choice of meaningful ones is based on homogeneity and Relevancy as described in the previous section. Given that CROKI2 algorithm uses k-means to cluster rows and columns, the number of clusters needs to be specified by user. Therefore, we extend the use of some validity indices, namely BH [5], proposed initially for one-way clustering to CROKI2 biclustering algorithm [9, 7]. Accelerated CROKI2 algorithm have been implemented in R environment.

Bicluster 1

Services	Annotations
ConsensusPathDB	ChemicalSubstance
getColoredKeggPathwayOfKeggIds	Compound
getKeggCompoundsOnKeggPathway	KEGG
getKeggIdsByKeggPathway	Pathway
getKeggPathwayAsGif	proteinInteraction
getKeggPathwaysByKeggID	
getMetaboCardIDs_by_PathwayService	
getPubChemSubstanceIdByKeggCompound	
getUniprotIdentifiersByKeyword	

Bicluster 2

Services	Annotations
EmbossMatcher	DNASequence
EmbossNeedle	PairwiseSequenceAlignment
EmbossWater	ProteinSequence

Bicluster 3

Services	Annotations
roseImplementationService	PhylogenicTree
runPhylipDnaml	Phylogeny
runPhylipProtpars	

Bicluster 4

Services	Annotations
runMatScanGFF	transcriptionFactor
runMatScanGFFCollection	GFF
	DNASequence

Bicluster 5

Services	Annotations
runFastaForNucleotides	six-frameTranslation
runFastx	SequencePairwiseAlignment
runTFasty	NucleotideSequence
runWUTBlastn	SequenceAlignment
runNCBIBlastnXML	ProteinSequence
runNCBITBlastxXML	
runNCBIBlastpXML	

Bicluster 6

Services	Annotations
Annotate3D	RNASecondaryStructure
Predict2D	RNASequence
Plot2D	RNATertiaryStructure

Fig. 3. Example of biclusters

Table 4. Biclusters and their corresponding Relevance and Homogeneity

Bicluster	Relevancy	Homogeneity
1	6%	37%
2	9%	100%
3	7%	100%
4	8%	100%
5	10%	54%
6	9%	100%

Best biclusters have high values of homogeneity and relevancy (fig.3 and Table 4). For example, biclusters 2, 3, 4 and 6 are the most homogeneous (H=100%)

and bicluster 5 is the most relevant (R=10%). Services and annotations that compose each selected bicluster are highly correlated. Each service in a bicluster is described by a subset of annotations and each annotation in a bicluster describe only services belonging to the same bicluster. All biclusters are significant from the bioinformatics view. For example, bicluster 1 is comprised by services related to pathway and protein interactions, bicluster 2 is composed of services related only to pairwise sequence alignment, in contrast with bicluster 5 which is comprised by services related to pairwise and multiple sequence alignment.

5 Conclusion

This work is part of our ongoing research work. We propose a semi-automatic approach to learn lightweight semantic annotations given a corpus of textual descriptions of Web services. The conducted experimentations show that the approach allows to generate high-quality annotations, mostly because of the fine-grained extraction rules of the approach and the regularity of the sublanguage used to describe Web services in the bioinformatics domain. Our approach consists on a good starting point towards building domain ontologies. As future work, we aim to develop a methodology of domain ontologies building devoted to semantic annotations of Web services by harvesting textual descriptions, WSDL files, and even existing domain ontologies. The main goal of the methodology would be the automatic construction of semantic Web services. Therefore, one motivation of this work is to facilitate the resource discovery within the bioinformatics domain. Thus, we rely on a block clustering algorithm to determine a set of biclusters of services coupled with a set of semantic annotations highly correlated. The results demonstrate the potential of block clustering to model the relatedness between both resources and annotations which is very prominent in the context of service discovery.

References

1. Hammad Afzal, James Eales, Robert Stevens, and Goran Nenadic. Mining semantic networks of bioinformatics e-resources from the literature. In *Semantic Web Applications and Tools for Life Sciences (SWAT4LS) Workshop*, 2009.
2. Hammad Afzal, Robert Stevens, and Goran Nenadic. Mining Semantic Descriptions of Bioinformatics Web Resources from the Literature. In *Proceedings of European Semantic Web Conference*, pages 535–549, 2009.
3. José Luis Ambite, Sirish Darbha, Aman Goel, Craig A. Knoblock, Kristina Lerman, Rahul Parundekar, and Thomas A. Russ. Automatically constructing semantic web services from online sources. In *International Semantic Web Conference*, volume 5823 of *Lecture Notes of Computer Science*, pages 17–32. Springer, 2009.
4. Nadia Yacoubi Ayadi, Zoé Lacroix, and Maria-Esther Vidal. Bionmap: a deductive approach for resource discovery. In *Proceedings of International Conference on Information Integration and Web-based Applications Services (iiWAS'08)*, pages 477–482. ACM, 2008.

5. Frank B. Baker and Lawrence J. Hubert. Measuring the power of hierachical cluster analysis. *Journal of the American Statistical Association*, pages 31–38, 1975.
6. Charles-Edmond Bichot. Co-clustering Documents and Words by Minimizing the Normalized Cut Objective Function. *Journal of Mathematical Modelling and Algorithms (JMMA)*, 9(2):131–147, June 2010.
7. Malika Charrad. *Analyse croisée des sites Web par des méthodes de bipartitionnement*. Editions Universitaires Européenne, 2011.
8. Malika Charrad, Yves Lechevallier, Mohamed Ben Ahmed, and Gilbert Saporta. Block clustering for web pages categorization. In *Proceedings of Intelligent Data Engineering and Automated Learning (IDEAL'2009)*, number 5788 in Lecture Notes in Computer Science, pages 260–267. Springer, 2009.
9. Malika Charrad, Yves Lechevallier, Mohamed Ben Ahmed, and Gilbert Saporta. On the number of clusters in block clustering algorithms. In *Proceedings of FLAIRS Conference*. AAAI Eds, 2010.
10. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.
11. H. Cunningham, D. Maynard, and V. Tablan. JAPE : a java annotation patterns engine (second edition). department of computer science, university of sheffield, 2000.
12. Khalid Elgazzar, Ahmed E. Hassan, and Patrick Martin. Clustering wsdl documents to bootstrap the discovery of web services. In *Proceedings of IEEE International Conference on Web Services (ICWS'10)*, pages 147–154, 2010.
13. G. Govaert. *Classification croisée*. PhD thesis, Paris 6, 1983.
14. Andreas He, Eddie Johnston, and Nicholas Kushmerick. ASSAM: A tool for semi-automatically annotating semantic web services. In *Proceedings of International Semantic Web Conference (ISWC'04)*, volume 3298 of *LNCS*, pages 320–334, 2004.
15. Jacek Kopecky, Tomas Vitvar, Carine Bournez, and Joel Farrell. SAWSDL: Semantic annotations for WSDL and XML schemas. *IEEE Internet Computing*, 11(6):60–67, 2007.
16. Victor Kunin and Christos A. Ouzounis. Clustering the annotation space of proteins. *BMC Bioinformatics*, 6:24, 2005.
17. Jiangang Ma, Yanchun Zhang, and Jing He. Efficiently finding web services using a clustering semantic approach. In *Proceedings of Context enabled source and service selection, integration and adaptation Workshop*, pages 51–58. ACM, 2008.
18. SC. Madeira and AL. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, pages 24–45, 2004.
19. David Martin, Mark Burstein, Drew Mcdermott, Sheila Mcilraith, Massimo Paolucci, Katia Sycara, Deborah L. Mcguinness, Evren Sirin, and Naveen Srinivasan. Bringing semantics to web services with OWL-S. *World Wide Web*, 10(3):243–277, 2007.
20. Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *IEEE Intelligent Systems*, 16:46–53, 2001.
21. C. Pedrinaci and J. Domingue. Toward the next wave of services: Linked services for the web of data. *Journal of Universal Computer Science*, 16(13):1694–1719, 2010.
22. Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubén Lara, Michael Stollberg, Polleres, Cristina Feier, Cristoph Bussler, and Dieter Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.

23. Marta Sabou, Chris Wroe, Carole Goble, and Gilad Mishne. Learning domain ontologies for web service descriptions: an experiment in bioinformatics. In *Proceedings of the 14th international conference on World Wide Web*, pages 190–198. ACM, 2005.
24. Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J. Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J. Mungall, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H. Scheuermann, Nigam Shah, Patricia L. Whetzel, and Suzanna Lewis. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, November 2007.

LDM: Link Discovery Method for new Resource Integration

Nathalie Pernelle[1] and Fatiha Saïs[1]

LRI(CNRS UMR 8623 & Paris-Sud 11 University), INRIA Saclay,
4 rue Jacques Monod, Parc Club Orsay Université, F-91893 Orsay Cedex, France
{Nathalie.Pernelle, Fatiha.Sais}@lri.fr

Abstract. In this paper we address the problem of resource discovery in the Linked Open Data cloud (LOD) where data described by different schemas is not always linked. We propose an approach that allows discovery of new links between data. These links can help to match schemas that are conceptually relevant with respect to a given application domain. Furthermore, these links can be exploited during the querying process in order to combine data coming from different sources. In this approach we exploit the semantic knowledge declared in different schemas in order to model: (i) the influences between concept similarities, (ii) the influences between data similarities, and (iii) the influences between data and concept similarities. The similarity scores are computed by an iterative resolution of two non linear equation systems that express the concept similarity computation and the data similarity computation. The proposed approach is illustrated on scientific publication data.

1 Introduction

The appearance of Web of documents (WWW) [1] has upset the way we create and share knowledge by breaking down barriers of publishing and accessing documents. Hypertext links allow users to navigate on the graph of documents and Web search engines to index the documents and answer to user queries. However, hyperlinks do not express explicit links between the various entities described in Web of documents. With the initiative of Open Linked Data cloud [3], the number of data providers on the Web is in a continuous growth leading to a global data space of billions of assertions where data and documents can be linked. However, until now the published data is very heterogeneous in the sense that it is incomplete, inconsistent, described according to different schemas and contains duplicates. In order to be able to automatically exploit this huge amount of heterogeneous data, an important work integration must be performed.

In this paper we focus our interest on the problem of resource discovery in the Linked Open Data cloud (LOD) where data described by different schemas is not always linked. We propose an approach[1] that allows discovery of new links between data. These links can help to match schemas that are conceptually relevant with respect to a given application domain.

[1] in the setting of the ANR (the French National Research Agency) project GeOnto.

Ontology alignment plays a key role for semantic interoperability of this data. Many approaches have been proposed for automatically identifying mappings between elements (concepts and relations) described in heterogeneous ontologies [18, 14]. These approaches may exploit lexical and structural information, user inputs, prior matches or external resources. When concept and relation instances are available, it is also possible to exploit them to find more mappings between ontologies. In [7], the common instances of concepts are exploited to compute mappings between concepts. Since, data is not described using the same URIs even when it describes the same entities, these common instances cannot be obtained straightforwardly. Conversely, discovering that two pieces of data refer to the same world entity is also a key issue for data integration. We propose an approach which simultaneously addresses both problems of ontology alignment and data linking. Thus, the results of data linking step is exploited to improve the results of ontology alignment step and vice versa. These two steps are performed alternatively until a fix point is reached. The two methods exploit the semantic knowledge that is declared in different schemas (ontologies) in order to model: (i) the influences between concept similarities, (ii) the influences between data similarities, and (iii) the influences between data and concept similarities. The similarity scores are computed using an iterative resolution of two non linear equation systems that express, respectively, the concept similarity computation and the data similarity computation.

Applying this approach allows one to infer mappings of equivalence between concepts of different schemas as well as to infer *owl:same-as* relations between instances that refer to the same entity. The obtained schema mappings allow discovery new resources and inferring if they are relevant with respect to a given application domain.

The paper is organized as follows: in section 2 we present the related work in data linking and ontology reconciliation fields. In section 3, we present the ontology and data model and give a short presentation of N2R method on which our work relies. Section 4 presents the proposed approach of link discovery. Finally, we conclude and give some future work in section 5.

2 Related Work

We denote by "*web data*" the network formed by the set of structured datasets described in RDF (Resource Description Framework) and linked by explicit links. Large amount of structured data have been published, including in the project Linking Open Data cloud (LOD).

Datasets are expressed in terms of one or several ontologies for establishing the vocabulary describing data. Web data requires linking together the various sources of published data. Given the big amount of published data, it is necessary to provide methods for automatic data linking. Several tools [17, 13, 10] have recently been proposed to solve partially this problem, each with its own characteristics. For instance, [10] have developed a generic framework for integrating linking methods in order to help users finding the link discovery methods that are more suitable for their relational data. They introduced LinQL, an extension of SQL that integrates querying with string matching (e.g. weighted jaccard measure) and/or semantic matching (i.e. using synonyms/hyponyms) methods. This approach takes advantage of the DBMS query engine

optimizations and it can easily be used to test elementary similarity measures. Nevertheless, this approach is not designed to propagate similarity scores between entities, i.e. their approach is not global.

Some other works address the problem of link discovery in the context semantic Web services. In [11], the authors propose to match a user request with semantic web service descriptions by using a combination of similarity measures that can be learnt on a set of labeled examples.

