A Recommender System for Web Services Discovery in a Distributed Registry Environment

Mohamed Sellami\(^1\), Samir Tata\(^1\), Zakaria Maamar\(^2\) and Bruno Defude\(^1\)
\(^1\)Institut TELECOM, CNRS UMR Samovar, Evry, France mohamed.sellami,samir.tata,bruno.defude@it-sudparis.eu
\(^2\)Zayed University, Dubai, U.A.E zakaria.maamar@zu.ac.ae

Abstract

In this paper, we discuss the suitability of using recommendation techniques for Web service discovery in distributed registry environments. The architecture we adopt consists in structuring registries of Web services into groups. We propose to adapt existing techniques for recommendation to ease the services discovery process. This consist of (1) recommendation of groups of registries using semantic matching and (2) recommendation of registries within these groups using user-characterization based technique. To put our proposed distributed registry architecture in practise and to test the efficiency of recommendation-based discovery, we propose to simulate a Peer-to-Peer (P2P) registries environment on top of JXTA platform.

1. Introduction

The number of Web services and registries that are made available for use continues to increase almost on a daily basis. A service consumer aiming to discover a Web service faces the following dilemma that does not help make the discovery exercise more focused: a large number of registries exist and a large number of similar Web services exist. Hence, the time for Web services discovery can be too long to be acceptable. In this context, if appropriate solutions are not considered, “traditional” Web services discovery mechanisms that consist of scanning all the registries would for instance to slow down the increase rate of web services.

To address the issue in Web services discovery, innovative solutions are then required and could be built upon different existing techniques like recommender systems. Recommender Systems are a special kind of information filtering approaches, designed to help users tackle the recurrent problem of information overload [1]. These systems filter large volume of information prior to suggesting items to a user according to her interests or behavior as well as other users’ past experiences and recommendations. So, recommender systems can be adopted to ease Web services discovery by reducing the search space and hence positively impact on the complexity and the time needed for discovery.

To show the provision of recommendations for discovery and to motivate our work, let’s consider the traditional example of trip planning: A user would like to book a flight and hotel room and then find an itinerary from the destination airport to the hotel and a restaurant. To carry out all these activities, we suppose that the user is given the opportunity to interact with a distributed network of service registries. Registries are here structured and grouped according to some business domains like Hotel, Travel, Sport, etc. Traditionally, the user expresses a query in terms of functional and non-functional requirements. This query will be executed on registries chosen by the semantics of the query [2], keywords from the query [3] or through a P2P routing mechanism [4]. As response, he receives a list of Web services, resulting from the execution of his query over the registries, that should answer his needs and from which he has to select one. If the number of registries over which the query will be executed is important or if there is a large number of published Web services, this step might become cumbersome and prone to errors. Hence, in such cases, Web services discovery can be improved using recommendation techniques from other domains, e.g. invocation histories and users’ feedbacks. So, we argue that assisting users using techniques from other domains, for instance recommendation as it helps dealing with large volume of data, can improve the discovery process and enhance user’s confidence in the “quality” of selected Web services.

In this research work, we propose to enhance the Web services discovery process, especially the adequate registries selection step, using recommendation techniques. We focus our research on Web services discovery in an environment of distributed and heterogeneous registries. Using distributed registries instead of a centralized one provides higher availability of registries. However, service discovery in such environments can be time consuming especially if requester’s queries have to be applied on many registries.

To deal with these issues, we show in this paper how recommendation techniques can be adapted and adopted to enhance Web services discovery in an environment of several distributed groups of registries. Based on a user’s query and characterization\(^1\), our contributions are as follows: (1) select the “appropriate” group of registries by matching a user’s query with the available groups, (2) find the registry(ies) in the selected group that has the highest probability of satisfying the query using a recommender

\(^1\) We define a characterization as a data structure containing useful information for recommendation characterizing a service requester (areas of interests, invocation history, etc.) (See Section 5.2.1)
technique, and finally (3) run the query over the selected registry so that appropriate Web services are identified. This paper is organized as follows. Section 2 presents a review of some recommendation-based approaches for Web service discovery. Section 3, presents our architecture for recommendation-based Web service discovery. In Section 4, we introduce our approach for Web services discovery. Then, in Section 5, we present techniques used to put in practice our approach. Finally, our implementation prototype is presented in Section 6 and Section 7 concludes the paper.

