Semantic-enabled and Hypermedia-driven Linked Service Discovery

Mahdi BENNARA[1], Michael MRISSA[2], and Youssef AMGHAR[1]

[1] Université de Lyon, LIRIS
INSA-Lyon - CNRS UMR5205
F-69621, France
{mahdi.bennara,youssef.amghar}@liris.cnrs.fr
[2] Université de Lyon, LIRIS
Université Lyon 1 - CNRS UMR5205
F-69622, France
michael.mrissa@liris.cnrs.fr

Abstract. Automating discovery and composition of RESTful services with the help of semantic Web technologies is a key challenge to exploit today's Web potential. In this paper, we show how semantic annotations on resource descriptions can drive discovery algorithms on the Web. We propose a semantically-enabled variant of the BFS discovery algorithm that aims at minimizing the number of links explored while maximizing result diversity. Our algorithm calculates semantic distances between resource descriptions and user request concepts to rank explored resources accordingly. We demonstrate the applicability of our solution with a typical scenario and provide an evaluation with a prototype.

Keywords: Linked Web services, semantic web, discovery, composition

1 Introduction

During the last few years, both the overall number of Web APIs exposed on the Web[3] and the increasing ratio of RESTful APIs has shown the interest of a Web of resources. Resources can be combined to answer complex user requests, in other words: to build Web applications. The emergence of RESTful services[4] has been a major success to enable interoperation on the Web. Moreover, service composition, or mashups, enables valued-added processes that combine several services to answer complex user needs. The success of Web services is highlighted via Web sites such as `http://www.programmableweb.com` that referenced 105 APIs available on the Web in 2005 and more than 15000 APIs in 2016, not counting mashups. The uniform interface, that comes with the correct use of HTTP verbs and their semantics, replaces the typical API build around functions

[3] http://www.programmableweb.com/api-research
[4] In the remainder of this paper, we use "resource" to describe a RESTful service accessed through a URI endpoint.

and sets of input/output parameters. The operations handled by server-side modules are now shifting towards the client modules. On top of that, recent advances in the semantic Web research area have been promoting linked data [2] and a set of languages and tools, such as RDF [10], that allow to annotate Web data, resources and services with explicit, machine-readable semantics that can be utilized in conjunction with advanced reasoning mechanisms to connect resources to each other. Linked services [13] benefit from these machine-readable semantic annotations.

Our paper is organized as follows. Section 2 introduces the motivating scenario of our contribution and details the research problem. Section 3 explains some background knowledge in order to understand our contributions. Section 4 presents related work and highlights the advantages our solution offers. Section 5 details how we semantically annotate resource descriptions and hypermedia links. Section 6 gives an evaluation and discusses the choices made and results obtained. Section 7 resumes our approach and lists some elements of future work.

2 Motivating scenario and research problem

2.1 Scenario and Motivation

We motivate our contribution with an online book selling scenario where a user wants to buy a book, make a payment online then get the book shipped to the given address. The process of buying involves selecting a set of books, choosing a shipping method, and paying with the appropriate solution. We assume that the URI of one of the book selling resources is known to the user. Our work is motivated by the need to enable distributed affordance principle [14], which means that the resource discovery process should be automated and hypermedia-driven (consisting in following links between resources). The advantages of distributed affordance include the possibility of generating opportunities of use for resources while exploring the Web as well as respecting the user preferences. Automating the discovery process can typically be achieved with the help of semantic annotations that can aid software agents decide what are the relevant links to follow. Building a solution to enable distributed affordance includes two elements. Discovery requires **semantic description of resources**. Semantic annotations provide the means to reason about the descriptions. Such descriptions should follow the HATEOAS principle [5]. Exploring the Web requires a **resource discovery algorithm**. It must make appropriate use of the semantic annotations on resources to optimize the search response time. The end user should only provide high level objectives to the software program, as well as an entry point (URI to start the discovery process). The software program should be in charge of interpreting the user request, finding out that buying a book online includes selecting a set of books, choosing a delivery option and paying online. It should explore the Web of resources to discover the ones that help answering the query,

[5] Hypermedia as the Engine of Application State

orchestrate the interactions and execute the created process. These are the challenges we address in the current paper.