Our proposal in this paper can also be compared to approaches studying the reference reconciliation problem, i.e., detecting whether different data descriptions refer to the same real world entity (e.g. the same person, the same paper, the same protein). Different approaches have been proposed. [5, 19, 2, 6] have developed supervised reference reconciliation methods which use supervised learning algorithm in order to learn parameters and help the duplicate detection. Such supervised approaches cannot be used in contexts where data amount is big and data schemas are different and incomplete.

In [16] we have developed an automatic method of reference reconciliation which is declarative and unsupervised reference reconciliation method. Besides, in this method we assumed that the data sets conform to the same schema, i.e. the problem of ontology reconciliation is already solved. Some ontology reconciliation approaches [7, 12] have proposed to exploit a priori reconciled instances in the ontology reconciliation process. When we aim at online reference and ontology reconciliation in the context of Linked Open Data, we cannot use these traditional reconciliation approaches, where solving the problem of reference reconciliation assumes the resolution of the ontology reconciliation and vice versa. Furthermore, up to our knowledge, there is no approach which deals with the two problems of discovering links in the ontology level and in the data level, simultaneously.

3 Preliminaries

In this section we will present the ontology and data model that we consider in this work. We will then present the Numerical method for Reference Reconciliation (N2R) [16] on which relies our link discovery approach.

3.1 Ontology and its Constraints

The considered OWL ontology consists of a set of concepts (unary relations) organized in a taxonomy and a set of typed properties (binary relations). These properties can also be organized in a taxonomy of properties. Two kinds of properties can be distinguished in OWL: the so-called relations (*owl:objectProperty*), the domain and the range of which are concepts and the so-called attributes (*owl:DatatypeProperty*), the domain of which is a concept and the range of which is a set of basic values (e.g. Integer, Date, Literal). In Figure 1, we give an extract $O1$ of the ontology that is used to describe the RDF data of the local data source of publications (see source 1 Figure 2) which we will use to illustrate our proposal.

We allow the declaration of constraints expressed in OWL-DL or in SWRL in order to enrich the domain ontology by additional and useful knowledge. The constraints that we consider are of the following types:

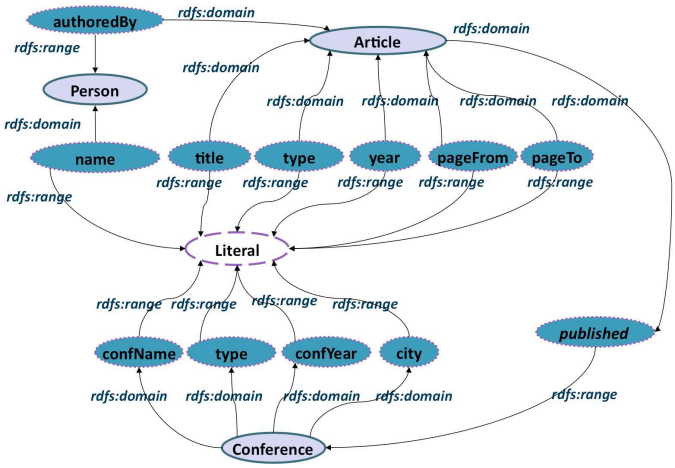


Fig. 1. An extract O1 of the local Ontology for publications

Source S1:
Article(S1_a1); title(S1_a1,"Implementing the TEA algorithm on sensors"); Person(S1_p1); Person(S1_p2); year(S1_a1, "2004"); name(S1_p1,"Olga V. Gavrylyako"); name(S1_p2,"Shuang Liu"); pageFrom(S1_a1,"64"); pageTo(S1_a1,"69");

Conference(S1_c1); confName(S1_c1, "Proceedings of the 42nd Annual Southeast Regional Conference, 2004, Huntsville, Alabama, USA, April 2-3, 2004"); confYear(S1_c1, "2004"); city(S1_c1, "Alabama")
authoredBy(S1_a1,S1_p1); authoredBy(S1_a1,S1_p2); published(S1_a1,S1_c1);

Article(S1_a2); title(S1_a2,"Weighted Hyper-sphere SVM for Hypertext Classification"); Person(S1_p3); Person(S1_p4); year(S1_a2, "2008"); name(S1_p3,"Shuang Liu"); name(S1_p4,"Guoyou Shi"); pageFrom(S1_a2,"733"); pageTo(S1_a2,"740");

Conference(S1_c2); confName(S1_c2, "Advances in Neural Networks - ISNN 2008, 5th International Symposium on Neural Networks, ISNN 2008, Beijing, China, September 24-28, 2008, Proceedings, Part I"); confYear(S1_c2, "2008") city(S1_c2, "Beijing") authoredBy(S1_a2,S1_p3); authoredBy(S1_a2,S1_p4); published(S1_a2,S1_c2);

Source S2:
Article(S2_a1); title(S2_a1,"Implementing the TEA algorithm on sensors.'); Person(S2_p1); Person(S2_p2); year(S2_a1, "2004"); name(S2_p1,"Olga V. Gavrylyako"); name(S2_p2,"Shuang Liu"); pageFrom(S2_a1,"64"); pageTo(S2_a1,"69");

Conference(S2_c1); confName(S2_c1, "42nd Annual Southeast Regional Conference, 2004"); confYear(S2_c1, "2004"); city(S2_c1,"Alabama") authoredBy(S2_a1,S2_p1); authoredBy(S2_a1,S2_p2); published(S2_a1,S2_c1);

Fig. 2. an extract of RDF data

– Constraints of disjunction between concepts: DISJOINT(C,D) is used to declare that the two concepts C and D are disjoint. In the ontology O_1 we declare that all the concepts $Article$, $Conference$ and $Person$ are pairwise disjoint.

- Constraints of functionality of properties: PF(P) is used to declare that the property P (relation or attribute) is a functional property. In O_1, we declare that all the properties are functional except the relation $authoredBy$ which means that one article may have several authors.
- Constraints of inverse functionality of properties: PFI(P) is used to declare that the property P (relation or attribute) is an inverse functional property. These constraints can be generalized to a set $\{P_1, \ldots, P_n\}$ of relations or attributes to state a combined constraint of inverse functionality that we will denote PFI($P_1, \ldots, P_n$). In $O1$, we declare that the combinations ($title, year$) and ($confName, confYear$) are inverse functional. For example, $PFI(title, year)$ expresses that one title and one year cannot be associated to several articles (i.e. both are needed to identify an article).

3.2 Data description and its constraints.

A piece of data has a reference, which has the form of a URI (e.g. `http://dblp.l3s.de/d2r/resource/authors/A._Joe_Turner`), and a description, which is a set of RDF facts involving its reference. An RDF fact can be either: (i) a concept-fact $C(i)$, where C is a concept and i is a reference, (ii) a relation-fact $R(i1, i2)$, where R is a relation and $i1$ and $i2$ are references, or (iii) an attribute-fact $A(i, v)$, where A is an attribute, i a reference and v a basic value (e.g. integer, string, date). We consider the Unique Name Assumption (UNA) which can be declared or not on a data source. Declaring UNA on a data source means that two different data descriptions having two different references, then we infer that they refer to distinct entities.

The data description that we consider is composed of RDF facts coming from the data sources which are enriched by applying the OWL entailment rules. Figure 2, provides examples of data coming from two RDF data sources S1 and S2, which conform to the same ontology describing the scientific publication domain previously mentioned.

In the N2R method which we will present in section 3.3, we consider that the descriptions of data coming from different sources conform to the same OWL ontology (possibly after ontology reconciliation). In the link discovery method, that we will present in section 4, the assumption of prior ontology reconciliation is not fulfilled, i.e. the considered data source do not conform to the same ontology.

3.3 N2R: a Numerical method for Reference Reconciliation

N2R is a numerical method which allows inferring reconciliation decisions between reference coming from different sources that conform to the same ontology, i.e. the problem on ontology reconciliation is already solved.

N2R [16] has two main distinguishing characteristics. First, it is fully unsupervised: it does not require any training phase from manually labeled data to set up coefficients or parameters. Secondly, it is based on equations that model the influences between similarities. In the equations, each variable represents the (unknown) similarity between two references while the similarities between values of attributes are expressed by constants. These constants are obtained, either by (i) exploiting a dictionnary of synonyms (e.g.

WordNet thesaurus, the dictionnary of synonyms generated by L2R method [15]); or (ii) using standard similarity measures on strings or on sets of strings [4]. Furthermore, ontology and data knowledge (disjunctions and UNA) is exploited by N2R in a filtering step to reduce the number of reference pairs that are considered in the equation system. The functions modeling the influence between similarities are a combination of maximum and average functions in order to take into account the constraints of functionality and inverse functionality declared in the OWL ontology in an appropriate way.

N2R can also take as input a set of reference pairs that are reconciled (sim =1) by another method (e.g. L2R [15] in the LN2R approach) or given by a user like the *owl:same-as* links available in the Open Linked Data cloud.

The equations modeling the dependencies between similarities. For each pair of references, its similarity score is modeled by a variable x_i and the way it depends on other similarity scores, is modeled by an equation: $x_i = f_i(X)$, where $i \in [1..n]$ and n is the number of reference pairs for which we apply N2R, and $X = (x_1, x_2, \ldots, x_n)$ is the set of their corresponding variables. Each equation $x_i = f_i(X)$ is of the form:

$$f_i(X) = max(f_{i-df}(X), f_{i-ndf}(X))$$

The function $f_{i-df}(X)$ is the maximum of the similarity scores of the value pairs and the reference pairs of attributes and relations with which the i-th reference pair is functionally dependent. The maximum function allows propagating the similarity scores of the values and the references having a strong impact. The function $f_{i-ndf}(X)$ is defined by a weighted average of the similarity scores of the value pairs (and sets) and the reference pairs (and sets) of attributes and relations with which the i-th reference pair is not functionally dependent. See [16] for the detailed definition of $f_{i-df}(X)$ and $f_{i-ndf}(X)$.

Iterative algorithm for reference pairs similarity computation. Solving this equation system is done by an iterative method inspired from the Jacobi method [8], which is fast converging on linear equation systems. To compute the similarity scores, we have implemented an iterative resolution method. At each iteration, the method computes the variable values by using those computed in the precedent iteration. Starting from an initial vector $X^0 = (x_1^0, x_2^0, ..., x_n^0)$, the value of the vector X at the k-th iteration is obtained by the expression: $X^k = F(X^{k-1})$. At each iteration k we compute the value of each x_i^k: $x_i^k = f_i(x_1^{k-1}, x_2^{k-1}, ...x_n^{k-1})$ until a fix-point with a precision ϵ is reached. The fix-point is reached when: $\forall i, |x_i^k - x_i^{k-1}| <= \epsilon$.

In order to illustrate the iterative resolution of the equation system, we consider an extract of RDF data given in Figure 2 corresponding to the set of RDF facts where the references $S1\_a1$, $S1\_c1$, $S2\_a1$ and $S2\_c1$ are involved. By considering the disjunctions between concepts of $O1$ and the UNA in $S1$ and $S2$, we obtain an equation system of six variables:

x_1 = Sim$_r$(S1_a1, S2_a1) ; x_2 = Sim$_r$(S1_c1, S2_c1) ;
x_3 = Sim$_r$(S1_p1, S2_p1) ; x_4 = Sim$_r$(S1_p1, S2_p2) ;
x_5 = Sim$_r$(S1_p2, S2_p1) ; x_6 = Sim$_r$(S1_p2, S2_p2).

We give bellow, the similarity scores of basic values obtained by using the Jaccard similarity measure. For clarity reasons, we denote the value of an attribute A associated to a reference i as: $A.val(i)$. For example, the $confYear$ value associate to the

reference $S2\_c2$ is denoted $confYear.val(S2\_c2)$ which equals to "2008". The similarity score of the two conference names that are needed in the equation system and that belong to $]0,1[$ is:
$Sim_v(confName.val(S1\_c1), confName.val(S2\_c1)) = 0.43$. All the similarity scores of basic values, that are needed in the computation, are either equal to 1 or equal to 0.

The weights that are used in the weighted average of equations are computed in function of the number of common attributes and common relations of the reference pairs. The similarity computation is illustrated by the equation system (see Table 1) obtained from the data descriptions shown in Figure 2 which conforms to the ontology $O1$. The detailed equations expressing the similarity computation of two articles and two conferences are as follows:
$x_1 = \max(\frac{1}{2}(Sim_v(title.val(S1\_a1), title.val(S2\_a1)) + Sim_v(year.val(S1\_a1),$
$year.val(S2\_a1))), \frac{1}{6}(x_2 + SJ(\{S1\_p1, S1\_p2\}, \{S2\_p1, S2\_p2\}),$
$Sim_v(pageFrom.val(S1\_a1), pageTo.val(S2\_a1)))$
with SJ is the $SoftJaccard_o$ similarity measure between sets of objects (see section 4.2)

$x_2 = \max(x_1, \max(\frac{1}{2}(Sim_v(confName.val(S1\_c1), confName.val(S2\_c1)) +$
$Sim_v(confYear.val(S1\_c1), confYear.val(S2\_c1))), \frac{1}{4}(Sim_v(city.val(S1\_c1),$
$city(S2\_c1)))$

$x_3 = \frac{1}{2} * x_1 + \frac{1}{2} * Sim_v(name.val(S1\_p1), name.val(S2\_p1))$

The equation system and the different iterations of the resulting similarity computation are provided in Table 1. We assume that fix-point precision ϵ equals to 0.005.