2. Recommender systems for WS discovery

Recent research projects such as [5], [6], [7], [8], [9] used recommendations techniques to enhance Web services discovery. In [5], Aliaksandr et al propose an implicit recommender system to improve Web service Discovery. Their approach benefits from past services developer’s discovery experiences: observed data from requests and their corresponding service invocations and executions. This is done without any explicit interaction with the developers. The system uses a Web service based on the implicit culture theory of service developers (IC-service). For each user’s request, the IC-service collects and manages observation data from the service’s invocation and execution. Using these data, the proposed system can compute similarity between a user’s request and the theory’s antecedent to recommend services. In [6], the authors note that semantics and syntax are insufficient to discover a service that best suits user’s needs in a grid infrastructure. They propose to add to semantics and syntax two additional dimensions of service description, namely: quality and usage patterns. On the basis of those four services description dimensions, they propose an architecture for recommendation-based service mediation. To handle user requirements may change over time, Zhang et al. [7] propose to proactively recommend services to users. On the basis of the outputs of the previously-chosen services, they define the user’s Interest model. The recommendation is done by matching between available service’s inputs and the user interests. To ensure a better accuracy in the matching, the authors semantically annotate services descriptions and users interests. In [8], authors combine semantic matching of services descriptions with recommendation techniques. To select a service, a semantic matching between the user’s requirements and available services descriptions is executed. Then, the list of services resulting from this matching is ordered by a recommender system based on previous user’s ratings. That’s why the proposed system has to explicitly ask users to rate services. Some works used syntactical matching to recommend Web services. In [9] for example, Blake et al. propose to compute a Web service recommendation score by matching strings composing the user’s file (string data collected during a user’s operational sessions) and strings composing the registered Web services (messages names and operation names). To compute this score, they consider the syntactical heterogeneity that can be observed in the services descriptions (i.e. services offering functionally same inputs, but who are syntactically different) and propose a recommendation algorithm considering this point. On the basis of the obtained score, they can judge if a user might be interested in some services.

The approach that we propose is radically different from those mentioned above. In fact, the previous approaches were developed for service discovery in centralized registries while we aim at discovery in a distributed registry environment. In addition, rather than recommending Web services, in our approach we recommend registries having the better probability to satisfy a service requester’s need.

3. An Architecture for Recommendation-based WSs Discovery

When working in a distributed registry environment, there can be a large number of registries that also contain a large number of Web services’ description. Generally, to limit the search space during a service discovery process, several approaches structure registries into groups according to business domains, functional, or non-functional properties of the services they host [10], [2], [3]. By structuring service registries, the discovery exercise will be simplified and made more efficient and robust. Meanwhile, when the number of registries in a group is large, discovering the “best” service to a requester’s needs in a registry group remains complex. To gear a service requester’s query towards adequate services registries, we use a recommendation technique based on his areas of interests, invocation traces and non functional requirements (explanations on those choices are given in Section 5.2.1). We call this data the requester’s characterization. In this section, we introduce our architecture for a recommendation-based Web services discovery system (Figure 1). As we are distributing Web services registries, the entire architecture is structured as a peer-to-peer (P2P) network and made of two layers: the presentation & trading layer and the service providing layer. Due to the dynamic nature of the registries network (at any time a new registry can join or leave the network) we think that a P2P network is well adapted to distribute Web service registries.