2.2 Problem statement and research contribution

Typical approaches to discover, compose, orchestrate and utilize linked services on the Web require complete overhauling of existing technique in order to harness the benefits provided by the REST principles and the semantic Web. In the present paper, we propose a generic solution to automate resource discovery based on semantic reasoning and following the HATEOAS principle. We follow the distributed affordance principle [14] and rely on an extension [1] to the Hydra [8] specification for semantically describing resources. Our contribution is two-fold:

– Resource description: The extension of our previous work on descriptions (presented in Sec. 5) describes the business-level semantics of HTTP operations on resources as well as links to other resources. Annotations on operations allow to automate the identification the tasks required by the user, and annotations on links guide the discovery process to other resources.
– Resource discovery: we extend the BFS [6] algorithm with semantic-awareness to improve its response time. Our annotation extends the social model presented in [9] to semantically qualify the relationships between resources.

3 Background Knowledge

3.1 Graph search algorithms

Breadth-First-Seach: Exploring very large graphs like the Web [11] needs high performance algorithms in order to obtain low response times. Efficiency of the exploration algorithm is key for our work, as we are discovering resources on the Web. Breadth-First-Search [7] algorithm yields high performances in large graphs, according to [12]. In addition, BFS is a natural search strategy in the context of Web. Also, compared to other efficient search algorithms, it has a relatively low computational cost for a large scale graph such as the Web. In our approach, BFS finds the most relevant resources to answer a user's request early enough to be considered efficient.

Depth-First-Search: DFS is also one of the well-known graph exploring algorithms [7]. It explores branches one by one and it stops only when it reaches the deepest node of that branch, the deepest node being the one with no successor. The application of DFS in Web crawling has proved to be difficult. This is due to the fact that the Web is a very large graph, and exploring one branch only can be extremely difficult, performance-wise. Most relevant nodes to the current research are generally not very deep but rather in different branches, which is the main weakness of the algorithm.

[6] Breadth First Search [7]

3.2 Web Resource Description

In this paper, we rely on the RESTful resource descriptor mechanism as well as and the discovery solution introduced in [1] in order to propose a solution for resource discovery problem in the context of semantic Web. The descriptor notion separates resource representation from its description and states that every resource must have a descriptor and a representation. The reason for detaching the representation from the description is to separate different concerns, in order to ease the resolution of each one apart. Resource representations relate to user Web browsing, while resource descriptions relate to M2M interactions and operations such as discovery and composition processes. The descriptor is typically accessed by calling a GET / HEAD operation on the resource URI, and retrieving the LINK header element in the HTTP response. A GET on the retrieved link returns a HTTP response whose body contains the descriptor itself.

4 Related Work

4.1 Semantic Description of Resources

Hydra core vocabulary: The Hydra core vocabulary [8] is a small vocabulary aimed to describe RESTful Web APIs. The purpose of developing the Hydra vocabulary is to simplify the development of RESTful APIs by leveraging the advantages offered by Linked Data. The basic idea of Hydra vocabulary is to allow RESTful APIs to publish valid state transitions to clients. As a result, the clients can utilize this information in order to construct valid HTTP requests in order to modify the resource, request/delete another one or create a new one. All this information is exchanged between server and client at run-time and is not hard-coded into client at design time. Hence, the clients can be decoupled from the servers and adapt their execution according to changes.

RESTdoc: RESTdoc [5] is a description solution that combines multiple micro-formats in order to semantically describe RESTful resources. RESTdoc offers also a discovery mechanism that distinguishes two different aspects of REST services discovery problem: (1) the discovery as a client concerns the client-side browsers. This discovery uses on HTML Link element on a Web site in order to point to other resource descriptions, and (2) the discovery as a service which is is the ability for a service to access and link to other related resources in the same application domain. This solution describes a fully peer to peer discovery mechanism. In our work we combine both discovery modes.

4.2 Resource discovery and composition

LinkedWS: LinkedWS [9] is a Web service discovery model based on human interactions in social networks in the context of SOA. The idea behind LinkedWS is to establish a social network of Web services where nodes are actual Web services and edges are relations between these Web services. What is really important about this work in the current paper is the categorization established on exposed functionality. These functionalities are either `Similar` so the Web services compete for participation in compositions or `Complementary`, so the Web services work towards the same compositions.

RESTdesc: RESTdesc [15] is a work about semantic description of Web APIs based on the Notation3 RDF syntax. The purpose of the description is to allow for an efficient way to discover the features that Web APIs offer. It uses operational semantics of Notation3 in order to allow a flexible discovery.we think that the N3 descriptions require a lot of effort to work with, even though N3 reasoning has seen a good advance. Our descriptor mechanism is aimed specifically to render descriptions use simple for both machines and Web API developers.