Iterations	0	1	2	3
$x_1 = \max(\frac{1}{2}(1+1), \frac{1}{6}(x_2 + XS_1 + 1 + 1))$	0	1	1	1
$x_2 = \max(x_1, \max(\frac{1}{2}(0.43+1), \frac{1}{4}(1)$	0	0.71	1	1
$x_3 = \frac{1}{2}(x_1 + 1)$	0	0.5	1	1
$x_4 = \frac{1}{2}(x_1 + 1)$	0	0.5	1	1
$x_5 = \frac{1}{2}(x_1 + 0)$	0	0	0.5	0.5
$x_6 = \frac{1}{2}(x_1 + 0)$	0	0	0.5	0.5

Table 1. *Example of iterative similarity computation*

The solution of the equation system is $X = (1, 1, 1, 1, 0.5, 0.5)$. This corresponds to the similarity scores of the six reference pairs. The fix-point has been reached after three iterations. If we fix the reconciliation threshold T_{rec} at 0.80, then we obtain four reconciliation decisions: two articles, two conferences and two pairs of persons.

4 Link Discovery Method (LDM)

We present in this section our LDM approach which aims to discover a LOD source that shares concepts and data with a data source described dy a domain ontology. Our

approach compares a local dataset on which domain knowledge can be declared and a LOD dataset by using a combined ontology reconciliation and reference reconciliation method. Since data that is provided by the LOD source and by the domain application source is not described using the same ontology, we have adapted N2R method in order to be able to compute data similarities when data do not belongs to non disjoint concepts but to similar concepts. Furthermore, we have defined how similarities between concepts of two ontologies can be computed when some of their references are common (i.e. same URI or *owl:same-as* links that have been previously asserted) or similar. The main steps of our link discovering approach are as follows:

1. application of an ontology mapping tool to obtain: (i) the set of equivalent/ comparable properties and (i) initial similarity scores for some concept pairs;
2. building of the two equation systems: the *conceptual equation system* which expresses the similarity computation between pairs of concepts in function of their labels, their structural similarity and their references; and the *instance level equation system* one which expresses the similarity computation between pairs of references in function of their common description and the similarity of the concepts they are instance of;
3. iterative resolution of the conceptual equation system until a fix point is reached;
4. iterative resolution of the instance level equation system until a fix point is reached.

The two steps (3) and (4) are iterated until a global fix point is reached, i.e., neither the resolution of the conceptual equation system nor the resolution of the instance level equation system does update the similarity scores.

In the following subsections, we will first describe the elementary similarity measures that are used to compute similarities. Then, we present the two equation systems that have been defined to compute concept similarities and data similarities. Finally, we illustrate our LDM approach on data and ontologies of publication domain.

4.1 Initialization

We first use an alignment tool which exploits lexical and structural information to find similarity scores between ontology elements (concepts and properties). Given a local ontology O1 and a LOD ontology O2, the used alignment tool finds a set of mappings and each mapping is described by the tuple $\{e_1, e_2, co, rel\}$ where e_1 is aligned with the confidence co to the element e_2 using the type of correspondence rel (e.g. equivalence, subsumption, overlap, closeness, etc.). These scores are used to initialize the similarity score sim_{Init} of each pair of concepts and to find a set of properties (relations or attributes) that are very similar ($rel = equivalence$ or $subsumption, co \geq th$ and th is a high threshold). These properties are then considered as equivalent.

4.2 Elementary similarity measures

We present in this section the elementary measures used to compute similarity scores between pairs of concepts of two ontologies. These elementary similarity measures take into account the lexical and the structural knowledge declared in the two ontologies.

Most of these elementary similarity measures are based on the *SoftJaccard* similarity measure which computes similarity between sets of basic values or between sets of objects (e.g., references, concepts).

SoftJaccard: a similarity measure for sets of objects. In [16], we have defined the *SoftJaccard* similarity measure which is an adaptation of the *Jaccard* similarity measure in the sense that: (i) instead of considering only basic values we consider sets of basic values and (ii) instead of considering the equality between values we consider a similarity score with respect to a threshold θ.

Let S_1 and S_2 be two sets of elements which can be basic values or objects. To compute the similarity score between S_1 and S_2 we compute, first, the set $CLOSE_T(S_1, S_2, \theta_k)$ which represents the set of element pairs of $S_1 \times S_2$ having a similarity score $sim_T \geq \theta$.

$CLOSE_T(S_1, S_2, \theta) = \{e_j \mid e_j \in S1 \text{ and } \exists e_k \in S_2 \text{ s.t. } Sim_T(e_j, e_k) > \theta\}$,

with T a parameter which indicates if the sets S_1 and S_2 contain basic values, then $T = v$ or contain objects, then $T = o$. When $T = v$, the function Sim_v corresponds to a similarity measure between basic values like $Jaccard$, $Jaro - Winkler$, and so on [4]. When $T = o$, the function Sim_o corresponds to a similarity score that can be provided by a tool dedicated to object comparison like N2R tool [16] for references or TaxoMap [9] tool for concepts.

$$SoftJaccard_T(S_1, S_2, \theta) = \frac{\mid CLOSE(S_1, S_2, \theta) \mid}{\mid S_1 \mid}, \; with \; \mid S_1 \mid \geq \mid S_2 \mid$$

Similarity measures used to compare concepts. To compute the similarity scores between concepts we exploit both the conceptual content which means the sets of ancestors and the sets of descendants but also the sets of shared properties with respect to a given equivalence relation. The similarity score between concepts is also function of the similarity scores of their references, i.e. instance level content.

Similarity of concept labels. In OWL ontologies sets of labels are usually associated to the concepts. In case of concepts where the labels are not given, we consider their corresponding URIs. Let L_1 be the set of labels of a concept c_1 and L_2 be the set of labels of the concept c_2. The label similarity sim_{label} is computed by applying the $SoftJaccard$ similarity measure on the two sets of basic values L_1 and L_2: $sim_{label}(c_1, c_2) = SoftJaccard_v(L_1, L_2, \theta_1)$.

Similarity of concept ancestors. For two concepts, we also compute the similarity of their ancestor sets in the two ontologies. Let A_1 be the set of ancestors of the concept c_1 and A_2 be the set of ancestors of c_2. The ancestor similarity sim_{anc} is computed by applying $SoftJaccard$ similarity measure on the two sets of concept ancestors (i.e. objects) which is defined as follows: $sim_{anc}(c_1, c_2) = SoftJaccard_o(A_1, A_2, \theta_2)$

Similarity of concept descendants. The similarity score of two concepts also depends on the similarity scores of their descendants in the two ontologies. Let D_1 be the set of descendants of the concept c_1 and D_2 be the set of descendants of c_2. The descendant similarity sim_{desc} is computed by applying $SoftJaccard$ similarity measure on the two sets of concept descendants which is defined as follows:

$sim_{desc}(c_1, c_2) = SoftJaccard_o(D_1, D_2, \theta_3)$

Similarity of shared properties of concepts. The similarity of two concepts depends on the proportion of equivalent properties compared to the full number of properties defined for both concepts. Let $R1_d$ (resp. $R2_d$) be the set of properties such that the concept c_1 (resp. c_2) is subsumed by the (equivalent) property domain and let $R1_r$ (resp. $R2_r$) the set of properties such that the c_1 (resp. c_2) is subsumed by one of the range of the (equivalent) property. The relation similarity sim_{rel} is defined as follows :

$$sim_{rel}(c_1, c_2) = \frac{\mid (R1_d \cap R2_d) \cup (R1_r \cap R2_r) \mid}{\mid (R1_d \cup R2_d \cup R1_r \cup R2_r) \mid}$$

Similarity of concept references. The similarity score of two concepts also depends on the set of their references. Let I_1 (resp. I_2) be the set of instances of c_1 (resp. c_2), the similarity of c_1 and c_2 depends on the similarity scores obtained for the pairs of references of $I_1 \times I_2$ and it is computed by applying the $SoftJaccard$ similarity measure on the sets I_1 and I_2 of references (i.e. objects). $sim_{ref}(c_1, c_2)$ is defined as follows: $sim_{ref}(c_1, c_2) = SoftJaccard_o(I_1, I_2, \theta_4)$.

4.3 Equation modeling the dependencies between similarities in LDM approach

In LDM approach the similarity of each pair of references is expressed by a variable x_i in the instance level equation system. Its value depends on the common description of the pair of references w.r.t the equivalent/ comparable properties (cf. N2R). It depends also on the similarity scores of the concepts sc_i that are instantiated by the pair of references. An equation of the instance level equation system$x_i = g_i(X)$, where $i \in [1..n]$ and n is the number of reference pairs and $X = (x_1, \ldots, x_n)$, is of the form:

$$g_i(X) = \frac{1}{2}(sc_i, f_i(X))$$

with sc_i is the similarity score computed by the resolution of the conceptual equation system presented in the following. The function $f_i(X)$ is expressed as in N2R method and we consider that knowledge on the (inverse) functionality of the shared properties declared in the local ontology is also fulfilled in the LOD ontology.

The similarity of each pair of concepts (c, c') is expressed by a variable xc_j in the conceptual equation system. Its value depends on the initial similarity score provided by the alignment tool, the similarity of their labels, the set of their equivalent / comparable properties and the similarity of their references represented respectively by the constants sim_{j-init}, $sim_{j-label}$, sim_{j-rel} and sim_{j-ref}. It depends also on the similarity of their ancestors and their descendants represented by the variables XSC_{j-anc} and XSC_{j-desc} computed using SotfJaccard function.

An equation $xc_j = h_j(XC)$, where $j \in [1..m]$ and m is the number of concept pairs and $XC = (xc_1, \ldots, xc_m)$, is of the form:

$$h_j(XC) = max(sim_{j-init},$$
$$\frac{1}{5}(XSC_{j-anc} + XSC_{j-desc} + sim_{j-rel} + sim_{j-label} + sim_{j-ref}))$$

The values of the constants sim_{j-init}, $sim_{j-label}$, sim_{j-rel} and sim_{j-ref} are computed using the similarity functions described in the above subsection.

The size m of the conceptual equation system is $\mid C_1 \times C_2 \mid$, where C_1 (resp. C_2) is the set of concepts of the ontology O_1 (resp. O_2). The size of the instance level equation system depends on the number k of comparable relations and on the size of their corresponding domain instances and range instances. Let r_{i1} and r_{i2} be two comparable relations. Let E_{i1} (resp. E_{i2}) be the set of domain instances of r_{i1} (resp. of r_{i2}) and E_{i3} (resp. E_{i4}) be the set of range instances of r_{i3} (resp. r_{i4}). It also depends on the number of comparable attributes k' and on the size of their corresponding domain instances. Let a_{j1} and a_{j2} be two comparable attributes. Let E_{j1} (resp. E_{j2}) be the set of domain instances of a_{j1} (resp. of a_{j2}). The number n of variables of the instance level equation system is:

$$n = \mid \bigcup_{i=1}^{i=k} ((E_{i1} \times E_{i2}) \cup (E_{i3} \times E_{i4})) \cup (\bigcup_{j=1}^{j=k'} (E_{j1} \times E_{j2})) \mid$$

The computation complexity of the LDM method is O($(n^2 * it_{ref}) + (m^2 * it_c)$), with it_{ref} is the number of iterations of the instance level equation system and it_c is the number of iterations of the conceptual equation system.

One of the most distinguishing characteristic of LDM is its ability to propagate similarities at different levels: (i) between pairs of concepts, (ii) between pairs of references and (iii) between sets of references and sets of concepts. By using two separated equation systems we avoid the propagation between references when we compute the concept similarity scores and we avoid also the propagation between concepts when we compute the reference similarity scores. Thus, we decrease the size of the equation system and we allow a user to visualize and validate the intermediate equation system results.

4.4 Illustrative example

We present in Figure 3 an extract of the DBLP ontology which is used to describe the DBLP data published in the LOD. The considered data set only contains a collection of conference proceedings and the collection of their corresponding research papers in computer science. In order to illustrate our approach of link discovery, we will compare the local RDF data of the source S1 given in Figure 2 with the extract of DBLP dataset of the LOD given in Figure 4.