3.1. The service providing layer

The service providing layer contains a P2P network of registry peers. Those registries are grouped according to the semantics of their services descriptions. So, each group of registries will host descriptions of services according to a specific business domain (Travel, Restorations). The structure of groups is managed and updated by the presentation & trading peers and the group management component. Groups are created on the basis of a domain ontology containing concepts defining the semantics of business domains. This ontology is stored in the data component (Section 3.2)
of the presentation & trading peers. Since our approach for service discovery is driven by users' characterizations (Section 5.2.1), each registry peer in the service providing layer keeps a copy of requester's characterization who successfully invoked a service discovered in its registry. Those characterizations will be stored in a characterizations repository in the registry peer.

Figure 1. WSs discovery architecture

3.2. The presentation & trading layer

The presentation & trading layer represents the access point for both requesters and providers of services. It offers interfaces according to the following activities:

- help requesters express their queries by assisting them to semantically annotate them.
- help providers semantically annotate elements of their services’ descriptions (e.g. inputs, outputs, interfaces, etc.). The semantic descriptions of those services will be based on YASA4WSDL language [11].

The presentation & trading layer consists of several presentation & trading peers (Figure 2), offering the above mentioned functionalities. When a requester/provider, connects to the architecture, the discovery/publication will be assisted by one of those presentation & trading peers.

In a discovery process, a presentation & trading peer ensures additional functionalities. To limit the search space, it recommends to the service requester the registry upon which he should run his query. This is done in two steps. First, the presentation & trading peer forwards a user’s query to the most adequate group of registries (group selection component). Then, the presentation & trading peer, through the recommender system, recommends to the user one or several registries that have the highest “probability” to satisfy the query. Further details about the group selection mechanism and the registries recommendation are given in Section 4. At this stage, the service requester can execute his query on the recommended registries. After invoking the discovered service, the invocation trace of the service consumer will be updated in his characterization and stored in the characterizations repository of the registry peer. In order to keep the trace of a user up to date, the watchdog component of the presentation & trading peer monitors the user’s interactions. In a publication process, a presentation & trading peer guides the service provider to the registry group that is the most suitable (in terms of business semantics) to the service to be published. This is also ensured through the group selection component.

Figure 2. Structure of a presentation & trading peer

To resume, the different component constituting a presentation & trading peer are:

- **Group selection component.** a semantic matchmaker to find the most suitable group of registries for a service description in a publication process or for a user’s query in a discovery process.
- **Recommender system.** recommends a list of registries to a user in the discovery process. The recommendation is done according to past users characterizations. Details on the used recommendation technique will be presented in Section 4 and 5.
- **Data component.** holds replications of the used ontologies and contains the list of existing registry groups.
- **GUI component.** the interface proposed for service providers/clients that interact with the discovery system.
- **Watchdog component.** monitors and updates traces of a service consumer characterization.

4. Recommender Techniques for WS-Discovery

When developing Web service-based applications, developers may have to select from a large number of already developed Web services those that satisfy these applications’ needs. The current selection techniques have a negative impact on the application development costs. Moreover, even after the selection of a Web service is done, the developer could still have doubts about the appropriateness of this Web service by asking himself if he selected the best Web service. A recommendation based on users ratings could for instance have an important role in the Web services discovery step. A recommender system could reduce the number of Web services to return to a developer’s query. It also can recommend Web services to a developer based on his latest used services or his characterization. In this
section, two important steps in our recommendation-based discovery approach are presented: recommending registry groups and recommending registries from a registry group. These steps are rather generic and independent on the used algorithms and techniques to implement them. More details on our algorithms and techniques are reported in Section 5.

4.1. Recommending registry groups

In our proposed architecture, registries are structured according to a business semantic domain of the services they contain. It will be useful to only execute the discovery query on groups that are semantically similar with the query. This step will be ensured by the group selection component (Section 3.2). So, to recommend the most appropriate registry group to a query, the group selection component executes a semantic matching between the service requester’s semantic annotated query and the list of groups of registries. The most similar groups to the query will be recommended to the requester. However, the recommended groups might, and will probably, contain an important number of registries.