4.3 Analysis

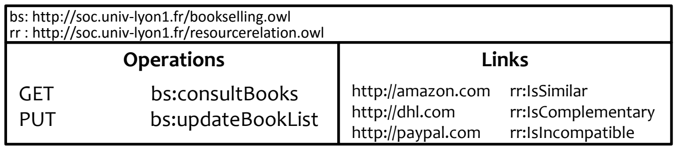
Using semantic Web advances to automate resource composition is a recent topic and has only been explored by few works in the literature [6]. The contribution we propose in this paper relies on exploring the semantic annotations over the links between resources, which are found in descriptors, in order to guide the discovery process into the links with the most potential to match with the request concepts. We reuse the Hydra core vocabulary in order to establish descriptions. However Hydra does not provide a good support for semantic annotations over links and operations. We extend hydra in order to allow resources description to have semantically annotated elements that can be exploited by discovery, selection and composition algorithms in order to enable a completely automated process to answer user's requests. The simplicity of our solution lies in the separation between resource representation and description as well as the separation between links and operations in the descriptions. In order to discover resources that can answer the user's request, the generic client has to start exploring the description of the resource given as an entry point by the user. Based on the semantic annotations given by the description, the decision making of (1) whether to account the current resource in the final composition and (2) what are the next resources to explore is easy to establish.

5 Contribution

5.1 Semantically Annotating Descriptor Links

Our descriptor-based solution allows generic clients to crawl from one resource to another in order to select interesting resources to answer the user's query. However, due to the huge number of resources on the Web, there is a need to improve the discovery algorithm that we use (i.e. BFS [7]) to only select the most interesting resources. The vocabulary we use to implement the descriptor concept is Hydra core vocabulary [8]. We introduce semantic annotations on descriptor links, and extend the BFS algorithm to take advantage of these annotations. The semantic annotations will guide the algorithm by excluding irrelevant links to the current application. Fig 1 illustrates the semantic annotation of descriptor links. Our semantic annotation is inspired from existing work [9] to define the properties that link resources to each other. We define that:

– Two resources are `similar` if they provide functionally substitutable services, sometimes varying in terms of non-functional properties.
– Two resources are `complementary` if they can be combined in the same process to answer user's needs, for example: a flight booking service and a hotel booking service inthe context of a trip.
– Two resources are `incompatible` if they cannot be involved together in the same process because of a given reason, for example: the eBay online seller could decide not to work with the UPS delivery company.



bs: http://soc.univ-lyon1.fr/bookselling.owl
rr : http://soc.univ-lyon1.fr/resourcerelation.owl

Operations		Links	
GET	bs:consultBooks	http://amazon.com	rr:IsSimilar
PUT	bs:updateBookList	http://dhl.com	rr:IsComplementary
		http://paypal.com	rr:IsIncompatible

Fig. 1. Descriptor example with annotated links

5.2 Semantic-enabled Discovery

Our solution builds a generic client that interacts with the resources through their respective APIs. The client software program needs to be able to automate the process of composing the functionality of the three resources of the scenario to answer the user's query. This includes the discovery of the resources. The maximum number of `similar` links to be operated can be limited in order to increase the performance of the BFS algorithm. However, this will limit the choices given to the user. A compromise between performance and result diversity is to be established using this parameter.

We also propose a solution inspired by the weight-based approach presented in [3] in order to :

– Sort the links on a resource description in order to guide the discovery algorithm while exploring similar links.
– Sort the results obtained after positive matching with a query concept

In other words, the set of similar resources inside a resource descriptor are sorted from the most similar link into the least similar one. Based on the query nature, the discovery algorithm starts exploring the most similar resources if the priority is to find more alternatives to the current resource or the least similar resources if the priority is to find more complementary and diverse resources. Many formulas to calculate semantic distance have been proposed in the state of the art [3, 4]. The one we adopt in our work is weight-based formula proposed in [3] because it can be directly used with our approach without any further calculation of additional parameters. Note that the work of annotating links and sorting them is not done during the discovery.

The discovery algorithm details are given in Algorithm 1.