The initialization step provides the following initial similarity scores for the concept pairs:
$sim_{init}(Article, InProceedings) = 0.3$; $sim_{init}(Article, Proceedings) = 0.1$;
$sim_{init}(Article, Agent) = 0.1$; $sim_{init}(Person, InProceedings) = 0.0$;
$sim_{init}(Person, Proceedings) = 0.0$; $sim_{init}(Person, Agent) = 0.3$;
$sim_{init}(Conference, InProceedings) = 0.2$;
$sim_{init}(Conference, Proceedings) = 0.2$; $sim_{init}(Conference, Agent) = 0.1$

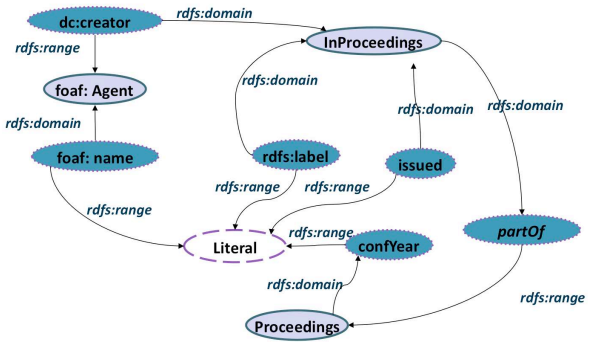


Fig. 3. An extract O2 of LOD DBLP ontology

LOD source S2:
InProceedings(S2_a1); label(S2_a1,"Implementing the TEA algorithm on sensors'); Agent(S2_p1); Agent(S2_p2); issued(S2_a1, "2004"); name(S2_p1,"Olga V. Gavrylyako"); name(S2_p2,"Shuang Liu");

Proceedings(S2_c1); label(S2_c1, "42nd Annual Southeast Regional Conference, 2004"); creator(S2_a1,S2_p1); creator(S2_a1,S2_p2); partOf(S2_a1,S2_c1);
InProceedings(S2_a2); label(S2_a2,"New Chaos Produced from Synchronization of Chaotic Neural Networks"); Agent(S2_p3); issued(S2_a2, "2008");
name(S2_p3,"Zunshui Cheng");

Proceedings(S2_c2); label(S2_c2, "Advances in Neural Networks - ISNN 2008, 5th International Symposium on Neural Networks"); creator(S2_a2,S2_p3); partOf(S2_a2,S2_c2);

Fig. 4. An extract of DBLP data set on the LOD

Since, there are no subsumption relations in $O1$ and in $O2$ the conceptual equations do not take into account the similarity scores of the ancestors and of the descendants. For example, the equation expressing the similarity of the two concepts $Article$ and $InProceedings$ is: $xc_1 = \max(0.3, \frac{1}{3}(\frac{2}{3} + 0 + sim_{1-ref}))$. In this example, the conceptual equation system consists of nine variables $(xc_1, \ldots, xc_9)$.

The instance level equation system consists of twenty-five equations representing all the reference pairs where the common description is not empty. For example, the equations expressing:

- The similarity of the two references $S1\_a1$ (Article) and $S2\_a1$ (InProceedings) is: $x_1 = \frac{1}{2}(sc_1, \max(\frac{1}{2}(Sim_v(label.val(S2\_a1), title.val(S1\_a1)) + Sim_v(issued.val(S2\_a1), year.val(S1\_a1)), \frac{1}{4}(SJ(\{S1\_p1, S1\_p2\}, \{S2\_p1, S2\_p2\}) + x_{14})$
- The similarity of the two references $S1\_c1$ (Conference) and $S2\_c1$(Proceedings) is : $x_{14} = \frac{1}{2}(sc_{14}, \max(x_1, \frac{1}{2}(Sim_v(label.val(S2\_c1), confName.val(S1\_c1)))$
- The similarity of the two references $S1\_p1$ (Person) and $S2\_p1$ (Agent) is:

  $x_5 = \frac{1}{2}((sc_5, \frac{1}{2}(x_1 + Sim_v(name.val(S1\_p1), name.val(S2\_p1))$

– The similarity of the two references $S1\_p1$ (Conference) and $S2\_p1$ (InProceedings) is: $x_{18} = \frac{1}{2}(sc_{18}, \max(Sim_v(confName.val(S1\_c1), label.val(S2\_a1))$

In Table 2 we show the iterative resolution of the conceptual equation system ES_1 modeling the similarity of all the pairs of concepts of the ontologies $O1$ and $O2$. The column sim_{init} represents the initial similarity score computed by an external concept alignment tool, like TaxoMap [9]. The Table 3 shows the results of the iterative resolution of the instance level equation system ES_2 of the pairs of references coming from the local source S1 of Figure 2 which conforms to the local ontology O1 and the S2 LOD source which conforms to the LOD DBLP ontology O2.

Variables of ES_1	**sim$_{init}$**	**Resolution1– iteration1**	**Resolution2– it1**
xc_1 = (Article, InProceedings)	0.3	$\max(0.3, \frac{1}{3}(\frac{4}{6})) = 0.3$	xc_1= 0.33
xc_2 = (Article, Proceedings)	0.1	$\max(0.1, \frac{1}{3}(\frac{1}{6})) = 0.1$	xc_2= 0.1
xc_3 = (Article, Agent)	0.1	$\max(0.1, \frac{1}{3}(0)) = 0.1$	xc_3= 0.1
xc_4 = (Person, InProceedings)	0.0	$\max(0, \frac{1}{3}(0)) = 0.0$	xc_4= 0.0
xc_5 = (Person, Proceedings)	0.0	$\max(0, \frac{1}{3}(0)) = 0.0$	xc_5= 0.0
xc_6 = (Person, Agent)	0.3	$\max(0.3, \frac{1}{3}(1)) = 0.33$	xc_6= 0.427
xc_7 = (Conference, InProceedings)	0.2	$\max(0.2, \frac{1}{3}(\frac{1}{4})) = 0.2$	xc_7= 0.2
xc_8 = (Conference, Proceedings)	0.2	$\max(0.2, \frac{1}{3}(\frac{2}{4})) = 0.25$	xc_8= 0.33
xc_9 = (Conference, Agent)	0.1	$\max(0.1, \frac{1}{3}(0)) = 0.1$	xc_9= 0.1

Table 2. *The two resolutions of the conceptual equation system ES_1*

The $Resolution1$ step of ES_1 corresponds to the first iterative resolution of ES_1 where sim_{ref} of all the concepts equals to 0. The fix-point of $\epsilon = 0.05$ is reached in two iterations. The $Resolution1$ step of ES_2 corresponds to the first iterative resolution of ES_2 where sc_i of all the references equals to sim_{init} (c.f. Table 2) . The fix-point of $\epsilon = 0.05$ is reached in three iterations. The $Resolution2$ step of ES_1 corresponds to the second iterative resolution of ES_1 where sim_{ref} of all the concepts equals to the similarity scores computed by ES_2 at the last iteration of $Resolution1$. The fix-point of $\epsilon = 0.05$ is also reached in two iterations. The $Resolution2$ step of ES_2 corresponds to the second iterative resolution of ES_2 where sc_i of all the references equals to the similarity scores computed by ES_1 at the last iteration of $Resolution1$. The fix point of $\epsilon = 0.05$ is reached in two iterations.

The global fix-point is reached after three resolutions. At $Resolution3$ [2] of the two systems ES_1 and ES_2 we obtain the same similarity scores than the last iteration of their corresponding $Resolution2$ step. The results obtained by ES_1 show that the method obtains the best similarity scores for the most possible equivalent concepts: $(Article, InProceedings)$, $(Person, Agent)$ and $(Conference, Proceedings)$. In an analogous way, the results obtained by ES_2 show that the best similarity scores are obtained for the most possible *owl:same-as* references. If we fix the reconciliation

[2] The scores are not shown here, they are equal to those obtained in the $Resolution2$ of ES_1 and ES_2.

Variables of ES_2	Resolution1– iteration 1	Res1–it2	Res1–it3	Res2–it1
x_1 = ($S1\_a1$,$S2\_a1$)	$\frac{1}{2}(0.3 + \max(\frac{1}{2}(2), \frac{1}{4}(0)) = 0.66$	0.66	0.66	0.665
x_2 = ($S1\_a2$,$S2\_a1$)	$\frac{1}{2}(0.3 + \max(\frac{1}{2}(0), \frac{1}{4}(0)) = 0.15$	0.165	0.175	0.19
...	...	...	...	...
x_5 =($S1\_p1$,$S2\_p1$)	$\frac{1}{2}(0.33 + \frac{1}{2}(1)) = 0.415$	0.58	0.58	0.62
x_6 = ($S1\_p2$,$S2\_p1$)	$\frac{1}{2}(0.33 + \frac{1}{2}(0)) = 0.165$	0.33	0.33	0.379
...	...	...	...	...
x_{14} = ($S1\_c1$,$S2\_c1$)	$\frac{1}{2}(0.25 + \max(0, \frac{1}{2}(0.438)) = 0.219$	0.455	0.455	0.477
...	...	...	...	...
x_{18} = ($S1\_c1$,$S2\_a1$)	$\frac{1}{2}(0.2 + \max(0)) = 0.1$	0.1	0.1	0.1
...	...	...	...	...

Table 3. *The resolution of the instance level equation system ES_2*

threshold at 0.45 we infer the reconciliation of the two papers ($S1\_a1$, $S2\_a1$), of the two persons ($S1\_p1$, $S2\_p1$) and of the two conferences ($S1\_c1$, $S2\_c1$).

In this example we have shown the applicability of the approach even when the considered ontologies are not syntactically close and when they have very poor structure (no subsumption relations) which means that the ancestors and the descendants are not considered.

5 Conclusion and Future Work

In this paper we have presented a Link Discovering Method (LDM) which allows discovery of new data sources that are published in the Open Linked Data cloud (LOD). Our approach is based on the idea of comparing a local dataset on which domain knowledge can be declared and a LOD dataset by using a combined ontology reconciliation and reference reconciliation method. By using our LDM method one may discover more *owl:same-as* links with datasets available on the LOD.

One of the most distinguishing characteristic of our link discovery approach resides on its ability to propagate similarities at different levels: (i) between pairs of concepts, (ii) between pairs of references and (iii) between sets of references and sets of concepts. By using two separated equation systems we avoid the propagation between references when we compute the concept similarity scores and we avoid also the propagation between concepts when we compute the reference similarity scores.

As a very short term perspective, we plan to test our LDM approach on real data sets and evaluate the quality of its results and its scalability. It will be worth to compare LDM method with those of existing link discovery methods like [10]. As future work, we plan to extend the approach to be able, in addition of the equivalent properties, take into account the other properties in oder to consider richer data descriptions. Moreover, we aim also to extend the LDM method to compute also similarities between the properties of the considered ontologies.

References

1. Berners-Lee, T., Cailliau, R., Groff, J.F., Pollermann, B.: World-wide web: The information universe. Electronic Networking: Research, Applications and Policy 1(2), 74–82 (1992)
2. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: KDD. pp. 39–48 (2003)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
4. Cohen, W.W., Ravikumar, P.D., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: IIWeb. pp. 73–78 (2003)
5. Cohn, D.A., Atlas, L.E., Ladner, R.E.: Improving generalization with active learning. Machine Learning 15(2), 201–221 (1994)
6. Dong, X., Halevy, A.Y., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD Conference. pp. 85–96 (2005)
7. Euzenat, J., Loup, D., Touzani, M., Valtchev, P.: Ontology alignment with ola. In: Sure, Y., Corcho, O., Euzenat, J., Hughes, T. (eds.) Proc. 3rd ISWC2004 workshop on Evaluation of Ontology-based tools (EON), Hiroshima (JP). pp. 59–68 (2004)
8. Golub, G.H., Loan, C.F.V.: Matrix computations (3rd ed.). Johns Hopkins University Press, Baltimore, MD, USA (1996), `http://portal.acm.org/citation.cfm?id=248979`
9. Hamdi, F., Safar, B., Niraula, N.B., Reynaud, C.: Taxomap in the oaei 2009 alignment contest. In: Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009) Chantilly, USA, October 25, 2009 (2009)
10. Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R.J., Wang, M.: A framework for semantic link discovery over relational data. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009. pp. 1027–1036 (2009)
11. Kiefer, C., Bernstein, A.: The creation and evaluation of isparql strategies for matchmaking. In: The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings. pp. 463–477 (2008)
12. Li, J., Tang, J., Li, Y., Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. IEEE Transactions on Knowledge and Data Engineering 21, 1218–1232 (2009)
13. Nikolov, A., Uren, V.S., Motta, E., Roeck, A.N.D.: Handling instance coreferencing in the knofuss architecture. In: Proceedings of the 1st IRSW2008 International Workshop on Identity and Reference on the Semantic Web, Tenerife, Spain, June 2, 2008 (2008)
14. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal 10(4), 334–350 (2001)
15. Saïs, F., Pernelle, N., Rousset, M.C.: L2r: A logical method for reference reconciliation. In: AAAI. pp. 329–334 (2007)
16. Saïs, F., Pernelle, N., Rousset, M.C.: Combining a logical and a numerical method for data reconciliation. J. Data Semantics 12, 66–94 (2009)
17. Scharffe, F., Liu, Y., Zhou, C.: Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In: Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR), Pasadena (CA US) (2009)
18. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches pp. 146–171 (2005)
19. Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. Inf. Syst. 26(8), 607–633 (2001)

Repairing Provenance Policy Violations by Inventing Non-Functional Nodes

Saumen Dey[1], Daniel Zinn[2], and Bertram Ludäscher[1 2]

[1] Dept. of Computer Science, University of California, Davis
[2] Genome Center, University of California, Davis

Abstract. In scientific collaborations, provenance is increasingly used to explain, debug, reproduce, and determine the validity and quality of data products. In such environments, it can be infeasible or undesirable to publish the complete provenance of all the final output data products. We have developed ProPub, a system that allows users to publish a customized version of their data provenance, based on a set of publication and customization requests, while observing certain provenance publication policies, expressed as logic integrity constraints. The user's customization requests may violate one or more integrity constraints. In previous work, we removed additional parts of the provenance graph (i.e., not directly requested by the user) to repair policy violations. In this paper, we present an alternative approach which ensures that all relevant nodes are retained in the provenance graph. The key idea is to introduce new (non-functional) nodes that are used to represent lineage dependencies, without revealing information that the user wants to protect. With this new approach, a user may now explore different provenance publication strategies, and choose the most appropriate one before publishing sensitive provenance data.