4.2. Recommending a registry from group

After the selection of a registry group for a query, the service requester might be again faced with the large number of services published in the registries of that selected group. Executing a traditional syntactic or semantic matching between the user’s query and all the available services to discover the most adequate service will be time consuming. To deal with that, we propose to limit the user’s search space using a recommender technique that will guide the service requester to the most appropriate registry to his needs. This recommendation will be based on user characterizations. In our proposed architecture, every registry peer keeps a copy of its published services. Recommendations will be given on the basis of the service requester characterization and the list of past requesters characterizations. So registries, which present the more similar past requester’s characterizations to the current service requester characterization will be recommended to the requester as they have the greatest chance to satisfy his query.

5. Our Approach in Use

The used techniques in the recommendation process are independent in the sense that a change in one step’s used technique doesn’t affect the entire recommendation process. In this section, we present how our approach is put in practice by proposing recommender techniques for recommending groups of registries and registries of services in the discovery process.

5.1. Semantic matching for registry groups recommendation

To find a suitable registry group for a service requester’s query, a matching algorithm is used. It evaluates the similarity between a user’s query and registry groups. Queries are written in Yet Another Semantic Annotation for WSDL (YASA4WSDL). Each registry group is also referenced to a concept in an ontology. The used matching algorithm is based on our algorithm introduced in [12]. In this algorithm, the matching result between a query and a registry group is computed from the matching between the concept of the <operation> element in the query and the concept of the registry group. The more similar group (or groups if there is an equality), will be recommended to the service requester.

5.2. User-characterization based technique to recommend registries

Registries of services recommendation is based on past user characterizations and service requester characterization. The recommendation is done by comparing the service requester characterization and a list of user characterizations (stored in the characterizations repositories of the registry peers). To facilitate this, we propose to merge the list of past user characterizations into a single characterization that we call global registry characterization. In this section we introduce the concepts of user characterization and global registry characterization and propose a technique to compute a similarity factor between them.

5.2.1. User-characterization. We define a user characterization as a data structure containing: the user’s areas of interest, his trace and his non-functional requirements.

Areas of interest. The business domains the user is interested in (e.g. travel, restoration, etc.). Those domains can be represented as concepts from a given domain ontology. Despite the fact that we aim to recommend a registry from several ones that belong to the same group (having the same semantic domain), the requester’s areas of interests have an impact on the recommendation. In fact, a user interested in Itineraries and Restoration services might have different needs from users interested in Airlines and Restoration.

Trace. The list of services invoked by a service consumer. In those traces, an invoked service is referenced by the concept of its invoked operation. Trace has an important role in recommendation since service requesters having the same past invocation traces, are mostly interested in the same services.

Non-functional requirements. In addition to the functional requirements (query), a service requester may specify non-functional ones. Those requirements are represented by (concept,value) pairs where concept is an ontology concept of a non-functional property and value is the desired weight for this property.

2. YASA4WSDL represents an extension of the W3C recommendation on semantics for Web services (SAWSDL) and uses two types of ontologies: a Technical Ontology type containing concepts defining semantics of services, their QoS, etc. and a Domain Ontology type containing the concepts defining the semantics of the business domain.
5.2.2. Global registry characterization. To compute the similarity between a user characterization and the past users characterization of a registry (users who already invoked a service from that registry), we merge users characterizations into a single global registry characterization (Figure 3) containing:

Global areas of interests. The list of all domains of interest found in a registry’s past user’s characterizations and the number of time they were identified.

Global trace. The list of all invoked operations from past user’s characterization traces and their number of occurrence.

Global non-functional requirements. A list of (concept, value) pairs where concept is identified from past user’s characterization and value is the average of all values corresponding to this concept.