Algorithm 1: BFS-based discovery algorithm

```
Input: conceptList: array of string
Input: currentLink: string
Input: similarLimit : integer
Output: result: array of string
bfsQueue: array of string
visited: array of string
while not conceptList.empty() and not bfsQueue.empty() do
    if not currentLink in visited then
        visited.insert(currentLink)
        Descriptor descriptor = getDescriptor(currentLink)
        foreach operation in descriptor.operations do
            foreach concept in conceptList do
                if conceptMatch(operation.annotation, concept) then
                    result.insert([concept, currentLink])
                    conceptList.remove(concept)
        similarCount: integer = 0
        foreach link in descriptor.links do
            if link.annotation = IsComplementary then
                bfsQueue.insert(link)
            else
                if link.annotation = IsSimilar and similarCount < similarLimit
                then
                    bfsQueue.insert(link)
                    similarCount = similarCount + 1
            //and if it is incompatible we do not take it into account in the first
            place
    currentLink = bfsQueue.next()
```

The algorithm takes as input three parameters:

- `conceptList` (array): contains the list of concepts that describe the operations needed in order to answer the user's query.
- `currentLink` (string): contains the URI of the resource being processed.
- `similarLimit`(int): is the maximum number of similar links per resource to be taken into account by the algorithm.

The algorithm returns as output the `result` array which contains all the pairs [`concept`,`URI`] where the resource identified by `URI` can perform an operation that semantically matches the paired `concept` classified by semantic distance from the query concept.

The set of variables used in this algorithm are the following:

- The `bfsQueue` is the queue that contains the ordered set of URIs for the next nodes to be explored by the BFS algorithm.
- The `visited` array contains URIs of resources already traveled. This variable's main objective is to prevent loops if the graph is cyclic. Further improvements on this part of the algorithm are possible in order to obtain better performance.
- The `similarCount` variable introduced in line 12 counts the number of `similar` links that are inserted in the BFS queue to be traveled. This counter cannot exceed `similarLimit`.

The algorithm consists of a main `While` loop. The exit condition is verified when there are no concepts to look for or no further resources in the graph to travel or when a certain amount of time passed since the beginning of the loop (timeout). Each iteration of this loop discovers a single resource whose link is `currentLink`. The algorithm verifies if it has not been visited yet, if not it is marked as visited. If the resource has not been processed yet, the algorithm gets its descriptor then checks if any of the operations provided by the resource is annotated by one of the remaining concepts. If so, the concept along with the resource URI are inserted into `result` then the concept is removed from `conceptList`. After that, the algorithm inserts the URIs of the related resources into `bfsQueue`, while respecting the fact that similar resources links inserted cannot exceed `similarLimit`.

6 Evaluation and Discussion

The resources composing the services previously presented in the scenario are implemented using Java TM Servlets using Jersey framework 7. We use Apache Tomcat 8 as a server-side software in order to accommodate our resources. The demonstration Web page can be found here: `https://liris.cnrs.fr/~mbennara/doku.php?id=medi2016`.

We show the number of traveled nodes as well as response time (in milliseconds) gain compared to the raw BFS algorithm respectively in Fig 2 and Fig 3. Each column represents a separate query that involves an increasing number of resources in the Web. We get better response times for the same request with the enhanced algorithm because it explores less nodes than the regular. This is due to the fact that when we travel the Web graph, we find more similar resources. The similar resources are ignored by the enhanced algorithm but taken into account by the regular one. However, this decrease in response time can also be accompanied by a decrease in result diversity.

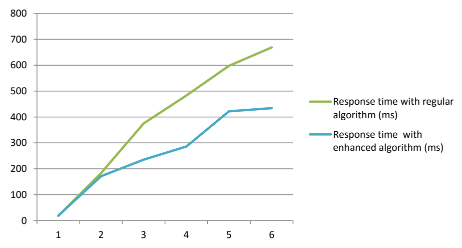


Fig. 2. Response time in ms

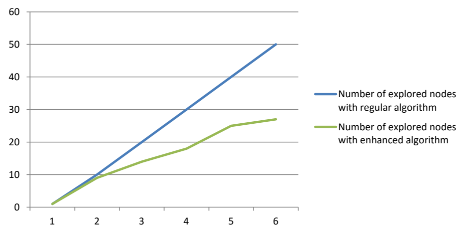


Fig. 3. Number of explored nodes

Enabling semantic annotations on links between resources allows the automation of the discovery process. Without the semantic annotations, the discovery algorithm has to explore every link in order to search for resources to answer the user's query. Having similar and complementary annotations on links allows the algorithm to explore the requested links based on selectivity measures. The maximum number of similar links to be explored is limited. This limit determines the performances of the discovery algorithm as well as the diversity of the results obtained. The lack of diversity is due to the possibility for similar resources to contain links into useful complementary resources. Sorting the similar resource links in the description is important in order to optimize the discovery algorithm. Depending on the user's query, the discovery process will prioritize the most or the least similar links while taking into account the similar limit as well.