1 Introduction

In the emerging paradigm of collaborative, data-intensive science, sharing data products even prior to publication is desirable [1,2]. Yet, without a proper scientific publication associated with openly published data, its validity and accuracy might be questionable. This is problematic in an open environment, where published data by one scientist is used by another scientist as input for further data analyses. In such an environment, data provenance (the lineage and processing history of data) can help to ensure data quality [3,4,5,6,7]. It is thus desirable to publish data products together with their provenance.

In many cases, however, provenance data can be sensitive and may contain private information or intellectual property that should not be revealed [7,8,5]. Consequently, a balancing act (Figure 1) is necessary between (i) the desire to publish provenance data so that collaborators can understand and rely on the shared data products, and (ii) the need to protect sensitive information, e.g., due to privacy concerns or intellectual property issues.

We view provenance as a bipartite, directed, acyclic graph, capturing which data nodes were consumed and produced, respectively, by invocation (i.e., computation) nodes. Our model thus corresponds to the Open Provenance Model (OPM) which captures the dependencies between data artifacts and invocations [9,10].

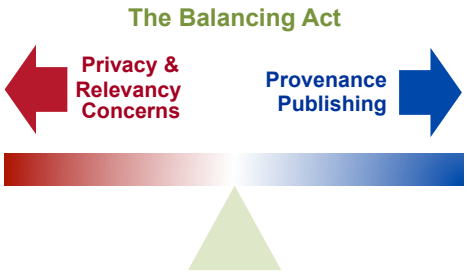


Fig. 1. In collaborative settings, scientists publish provenance for an improved understanding of the result data. With increasing privacy concerns, collaborators have to choose the right balance between providing sufficient provenance data and protecting sensitive information.

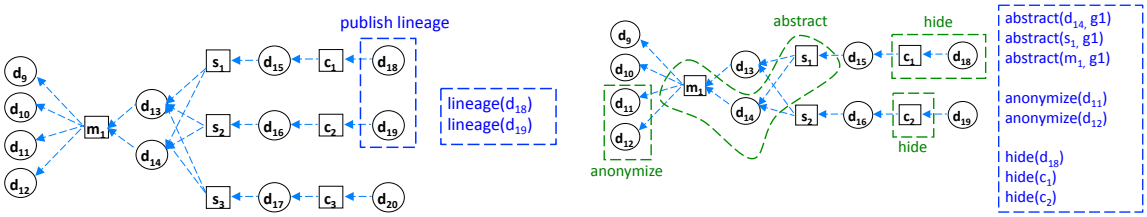
To sanitize provenance graphs, a scientist can remove sensitive data nodes or invocations nodes from the provenance graph. Alternatively, she can *abstract* a set of sensitive nodes by grouping them into a single, abstract node. This update may violate some of the integrity constraints of the provenance graph [11]. For example, grouping multiple nodes into one abstraction node may introduce new dependencies which were absent in the initial provenance graph. Hiding nodes may also make some nodes in the final graph appear independent of each other even though they are dependent in the initial graph. Thus, one can no longer trust that the published provenance data is "correct" (e.g., there are no false dependencies) or "complete" (e.g., there are no false independencies). Therefore, we propose a system that allows a publisher to provide a high-level specification what parts of the provenance graph are to be published and which parts are to be sanitized, *while guaranteeing* that at the same time certain provenance publication constraints are observed.

2 Motivating Example

Figure 2(a) shows the provenance graph (PG) taken from the First Provenance Challenge [12]. *Data nodes* are depicted as circles and *invocation nodes* (representing computations) as boxes; dependencies among them are shown as directed edges. These edges capture the lineage of data and thus are typically drawn from right (newer nodes) to left (older nodes). For example, $\mathtt{d}_{16}$ *was generated by* an invocation $\mathtt{s}_2$, and was in turn *used* by invocation $\mathtt{c}_2$, denoted by, respectively $\mathtt{s}_2 \overset{gen\_by}{\longleftarrow} \mathtt{d}_{16}$ and $\mathtt{d}_{16} \overset{used}{\longleftarrow} \mathtt{c}_2$.

Let us assume, the user wants to publish data products $\mathtt{d}_{18}$ and $\mathtt{d}_{19}$ along with their lineage data. Then, she will issue the publication requests as shown in Figure 2(a). A recursive query is used to retrieve all data and invocation nodes upstream from $\mathtt{d}_{18}$ and $\mathtt{d}_{19}$ and we get a modified provenance graph (PG') as shown in Figure 2(b). Note that the lineage of $\mathtt{d}_{20}$ up to $\mathtt{s}_3$ is not relevant for $\mathtt{d}_{18}$ and $\mathtt{d}_{19}$ and hence not included in PG'. Further assume that before publishing PG', the user also requests a set of customizations as shown in Figure 2(b).

Figure 3 shows the provenance graph we get after applying all the customization requests. We see that this provenance graph violates three provenance policies: There



(a) Provenance graph (PG) and publish request (b) User requests: `anonymize`, `abstract`, `hide`

Fig. 2. (a) User requests to publish the provenance of $\{d_{18}, d_{19}\}$; and (b) customization requests to *anonymize* data nodes $\{d_{11}, d_{12}\}$, to *abstract* nodes $\{m_1, d_{14}, s_1\}$, and to *hide* $\{c_1, d_{18}, c_2\}$

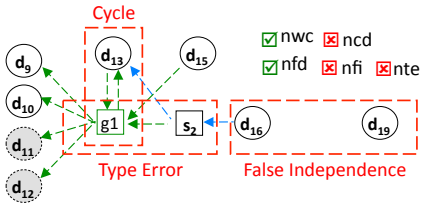


Fig. 3. Provenance graph after applying all user requests. Provenance policies No-Type Error (NTE), No-Cyclic Dependency (NCD) and No-False Independence (NFI) are violated, while No-Write Conflict (NWC) and No-False Dependence (NFD) are satisfied.

is a cycle between d_{13} and g_1, a type error for the edge from s_2 to g_1 (the graph should be bipartite), and there is no dependency between d_{19} and d_{16}, violating, respectively, the No-Cyclic Dependency (NCD), No-Type Error (NTE) and No-False Independence (NFI) policies. On the other hand, the provenance policies No-Write Conflict (NWC) and No-False Dependence (NFD) are not violated by these customization requests.

Outline and Contributions. In Section 3, we first describe the provenance model, user requests, provenance policies, and logical architecture of ProPub. This overall framework was proposed recently in [11]. In Section 4 we present our main contribution, i.e., a new way to repair policy violations, not by removing additional nodes (as in our prior work), but by introducing new (non-functional) nodes that represent the original lineage dependencies, without revealing information that the user wants to protect. We describe in detail how policy violations will be repaired such that all relevant nodes are retained in the final provenance graph. Related work is discussed in Section 5 and Section 6 presents some concluding remarks and suggestions for future work.

3 Provenance Publisher (ProPub)

In our recent work, we developed the system ProPub [11], which uses a declarative approach to publish customized policy-aware provenance. ProPub accepts the initial provenance graph and two types of input specifications. (i) User Requests: the publication and customization requests, and (ii) Provenance Policies: the integrity constraints

Relation Name	Description
`used(I, D)`	An edge specifying that the invocation I used the data artifact D.
`gen_by(D, I)`	An edge to indicate that the data artifact D was generated by invocation I.
`actor(I, A)`	An invocation node I, which was executed by actor A.
`data(D, R)`	A data artifact node, whose value can be retrieved using the reference R.
`dep(X, Y)`	An auxiliary relation and defined as `dep` $=$ `used` $\cup$ `gen_by` and to specify that node X depends on node Y, irrespective of the node types.

Table 1. Provenance Model for PROPUB

User Request	Description
`ur:lineage(D)`	Selects the complete lineage for the data artifact D
`ur:anonymize(N)`	Erases the actor/process identify or the data reference from the node N
`ur:hide(N)`	Removes the invocation or data node N
`ur:abstract(N, G)`	Collapses all nodes N to the abstract group G
`ur:retain(N)`	Keeps the node N in the customized provenance

Table 2. User requests for lineage publication and customization

to be observed. PROPUB then applies all user requests on the initial provenance graph and checks for policy violations. In case there is a violation, it applies repairs and generates the customized provenance graph.

Provenance Model. The provenance model used in PROPUB is based on OPM, the Open Provenance Model [13] and our earlier work [14]: A *provenance* (or *lineage*) *graph* is an acyclic graph $PG = (V, E)$, where the nodes $V = \mathrm{D} \cup \mathrm{I}$ represent either *data* items D or actor *invocations* I. The graph G is bipartite, i.e., the edges $E = E_{\texttt{use}} \cup E_{\texttt{gby}}$ are either *used* edges $E_{\texttt{use}} \subseteq \mathrm{I} \times \mathrm{D}$ or *generated-by* edges $E_{\texttt{gby}} \subseteq \mathrm{D} \times \mathrm{I}$. Here, a *used* edge $(\texttt{i}, \texttt{d}) \in E$ means that invocation `i` has read `d` as part of its *input*, while a *generated-by* edge $(\texttt{d}, \texttt{i}) \in E$ means that `d` was *output* data, written by invocation `i`. We use the schema shown in Table 1.

User Requests. The user requests supported by the PROPUB framework are summarized in Table 2. The PROPUB system expects user requests to be asserted as relational facts that can then be used by a Datalog rule engine. An user request can be a publication request or a customization request. A customization user request can request to remove a node or an edge or to keep that in the final graph.

Provenance Policies. The provenance graph supported by PROPUB is a bipartite directed acyclic graph. Also, an invocation can read many data artifacts, but a data artifact is written by exactly one invocation. We developed three provenance policies to verify if these structural properties are satisfied in the provenance graph $\mathrm{PG}'^{\Delta u}$, which we get after applying all the customization requests on PG'. PROPUB has two more provenance policies to ensure the correctness and completeness of information. These provenance policies are briefly defined in Table 3.

Provenance Policy	Description
No-Write Conflict (NWC)	A data artifact can be written by only one invocation.
No-Cyclic Dependency (NCD)	There is no cycle between any two nodes X and Y.
No-Type Error (NTE)	Two nodes with a direct dependency are of different types.
No-False Dependence (NFD)	Two nodes are dependent in $\mathrm{PG}'^{\Delta u}$ only if they are dependent in PG'.
No-False Independence (NFI)	Two nodes are independent in $\mathrm{PG}'^{\Delta u}$ only if they are independent in PG'.

Table 3. Provenance Policies

Constraint	Description
ic:wc(X, Y)	Write conflict: invocations X and Y are creating the same data node.
ic:cd(X, Y)	Cyclic dependency between nodes X and Y.
ic:te(X, Y)	Type error: nodes X and Y are connected via *used* or gen_by edges, but don't have the corresponding node types.
ic:fd(X, Y)	False dependency: node Y depends on X in $\mathrm{PG}'^{\Delta u}$, but not in PG'.
ic:fi(X, Y)	False independence: node Y depends on X in PG', but not in $\mathrm{PG}'^{\Delta u}$.

Table 4. Integrity constraint relations used to detect policy violations

We use a set of integrity constraints (ICs) to check whether the provenance policies defined in Table 3 are satisfied. Table 4 lists the "witness relations" that are defined by rules (not shown) and which are used to detect particular IC violations.[3]

3.1 Logical Architecture

The logical architecture of the PROPUB system is shown in Figure 4. The user submits a set of publication and customization requests U_0. The module Direct-Conflict-Detection detects *direct conflicts* among the given user-requests. For example, a hide and a retain request on the same node is an obvious conflict. The user needs to update her original requests until all *direct conflicts* are resolved, resulting in a conflict-free user request U. The Lineage-Selection module computes the sub graph PG', which contains all to-be-published data items (specified using the 'lineage' predicate) together with their complete provenance.

The Request-Policy-Evaluation module calculates the updates (Δu: inform of *insert* and *delete*) needed to apply all the user requests from U on PG'. It applies Δu on PG' and get a customized provenance graph $\mathrm{PG}'^{\Delta u}$. Then it checks if all the selected provenance policies (PP) are observed by evaluating respective integrity constraints. In case some of the policies are violated, this module calculates updates (Δp: inform of *insert* and *delete*) needed to repair the violations. In a final conflict resolution step using the module Implied-Conflict-Detection-Resolution, the system detects all such implied

[3] For example., we can detect whether a data node is created by different invocations X and Y and record this as ic:wc(X, Y).