5.2.3. Computing the recommendation factor. Using the service requester characterization and the global registries characterizations, the recommender system can recommend one or several registries from a group to the requester. This recommendation will be based on a syntactic matching between the requester’s characterization and the list of global characterizations of registries in that group. The recommendation factor \( (rf) \) between a service requester characterization \( (RC) \) and the global registry characterization \( (GRC) \) is computed formulas follows:

\[
rf(GRC, RC) = \alpha \times sf(Interest) + \beta \times sf(Trace) + \delta \times sf(NonFunctionalreq)
\]

Where:

- \( sf(Interest) \), \( sf(Trace) \) and \( sf(NonFunctionalreq) \) respectively represent the similarity factors between the areas of interest, traces and the non-functional requirements of the RC and the GRC.
- \( \alpha \), \( \beta \) and \( \delta \) weight of the similarity factors. Those values represent the importance of each of the three similarity factors vis-à-vis the global recommendation factor.

The registry that has the highest recommendation factor will have the “greatest” chance to satisfy the user’s request. In fact, this factor is calculated on the basis of similarities between the interests, traces and non functional requirements of a RC and a list of GRC. A GRC represents the characterization of past users who used a specific registry. So a high similarity between RC and GRC means a resemblance between the service requester and past user’s of that registry.
(β = 0.8 > α = 0.5 > δ = 0.3)³, the following recommendation factors are obtained:

\[ r_f(GRC_1/RC) = 0.5 \times 0.76 + 0.8 \times 0.23 + 0.3 \times 0.66 = 0.25 \]

\[ r_f(GRC_2/RC) = 0.5 \times 1 + 0.8 \times 0.41 + 0.3 \times 0.8 = 0.35 \]

On the basis of those values, the recommender system will route the query to the registry of GRC_2.

6. Implementation prototype

To put our proposed distributed registry architecture in practice and to test the efficiency of recommendation based discovery, we plan to simulate a Peer-to-Peer (P2P) registries environment on top of Sun Microsystems’s JXTA platform [13]. The JXTA platform is suitable for implementing our distributed registries network test bed. A registry group is then viewed as a peer group where each services registry corresponds to a JXTA peer. To simulate a distributed registry environment (the service providing layer), we can implement several peer groups, i.e., registries groups, hosting several peers, i.e., services registries. Those JXTA peers will create a virtual network on top of the physical network (Figure 5). For the experimentation purpose, several services descriptions will be created and stored in registries on top of those platforms. Each registry serves services from a specific semantic domain.

On the basis of the proposed prototype, we foresee to execute different discovery queries with different weights for the similarity factors (α, β and δ). On the basis of the observed recommendation factors, those weights can be adjusted to ensure a better recommendation. In addition, taking performance indicators (discovery time, search accuracy, etc.) from our experimentations will allow us to compare our recommendation based discovery approach to a matching based approach (i.e. where discovering the right service in a registry group, implicates a matching between the query and all the services in that group). When using our approach, we might obtain less pertinent results in the discovered Web services. However, the discovery time will be better since we don’t have to scan all registries. On the basis of our prototype, it will be interesting to observe the ratio of results pertinence to discovery time using our approach.

7. Conclusion & Future Works

In this paper, we presented our approach for enhancing Web services discovery in a distributed registry environment. We used a recommender system exploiting past users characterizations to route a query to the most appropriate registries. We also proposed an architecture and an implementation prototype for our recommendation-based Web services discovery approach.

On the basis of the proposed prototype, we foresee to execute different discovery queries with different weights for the similarity factors (α, β and δ). On the basis of the observed recommendation factors, those weights can be adjusted to ensure a better recommendation. In addition, taking performance indicators (discovery time, search accuracy, etc.) from our experimentations will allow us to compare our recommendation based discovery approach to a matching based approach (i.e. where discovering the right service in a registry group, implicates a matching between the query and all the services in that group). When using our approach, we might obtain less pertinent results in the discovered Web services. However, the discovery time will be better since we don’t have to scan all registries. On the basis of our prototype, it will be interesting to observe the ratio of results pertinence to discovery time using our approach.

References