7 Conclusion

In this paper we propose an annotation of Web resource descriptions based on a social model that relies on similar and complementary relations. These annotations provide information for the discovery process in order to respond to the user's request faster and more accurately. Then, we provide a semantically-enhanced BFS-based algorithm to discover resources. It relies on the semantic annotations in order to determine whether a resource is worth exploring.

Future work includes exploring advanced heuristics to reach a better compromise between performance and result diversity. We envision to extend our model to support quality of service aspects in order to further enhance the discovery and selection processes. We aim also to enable an automatic service composition process in order to fully automate answering users' requests.

8 Acknowledgment

We would like to thank Mehdi Terdjimi for his support and help with the implementation of the algorithm, the ontology and the concept matching component.

References

1. Bennara, M., Amghar, Y., Mrissa, M.: Managing web resource compositions. In: Reddy, S. (ed.) 24th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE Workshops 2015, Larnaca, Cyprus, June 15-17, 2015. pp. 176–181. IEEE (2015)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
3. Ge, J., Qiu, Y.: Concept similarity matching based on semantic distance. In: Semantics, Knowledge and Grid, 2008. SKG'08. Fourth International Conference on. pp. 380–383. IEEE (2008)
4. Hau, J., Lee, W., Darlington, J.: A semantic similarity measure for semantic web services. In: Web Service Semantics Workshop at WWW. pp. 10–14 (2005)
5. John, D., Rajasree, M.S.: RESTDoc: Describe, Discover and Compose RESTful Semantic Web Services using Annotated Documentations. International Journal of Web & Semantic Technology (IJWesT) 4(1) (2013)
6. Kovatsch, M., Hassan, Y.N., Mayer, S.: Practical semantics for the internet of things: Physical states, device mashups, and open questions. In: Internet of Things (IOT), 2015 5th International Conference on the. pp. 54–61. IEEE (2015)
7. Kozen, D.: Depth-first and breadth-first search. In: The Design and Analysis of Algorithms, pp. 19–24. Texts and Monographs in Computer Science, Springer New York (1992), `http://dx.doi.org/10.1007/978-1-4612-4400-4_4`
8. Lanthaler, M., Guetl, C.: Hydra: A Vocabulary for Hypermedia-Driven Web APIs. In: Bizer, C., Heath, T., Berners-Lee, T., Hausenblas, M., Auer, S. (eds.) LDOW. CEUR Workshop Proceedings, vol. 996. CEUR-WS.org (2013)
9. Maamar, Z., Wives, L.K., Badr, Y., Elnaffar, S., Boukadi, K., Faci, N.: Linkedws: A novel web services discovery model based on the metaphor of "social networks". Simulation Modelling Practice and Theory 19(1), 121–132 (2011)
10. Manola, F., Miller, E.: RDF Primer. W3C Recommendation, `http://www.w3.org/TR/rdf-primer/`
11. Meusel, R., Vigna, S., Lehmberg, O., Bizer, C.: The graph structure in the web–analyzed on different aggregation levels. The Journal of Web Science 1(1) (2015)
12. Najork, M., Wiener, J.L.: Breadth-first crawling yields high-quality pages. In: Proceedings of the 10th international conference on World Wide Web. pp. 114–118. ACM (2001)
13. Pedrinaci, C., Domingue, J.: Toward the Next Wave of Services: Linked Services for the Web of Data. J. UCS 16(13), 1694–1719 (2010)
14. Verborgh, R., Hausenblas, M., Steiner, T., Mannens, E., de Walle, R.V.: Distributed affordance: an open-world assumption for hypermedia. In: Carr, L., Laender, A.H.F., Lóscio, B.F., King, I., Fontoura, M., Vrandecic, D., Aroyo, L., de Oliveira, J.P.M., Lima, F., Wilde, E. (eds.) WWW (Companion Volume). pp. 1399–1406. International World Wide Web Conferences Steering Committee / ACM (2013)
15. Verborgh, R., Steiner, T., Deursen, D.V., Roo, J.D., de Walle, R.V., Vallés, J.G.: Description and Interaction of RESTful Services for Automatic Discovery and Execution. In: Proceedings of the FTRA 2011 International Workshop on Advanced Future Multimedia Services (Dec 2011)