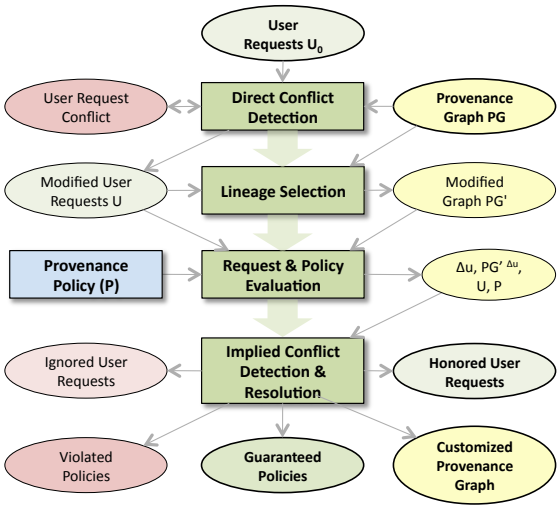


Fig. 4. PROPUB Architecture

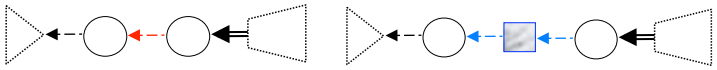
conflicts by comparing Δu and Δp. In case an *implicit conflict* is detected, it selects another subset from the given U and PP following the user preferences. These steps are repeated until there is no more policy violations. It then applies Δp on $\mathrm{PG}'^{\Delta u}$ to get the customized provenance graph (CG) ready to be published.

4 Repairing Policy Violations

If we apply the customization user requests on PG', we get an intermediate provenance graph $\mathrm{PG}'^{\Delta u}$ as we shown in Figure 2(a), 2(b) and 3. But, $\mathrm{PG}'^{\Delta u}$ may violate one or more provenance policies. In case the $\mathrm{PG}'^{\Delta u}$ violates a structural policy (NWC, NCD, and NTE), it will no more be a proper provenance graph. Also, in case it violates a non-structural policy (NFI and NFD), $\mathrm{PG}'^{\Delta u}$ may contain incorrect information or may become incomplete. Thus, user will not be able to publish $\mathrm{PG}'^{\Delta u}$. To resolve this issue, we apply the customization requests U on PG' in a strategic way such that it confirms to all the provenance policies.

Our strategy is primarily based on two ideas (i) inventing non-functional nodes, and (ii) converting user requests using other forms of user requests.

Inventing Non-functional Nodes. In case $\mathrm{PG}'^{\Delta u}$ has a structural violation, PROPUB resolves the violation by adding a new non-functional node. A non-functional node is added to maintain the structure of provenance graph. Presence of a non-functional node in the final customized graph may represent one data or invocation node or a set of data and invocation nodes. No mapping is maintained between the non-functional node and the nodes it replaced. Also, it will not carry any URL. Thus, no one will be able to reach to the value of a data artifact or the source code of an actor from a non-functional



(a) Policy Violation (b) Fix

Fig. 5. (a) direct dependency between data nodes causing a type error (NTE violation); (b) PROPUB resolves this by inventing a non-functional invocation node.

node. PROPUB invents minimum numbers of non-functional nodes to resolve a policy violation.

PROPUB uses the same strategy to resolve NFD policy violations. The fix to the violation of this policy is complex and may need more than one non-functional node to be added. In spite of this complexity, PROPUB resolves the violation using the minimum numbers of non-functional nodes.

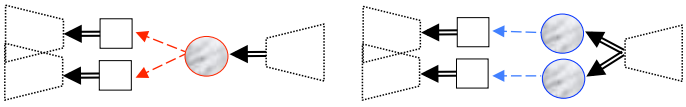
Converting User Requests. A publisher can use *ur:hide* requests to hide individual nodes or the partial structure of the provenance graph. When we apply these user requests all the selected nodes and the associated edges are removed from the provenance graph PG' and a set of independence may be created which violates the NFI policy. We can use the inventing *new non-functional node* strategy as discussed above and replace the selected node by a non-functional node to resolve this policy violation. But, this approach keeps the structure of the original provenance graph in the final provenance graph. Instead, PROPUB converts these *ur:hide* user requests into an equivalent set of *ur:abstract* user requests so that all the selected nodes are removed and no unintended dependencies are removed.

4.1 Repairing Structural Policy Violations

No-Type Error. This policy is violated in case there is a direct dependency between two nodes of same type (i.e. a dependency between two data artifacts or a dependency between two invocations). PROPUB invents a non-functional invocation node in case the policy violation is between two data artifacts as shown in Fig. 5. In the similar way, PROPUB invents a non-functional data node in case the policy violation is between two invocation nodes. We used the rules as shown below to create the non-functional nodes and fix the violations of this policy.

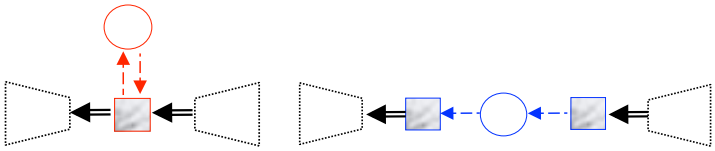
```
del_dep(X,Y) :- ic:te(X,Y).
add_data(f(X,Y),T) :- ic:te(X,Y), d_actor(X,_), T='ic:te'.
add_actor(f(X,Y),T) :- ic:te(X,Y), d_data(X,_), T='ic:te'.
add_dep(X,f(X,Y)) :- ic:te(N1,N2), X is N1.
add_dep(f(X,Y),Y) :- ic:te(N1,N2), Y is N2.
```

No-Write Conflict. This policy is violated in case there are $N(N \geq 2)$ *gen_by* edges for a data node. To resolve this violation PROPUB removes incorrect *gen_by* edges for the violated data node and keeps only one *gen_by* edge, which is there in PG. But, this



(a) Policy Violation (b) Fix

Fig. 6. In (a) there are two *gen_by* edges with the data node causing the No-Write conflict policy violation and PROPUB resolves this by inventing a non-functional data node as shown in (b).



(a) Policy Violation (b) Fix

Fig. 7. In (a) there is a cycle between the data node and the invocation node and PROPUB resolves this by inventing the non-functional invocation node as shown in (b).

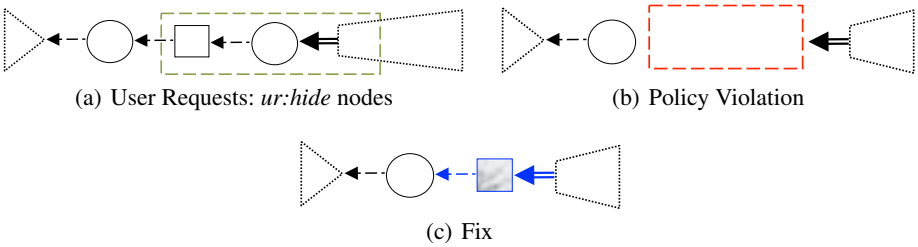
may violate the NFI policy as it removes dependencies for $N-1$ invocation nodes. To get around this side effect PROPUB invents $N-1$ non-functional data nodes and creates $N-1$ *gen_by* edges as shown in Fig. 6. Lastly, it copies all the *used* edges for the violated data node over to all $N-1$ non-functional data nodes. Following rules are used to create the non-functional nodes and fix the violations of this policy:

```
del_data(D) :- ic:wc(D).
del_dep(D,I) :- ic:wc(D), d_gen_by(D,I).
add_data(f(D,I),T) :- ic:wc(D), d_gen_by(D,I), T='ic:wc'.
add_dep(f(D,I),I) :- ic:wc(D), d_gen_by(D,I).
add_dep(I,f(D,I1)) :- ic:wc(D), d_gen_by(D,I1), d_used(I,D).
```

No-Cyclic Dependency. This policy is violated in case a node is reachable from itself. In Fig. 7, there is a cycle between a invocation node and a data node. To fix this violation PROPUB invents a non-functional invocation node and creates a *used* edge between the data node and the non-functional invocation node. Then it removes all the *gen_by* edges from the invocation node (except the one with the data node with which it has the cycle) and copies them over to the non-functional invocation node. In the similar way, PROPUB resolves this violation between two invocation nodes.

4.2 Repairing No-False Independence (NFI) Policy Violations

This policy is violated in case two nodes are not dependent in $\mathrm{PG}'^{\Delta u}$ even though they are in PG'. This may occur in case the *ur:hide* user requests are applied on PG' as shown in Fig. 8. One way to resolve this violation is to insert direct dependencies, which



(a) User Requests: *ur:hide* nodes (b) Policy Violation

(c) Fix

Fig. 8. PG' and user requests *ur:hide* are shown in (a). In (b) some dependencies are removed between nodes in $\mathrm{PG}'^{\Delta u}$. ProPub then resolves this in two steps (i) transforms these *ur:hide* requests into equivalent *ur:abstract* requests and (ii) applies these *ur:abstract* requests on PG' and gets the customized graph is shown in (c).

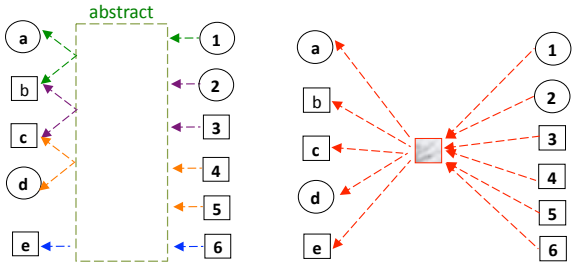
are there in PG' but missing in $\mathrm{PG}'^{\Delta u}$, between any two nodes in $\mathrm{PG}'^{\Delta u}$. But, this process may add too many edges and the graph may become unreadable. One optimization to this process is to develop transitive dependencies to reduce the total number of new edges needed. This may be computation intensive. ProPub uses a different strategy to fix this violation. Following rules are used to transform the *ur:hide* requests into an equivalent set of *ur:abstract* requests:

```
hide_connected(X,Y) :- ur:hide(X), ur:hide(Y), dep(X,Y).
hide_connected(X,X) :- ur:hide(X).
hide_connected(X,Y) :- hide_connected(Y,X).
hide_connected(X,Y) :- hide_connected(X,Z), hide_connected(Z,Y).
smaller(X) :- hide_connected(X,Y), X < Y.
minimum(X) :- ur:hide(X), not(smaller(X)).
abstract_hide(X,G) :- hide_connected(X,G), minimum(G).
```

The customization user requests *ur:abstract* removes nodes from PG', but does not violate the NFI policy. To avoid the NFI polic violations ProPub transforms the *ur:hide* user requests into an equivalent set of *ur:abstract* user requests. These will be applied to PG' in the same way the User issued *ur:abstract* requests are applied.

4.3 Repairing No-False Dependence (NFD) Policy Violations

This policy is violated in case two nodes are dependent in $\mathrm{PG}'^{\Delta u}$ even though they are not in PG'. This may occur in case the *ur:abstract* user requests are applied on PG' as shown in Fig. 9. In Fig. 9(a) we have a partial provenance graph showing the *ur:abstract* requests and the nodes with direct dependencies with one or more nodes selected to be abstracted. This figure shows that in PG' the data artifact '1' depends on data artifact 'a' and 'b'. In the similar way, the data node '2' depends on invocation nodes 'b' and 'c' and so on. Now, if we apply these *ur:abstract* requests by collapsing all the selected nodes into a abstracted node then in $\mathrm{PG}'^{\Delta u}$ the data artifact '1' become depended on nodes 'a', 'b', 'c', 'd', and 'e' and thus making $\mathrm{PG}'^{\Delta u}$ incorrect, as shown in Fig. 9.



(a) The *ur:abstract* user requests

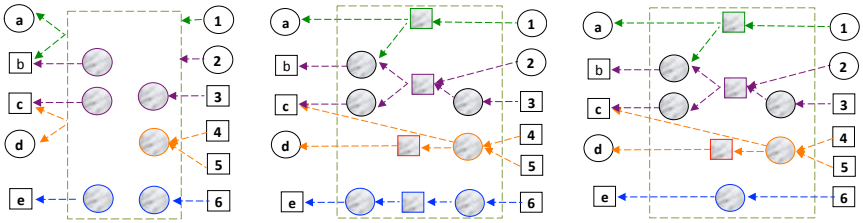
(b) The dependencies created by applying these *ur:abstract* user requests

Fig. 9. In (a) we show the boundary of one *ur:abstract* user requests set and the nodes with a direct dependency with one or more nodes selected to be abstracted. After these *ur:abstract* user requests are applied on PG' we get a new set of dependencies as shown in (b).

To avoid this policy violations ProPub takes a systematic three stages approach to apply the *ur:abstract* user requests. Instead of collapsing into one abstracted node, it invents a number of non-functional data and invocation nodes to maintain the dependencies between any two nodes in $\mathrm{PG}'^{\Delta u}$ as they are in PG'. This systematic approach ensures that the minimum number of non-functional nodes are invented. In the first stage, ProPub develops two sets in and out. The in is a set of data nodes which is used by some of the invocation nodes selected to be abstracted and invocation nodes which generated some of the data nodes selected to be abstracted. The out is a set of data nodes which is generated by one of the invocation node selected to abstracted and invocation nodes which used some of the data nodes selected to abstracted. It also calculates the dependencies for each of the node in set out on the nodes of the set in.

Now, ProPub creates non-functional data nodes for each of the invocation nodes from the sets in and out. One non-functional data node is created for exactly one invocation node from the set in through a *gen_by* edge. One non-functional data node is created for more then one invocation node from the set out through *used* edges in case these invocations have the same dependencies on the set in. At this stage, a non-functional data node is connected either to a node from the set in or to one or many nodes from the set out. For example, invocation node '4' '5' and depends on invocation nodes 'b' and 'c' and ProPub will create only one non-functional data node and two *used* edges. This is shown in Fig. 10(a).

In the second stage, it calculates the list of dependencies of all nodes from the set out to the nodes from in. ProPub creates one non-functional invocation node for each of these unique dependency lists and it creates *gen_by* edges for nodes from the set out which has the same dependency list. Then it creates *used* edges to connect to the nodes in in set from any of these non-functional invocation nodes. It will connect with respective non-functional data node created in the last stage in case an edge needs to be created with an invocation either from in or out. This outcome is shown in Fig. 10(b).



(a) Stage 1 (b) Stage 2 (c) Fix Stage 3

Fig. 10. Repairing No-False Dependence Policy Violations

Algorithm: CALCULATECUSTOMPG
INPUT: provenance graph PG, user requests U and provenance policies PP
OUTPUT: customized provenance graph CG

1. Test for *Direct Conflicts* // as explained in Section 3.1
2. IF there are *Direct Conflicts* THEN
3. RETURN *false* // User can resubmit after changing U
4. ELSE
5. Compute PG′// as explained in Section 3.1
6. Transform *ur:hide* user requests into *ur:abstract* user requests // as explained in Section 4.2
7. Apply all *ur:abstract* user requests on PG′ to get $PG'^{\Delta u}$// as explained in Section 4.3
8. Resolve NCC violations on $PG'^{\Delta u}$// as explained in Section 4.1
9. Resolve NWC violations on modified $PG'^{\Delta u}$// as explained in Section 4.1
10. Resolve NFT violations on modified $PG'^{\Delta u}$// as explained in Section 4.1
11. CG = $PG'^{\Delta u}$// Final customized provenance graph
12. RETURN CG

Fig. 11. Computing CG using the *Inventing Non-Functional Nodes* approach

In the final stage, PROPUB combines nodes if possible. For example, in Fig. 10(b) the path from node '6' to node 'e' has three consecutive non-functional nodes with no other dependencies. These three nodes can be replaced by only one non-functional data node. The result is shown in Fig. 10(c). Now, PROPUB removes all the nodes selected to be abstracted and associated edges from PG′.

4.4 Algorithm

The algorithm mentioned in Fig. 11 finds the customized provenance graph, if available. In this approach, we add non-functional nodes to repair policy violations. In Figure 3 we show that $PG'^{\Delta u}$ has a structural policy violation between nodes g_1 and s_2. Using this approach, we introduce a non-functional data node d such that d is dependent on g_1; and s_2 is dependent on d. Now, to fix the cycle between d_{13} and g_1 we introduce the non-functional invocation node g_2 and create a dependency (*gen_by*) edge from d_{15} to g_2. Then, we get the final CG as shown in Figure 12. Note that we are now able to keep all the relevant nodes in CG.

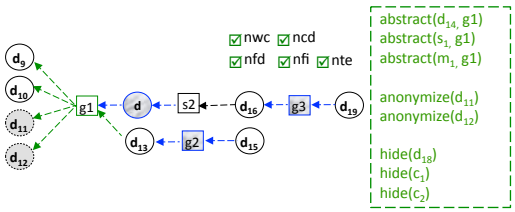


Fig. 12. Customized Provenance Graph after repairing all policy violations

5 Related Work

In [3,4,5,6,7], it has been observed that provenance can be used, e.g., to interpret results, diagnose errors, fix bugs, improve reproducibility, and generally to build trust on the final data products and the underlying processes. In addition, provenance can be used to enhance exploratory processes [15,16,17], and techniques have been developed to deal with provenance efficiently [18,19].

In many cases, provenance carries sensitive information, which can cause privacy concerns related to a data, actor, or workflow specification. Studying provenance, one can capture the functionality (being able to guess the output of the actor given a set of inputs) of an actor (module), or the execution flow of a workflow [8].

The security view approach [5] limits the available provenance to a user by providing a partial view of the workflow through a role-based access control mechanism, and by defining a set of access permissions on actors, channels, and input/output ports as specified by the workflow owner at design time. The ZOOM*UserViews approach [20] allows to define a partial, zoomed-out view of a workflow, based on a user-defined distinction between relevant and irrelevant actors. Provenance information is restricted by the definition of that partial view of the workflow.

In our recent work [11], we developed ProPub, which uses a declarative approach to publish customized policy-aware provenance. In this paper, we developed a new way to repair policy violations, not by removing additional nodes (as in [11]), but by introducing new (non-functional) nodes that represent the original lineage dependencies, without revealing information that the user wants to protect. We described in detail how policy violations will be repaired such that all relevant nodes are retained in the final provenance graph.

6 Conclusions

We discussed the need for provenance in scientific collaboration. Provenance data helps to build trust in the published results and data. However, provenance can also contain sensitive data and/or too much irrelevant detail. Thus, scientists should be able to "customize" provenance data before sharing it.

Our current ProPub system is based on the open provenance model (OPM). We plan to extend ProPub to include model extensions, e.g., to support structured data

structures, in particular nested collections [19]. Furthermore, PROPUB currently suggests only one specific modified graph based on a given U and PP. In future work, we plan to investigate how to extend this approach to rank alternative solutions, thus supporting scientists even more in finding the desirable balance between revealing provenance information and preserving privacy when sharing data with collaborators.

References

1. Nature: Special Issue on Data Sharing. Volume 461. (September 2009)
2. Missier, P., Ludäscher, B., Bowers, S., Dey, S., Sarkar, A., Shrestha, B., Altintas, I., Anand, M., Goble, C.: Linking multiple workflow provenance traces for interoperable collaborative science. In: Workflows in Support of Large-Scale Science (WORKS), 2010 5th Workshop on, IEEE 1–8
3. Bose, R., Frew, J.: Lineage retrieval for scientific data processing: a survey. ACM Computing Surveys (CSUR) **37**(1) (2005) 1–28
4. Simmhan, Y., Plale, B., Gannon, D.: A survey of data provenance in e-science. ACM SIGMOD Record **34**(3) (2005) 31–36
5. Chebotko, A., Chang, S., Lu, S., Fotouhi, F., Yang, P.: Scientific workflow provenance querying with security views. In: Web-Age Information Management, 2008. WAIM'08. The Ninth International Conference on, IEEE (2008) 349–356
6. Freire, J., Koop, D., Santos, E., Silva, C.T.: Provenance for Computational Tasks: A Survey. Computing in Science and Engineering **10**(3) (2008) 11–21
7. Davidson, S., Khanna, S., Roy, S., Boulakia, S.: Privacy issues in scientific workflow provenance. In: Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science, ACM (2010) 1–6
8. Davidson, S.B., Khanna, S., Panigrahi, D., Roy, S.: Preserving Module Privacy in Workflow Provenance. CoRR **abs/1005.5543** (2010)
9. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., et al.: The open provenance model core specification (v1. 1). Future Generation Computer Systems (2010)
10. Anand, M., Bowers, S., Ludascher, B.: Provenance browser: Displaying and querying scientific workflow provenance graphs. In: Data Engineering (ICDE), 2010 IEEE 26th International Conference on, IEEE (2010) 1201–1204
11. Dey, S., Zinn, D., Ludäscher, B.: PROPUB: Towards a Declarative Approach for Publishing Customized, Policy-Aware Provenance. In: Scientific and Statistical Database Management Conference (to appear). (2011)
12. Moreau, L., Ludäscher, B., Altintas, I., Barga, R., Bowers, S., Callahan, S., Chin, J., Clifford, B., Cohen, S., Cohen-Boulakia, S., et al.: Special issue: The first provenance challenge. Concurrency and Computation: Practice and Experience **20**(5) (2008) 409–418
13. Moreau, L., Clifford, B., Freire, J., Gil, Y., Groth, P., Futrelle, J., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Simmhan, Y., Stephan, E., den Bussche, J.V.: OPM: The Open Provenance Model Core Specification (v1.1). http://openprovenance.org/ (December 2009)
14. Anand, M., Bowers, S., McPhillips, T., Ludäscher, B.: Exploring scientific workflow provenance using hybrid queries over nested data and lineage graphs. In: Scientific and Statistical Database Management, Springer (2009) 237–254
15. Davidson, S., Freire, J.: Provenance and scientific workflows: challenges and opportunities. In: SIGMOD Conference, Citeseer (2008) 1345–1350

16. Freire, J., Silva, C., Callahan, S., Santos, E., Scheidegger, C., Vo, H.: Managing rapidly-evolving scientific workflows. Provenance and Annotation of Data (2006) 10–18
17. Silva, C., Freire, J., Callahan, S.: Provenance for visualizations: Reproducibility and beyond. Computing in Science & Engineering (2007) 82–89
18. Heinis, T., Alonso, G.: Efficient Lineage Tracking For Scientific Workflows. In: Proceedings of the 2008 ACM SIGMOD conference. (2008) 1007–1018
19. Anand, M., Bowers, S., Ludäscher, B.: Techniques for efficiently querying scientific workflow provenance graphs. In: Proceedings of the 13th International Conference on Extending Database Technology, ACM (2010) 287–298
20. Biton, O., Cohen-Boulakia, S., Davidson, S.: Zoom* userviews: Querying relevant provenance in workflow systems. In: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment (2007) 1366–1369

C-Set : a Commutative Replicated Data Type for Semantic Stores

Khaled Aslan[1] and Pascal Molli[1] and Hala Skaf-Molli[1] and Stephane Weiss[2]

[1] University of Nantes, France
[2] INRIA Rennes, France

Abstract. Web 2.0 tools are currently evolving to embrace semantic web technologies. Blogs, CMS, Wikis, social networks and real-time notifications, integrate ways to provide semantic annotations and therefore contribute to the linked data and more generally to the semantic web vision. This evolution generates a lot of semantic datasets of different qualities, different trust levels and partially replicated.
This raises the issue of managing the consistency among these *replicas*. This issue is challenging because semantic data-spaces can be very large, they can be managed by autonomous participants and the number of replicas is unknown.
A new class of algorithms called *Commutative Replicated Data Type* are emerging for ensuring eventual consistency of highly dynamic content on P2P networks. In this paper, we define C-Set a CRDT specifically designed to be integrated in Triple-stores. C-Set allows efficient P2P synchronisation of an arbitrary number of autonomous semantic stores.

Keywords: *scalability, synchronisation, peer-to-peer, consistency, replication*

1 Introduction

Web 2.0 tools are currently evolving to embrace semantic web technologies. Blogs, CMS, Wikis, social networks and real-time notifications, integrate ways to provide semantic annotations and therefore contribute to the linked data [1] and more generally to the semantic web vision. This evolution generates a lot of semantic datasets of different qualities, different trust levels and partially replicated.

The problem of updating and synchronizing data in the semantic web has been raised by Berners-Lee in [10]. Efficient synchronization of semantic stores organized in a peer-to-peer network can leverage problems of scalability and allows different autonomous participant to collaboratively combine, improve and enrich semantic datasets.

Synchronizing semantic stores is challenging. Semantic data-spaces can be very large, they can be managed by autonomous participants and the number of replicas is unknown, potentially high. Only few replication algorithms can fulfill these constraints and they all belong to the optimistic replication class

of algorithms [9]. An optimistic replication model considers an unknown number of sites, each site has a copy a shared object. An object can be modified by applying locally an operation. Next, this operation is broadcasted to other sites in order to be re-executed. The system is correct if it ensures the eventual consistency property [5] i.e. all replicas are identical when the system is idle. Thomas write rule [5] was the first algorithm to ensure eventual consistency in duplicated databases. However, Thomas write rule requires the knowledge of the number of participants (in order to provide a safe garbage collection scheme). This constraint is not compatible with our context.

Recently, a new class of optimistic replication algorithms called CRDTs is emerging [7,14]. CRDT stands for *Commutative Replicated Data Types*. CRDTs propose to define new abstract data type that provides commutative operations. CRDTs ensure eventual consistency regardless of the order of operations at reception. CRDT has been defined for arrays, linear sequence and tree.

In this paper, we define a specific data type for sets called C-Set that can be applied for a triple store and we prove that this type ensures eventual consistency. With a traditional set, operations insert/delete do not commute. C-Set provides commutative insert, delete operations, does not require causal reception of operation and can leverage some problems of garbage collection. C-Set is designed to be integrated within a semantic store in order to provide P2P synchronisation of autonomous semantic store.

The paper is organized as follow: Section 2 presents backgrounds and related works. Section 3 defines a new CRDT type for set designed for triple stores. Section 4 discusses our approach. Section 5 summarizes contributions and presents future work.

2 Background and related work

Many previous work on replication in semantic P2P systems focused on sharing RDF resources. They did not enable collaborative working for maintaining RDF stores. In the context of data sharing, some sites publish their RDF data while other sites consume this data. So there is no concurrent modifications and/or operations. While collaborative systems consider a shared object between the peers. This imply that the peers can modify the object concurrently and then they can run a consolidation algorithm to ensure the consistency of the shared object.

Tim Berners-Lee and Dan Connolly proposed an ontology for the distribution of differences between RDF graphs called *Delta* [10]. In their approach they rely on the standard serialization of RDF graphs into text files then running a *diff* between the resulted text files. However, the authors do not detail how eventual consistency can be efficiently reached by their algorithms.

RDFGrowth [12] and Publish/Subscribe Networks [4] focus on semantic data sharing where only one peer can modify the shared knowledge while others can read them. However, as it was mentioned earlier, sharing is different from collaboration. In sharing, some peers publish data while others can only read these

data and concurrent updates are not managed. In collaborative working environments, some peers publish data, others can read and write these data and a synchronization algorithm integrates concurrent updates. Collaborative environments improve the quality of data, the experience of the collaborative Wikipedia demonstrates this.

Edutella [6] proposes a RDF-based metadata infrastructure for P2P applications. It focuses on querying RDF metadata stored in distributed RDF repositories. The objective is providing access to distributed digital educational resources. Edutella distinguishes between two kinds of peers, simple and super peer. Simple peers provide data resource along with its schema. Super peers are used for several purposes, including data mediation, integration and query routing. Edutella proposes a replication service. However, it is not mentioned how to replicate and synchronize metadata.

RDFSync [11] synchronizes a target RDF graph with a source one. RDF graphs are decomposed unequivocally into minimal subsets of triples (Minimum Self-Contained Graphs MSGs) and canonically represented by ordered lists of the identifiers (hashes) of its composing MSGs. The synchronization algorithm performs a *diff* between the source and the target of the ordered list of MSGs. RDFSync can perform three kinds of synchronization, in the Target Growth Sync (TGS) the target becomes equal to the merge of both graphs, in the Target Erase Sync (TES) the target deletes unknown information by the source and finally in Target Change Sync (TCS) the target becomes equal to the source. Figure 1 shows an example of RDF graph synchronization using RDFSync.

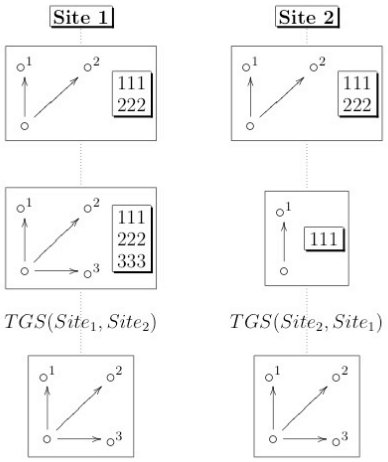


Fig. 1: RDF synchronization using RDFSync

RDFPeers [3] is a scalable distributed RDF repository. It is based on a structured P2P network. To enable faults tolerance, RDFPeers uses partial replication of RDF data. Every RDF triple is stored at three nodes of the network. However, it is not explicitly specified what happens in the case of concurrent updates on copies.

Thomas write rule [5] present techniques by which a number of loosely coupled processes can maintain duplicate copies of a database, despite the unreliability of their only means of communication. The copies of the database can be kept consistent. However, in order to remove the old deleted entries "garbage collection" they propose the following scheme: each site could notify the other sites whenever it hears about a deletion. If these notifications are transmitted in order with the "normal" sequence of modifications, then upon receipt of such a notification a site can be sure that the sending site has delivered any outstanding assignments to the deleted entry, has marked it as deleted, and will not generate any new assignments to it. This implies the knowledge of all the sites in the system. This constraint is not compatible with the P2P networks context.

In summary, P2P Semantic Web replication researches focus on knowledge sharing and querying. No algorithm have been designed for collaborative editing of RDF graphs. Consequently, they do not take into account concurrent updates on RDF graphs.

3 C-Set definition

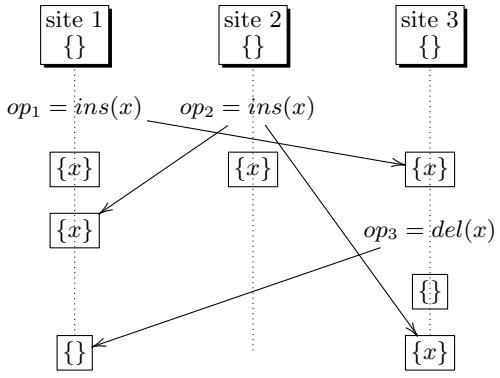


Fig. 2: A Traditional set counter-exemple

A CRDT is a data structure where all concurrent operations commute, so they do not require merge algorithms or integration procedure to enforce consistency. The replicas of a CRDT converge automatically, without complex concurrency control. Examples of algorithms that implement this data type are Woot [7],

TreeDoc [8], Logoot [13] and Logoot-Undo [14]. None of the existing algorithms handles the set data type.

A set with operations $insert(element)$ and $delete(element)$ is not a CRDT. The counter-example is presented in figure 2. The example illustrates how three sites can receive operations op_1, op_2, op_3 in different order. Site 1 executes the sequence $[op_1; op_2; op_3]$ while Site 3 executes the sequence $[op_1; op_3; op_2]$. If the execution of operations does not commute, then eventual consistency is violated i.e. the set on site 1 is empty while the set on site 3 contains $\{x\}$.

3.1 C-Set Data Structure

We want to define a CRDT for sets data type, where each element in this set corresponds to an RDF triple. We represent the set S of elements as a set of a couple of $(e : element, x : \mathbb{Z})$. We define on this set four operations : $ins(e : element)$, $del(e : element)$, $rins((e : element, k : \mathbb{Z}))$ and $rdel((e : element, k : \mathbb{Z}))$ detailed in the following section.

3.2 C-Set Algorithms

The operation $ins(e : element)$ can be executed locally immediately. It generates and sends remote insert operation $rins((e : element, k : \mathbb{Z}))$ that is executed remotely. In our model we give the *ins* operation precedence over the *del* operation. So whenever an *ins* operation is executed it compensate the effect of all the previously received *del* operations.

```
ins(e : element):
if (∃k ∈ ℤ : (e, k) ∈ S) then
 if (k ≤ 0)
    S = (S/{(e, k)}) ∪ {(e, 1)};
    send(rins((e, +|k| + 1)));
 else if (k > 0)
        S = (S/{(e, k)}) ∪ {(e, k + 1)};
        send(rins((e, +1)));
      endif
 endif
else
  S = S ∪ {(e, 1)};
  send(rins((e, +1)));
endif
```

Algorithm 1: *ins* algorithm

```
rins((e : element, k : ℤ*)):
if (∃i ∈ ℤ : (e, i) ∈ S) then
  S = (S/{(e, i)}) ∪ {(e, k + i)};
else
 S = S ∪ {(e, k)};
endif
```

Algorithm 2: *rins* algorithm

The operation $del(e : element)$ is executed locally. It generates and sends the remote delete operation $rdel((e : element, k : \mathbb{Z}^*))$ that is executed remotely.

```
del(e : element):
if (∃k ∈ ℤ : (e, k) ∈ S)
  if (k ≤ 0) then
     S = (S/{(e, k)}) ∪ {(e, k − 1)};
     send(rdel((e, −1)));
  else // k>0
    S = S/{(e, k)};
    send(rdel((e, −k)));
  endif
endif
```

Algorithm 3: *del* algorithm

```
rdel((e : element, k : ℤ*)):
if (∃i ∈ ℤ : (e, i) ∈ S)
        S = (S/{(e, i)}) ∪ {(e, i + k)};
else
        S = S ∪ {(e, +k)};
endif
```

Algorithm 4: *rdel* algorithm

Figure 3 shows an execution example of C-Sets. The first execution corresponds to the example presented in figure 2. Site 1 executes sequence $[op_1; op_2; op_3]$ which in fact executes $[(+1) + (+1) + (-1)]$ on the counter associated to x value in the C-Set. When the same operations are executed in a different order on site 3, the sequence $[op_1; op_3; op_2]$ force the computation of $[(+1) + (-1) + (+1)]$ on the counter associated to x value in the C-Set. Of course both sites converges. An alternative execution is also presented in figure 3. This execution illustrates how the operation op_3 can trigger the sending of operation $rdel(x, -2)$.

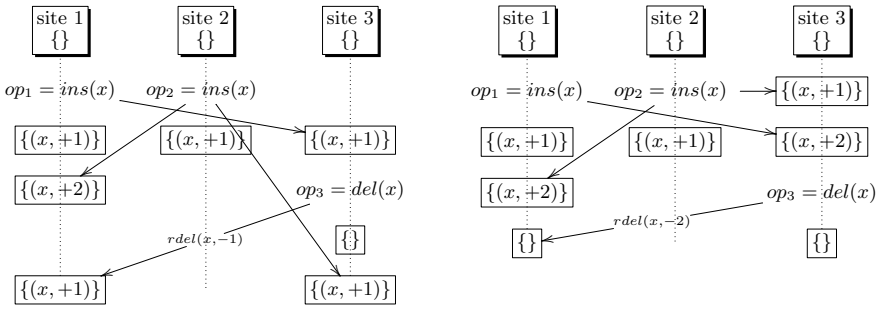


Fig. 3: C-Set : Two convergent executions

The proof that C-Set preserves eventual consistency is straightforward. On each site, C-Set generates sequence of additions of elements that belong to $(\mathbb{Z}, +)$. As addition in $(\mathbb{Z}, +)$ is commutative, C-Set converge.

4 Discussions

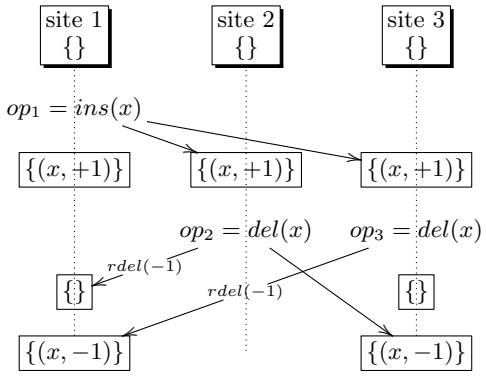


Fig. 4: C-Set and Tombstones

C-Sets is a CRDT data type for sets that ensures eventual consistency. However, as shown in figure 4, C-Set, as the Thomas write rule, relies on tombstones. This means that a deleted element remains in the store. In the example in figure 4 the final value $\{(x,-1\}$ is clearly a tombstone. Garbage collecting tombstones is a challenging issue in a dynamic P2P network as in our context. It requires each site to have knowledge about the state of all the others sites. Consequently it makes the synchronisation dependant of the number of sites and requires procedures to join and leave synchronisation groups.

An interesting property of C-Sets is that tombstones can be removed locally on one site without communication with others sites when counters associated to set elements are equal to 0. Tombstones will remain if same elements are concurrently deleted on several sites which is not the more frequent scenario. Extensive experimentation is needed to prove the efficiency of this hypothesis.

5 Conclusion and perspectives

In this paper we presented C-Set : a CRDT for sets that ensure eventual consistency. C-Set is designed to be integrated within a semantic store in order to provide P2P synchronisation of autonomous semantic store. C-Sets is more efficient than Thomas write rule, especially in managing the delete operations.

Future work will integrate C-Set within an existing semantic store. Next, we will be able to to evaluate the overhead of C-Sets using real-world semantic web data such as DBpedia [2].

References

1. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 4(2):1–22, January 2009.
2. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, September 2009.
3. Min Cai and Martin Frank. RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network. In *Proceedings of the 13th international conference on World Wide Web*, pages 650–657. ACM, 2004.
4. P.A. Chirita, Stratos Idreos, Manolis Koubarakis, and Wolfgang Nejdl. Publish/-subscribe for rdf-based p2p networks. *The Semantic Web: Research and Applications*, pages 182–197, 2004.
5. P. Johnson and R. Thomas. RFC677: The maintenance of duplicate databases. 1976.
6. Wolfgang Nejdl, Boris Wolf, Changtao Qu, and Stefan Decker. EDUTELLA: a P2P networking infrastructure based on RDF. *Proceedings of the*, 2002.
7. Gérald Oster, Pascal Urso, Pascal Molli, and Abdessamad Imine. Data Consistency for P2P Collaborative Editing. In *Conference on Computer-Supported Cooperative Work*, 2006.
8. Nuno Preguica, Joan Manuel Marques, Marc Shapiro, and Mihai Letia. A Commutative Replicated Data Type for Cooperative Editing. *2009 29th IEEE International Conference on Distributed Computing Systems*, pages 395–403, June 2009.
9. Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Computing Surveys*, 37(1):42–81, March 2005.
10. Berners-Lee Tim and Connolly Dan. Delta: an ontology for the distribution of differences between RDF graphs. *http://www.w3.org/DesignIssues/Diff*, 2004.
11. Giovanni Tummarello, Christian Morbidoni, R. Bachmann-Gmur, and Orri Erling. RDFSync: efficient remote synchronization of RDF models. *Proceedings of the ISWC/ASWC2007*, pages 537–551, 2007.
12. Giovanni Tummarello, Christian Morbidoni, Joakim Petersson, Paolo Puliti, and F. Piazza. RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications. *The First International Workshop on Peer-to-Peer Knowledge Management*, 2004.
13. Stéphane Weiss, Pascal Urso, and Pascal Molli. Logoot : a scalable optimistic replication algorithm for collaborative editing on p2p networks. In *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2009.
14. Stephane Weiss, Pascal Urso, and Pascal Molli. Logoot-undo: Distributed collaborative editing system on p2p networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(8), 2010.