Collective Intelligence support for data service exploration and retrieval (discussion paper)

Devis Bianchini, Valeria De Antonellis, Michele Melchiori

Dept. of Information Engineering University of Brescia
Via Branze, 38 - 25123 Brescia (Italy)
{devis.bianchini|valeria.deantonellis|michele.melchiori}@unibs.it

Abstract. Agile design of data intensive web applications can rely on existing and well-tested third parties data services as components providing access methods to valuable data sources. Producers of real data services are more and more publishing their services in repositories according to light semantic profiles with simple tag-based descriptions. In this paper, we discuss techniques for data service exploration and retrieval that consider service co-usage in existing applications, patterns of tags co-usage, and ratings declared by developers who used a data service in their own development experiences.

Keywords: data service, exploratory search, collective intelligence, web-oriented architecture

1 Introduction

Nowadays, we have been observing a growth of research efforts for explorative techniques and methods to deal with the increasing volume and heterogeneity of information made available over the Web [1]. Building web applications, that integrate information available over the web, increasingly requires frameworks to support the discovery of available services providing access to web data sources.

In literature, service recommendations approaches have been proposed to consider service co-occurrence in existing applications as an implicit measure of relatedness of services [2], or suggest services chosen by similar users in a social network based on their usage information [3]. Considering this scenario, in this paper we discuss techniques for data service exploration and retrieval based on selecting and organizing suitable collective intelligence made available in developers communities. In particular, we consider service co-usage in existing applications, patterns of tags co-usage, and ratings declared by developers who used a data service in their own development experiences. Our contribution with respect to existing service recommendation approaches, including our previous work in [4], is that here we adopt an explorative viewpoint, enabling developers to iteratively discover services of interest and progressively increase their knowledge on available services.

The paper is organized as follows. In Section 2 provides some comparison with related work to underline the specific features of our approach. Section 3

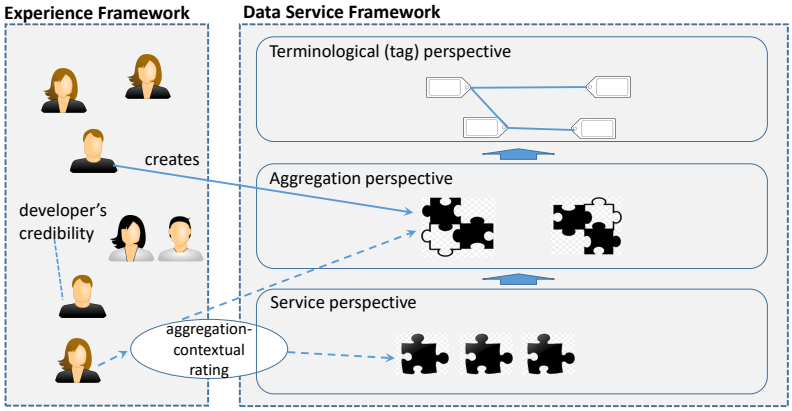

Fig. 1. Overview of the multi-perspective data service model.

describes a multi-perspective data service model. In Section 4 we describe the data service exploration process. Section 5 discusses some preliminary validation. Finally, Section 6 closes the paper with some final remarks.

2 Related work

There are several approaches related to our work. In particular, we mention recent Web service recommendation approaches, that rely on lightweight descriptions of services, such as the ones featuring public repositories (e.g., `ProgrammableWeb` or `Mashape.com`): categories, tags or semantic tags, with the application of advanced IR techniques to enhance topic-based service recommendation [5], natural language API description [6], the number of times a service has been used in the past and the co-occurrence of services in existing applications [2], latent factors (e.g., related to the perceived QoS) that affect users to make service selection, identified mainly using matrix factorization techniques [7]. In this context, approaches like [8] leverage factors to estimate past experiences of data service usage are considered, such as votes/ratings assigned by users to services. These approaches overcome the complexity of traditional state-of-the-art approaches on service discovery (see [9] for a recent survey), that are hampered by the availability of complex, structured service descriptions (e.g., WSDL, WADL and semantic web service formalisms [10]). Compared to these contributions, our aim is to define an explorative approach, exploiting specific collective intelligence, for service recommendation.

Our proposal supports a conversation between the system, used to search for services, and the developer, who is designing a new web application through the selection and aggregation of services.

3 Multi-perspective data service model

We model data services and organize collective intelligence on them by two interconnected frameworks, namely a Data Service Framework and an Experience Framework, that are illustrated in Figure 1 and separately detailed in the next sections. Each framework focuses on specific elements further enriched with cross-framework relationships. Elements considered in such a multi-perspective data service model are the ones included in lightweight descriptions available within most popular repositories (e.g., `Mashape.com`, `ProgrammableWeb.com`).

Service	Service name	Technical features	Tags
s_1	HotWire	$\mathcal{F}_{s_1}^{DataFormat} = \{\texttt{XML,JSON}\}$ $\mathcal{F}_{s_1}^{Protocol} = \{\texttt{RSS, Atom, REST}\}$	{City, Star, Hotel, Travel}
s_2	EasyToBook	$\mathcal{F}_{s_2}^{DataFormat} = \{\texttt{XML}\}$ $\mathcal{F}_{s_2}^{Protocol} = \{\texttt{SOAP}\}$	{City, Hotel, Travel}
s_3	MyAgentDeals	$\mathcal{F}_{s_4}^{DataFormat} = \{\texttt{XML,JSON}\}$ $\mathcal{F}_{s_4}^{Protocol} = \{\texttt{HTTP}\}$	{City, Star, Near, Hotel, Travel}

Fig. 2. Data service descriptions used in the running example.

3.1 Data Service Framework

The Data Service Framework is organized according to three perspectives as shown in Figure 1. Each perspective considers specific elements, namely services, aggregations and terms used to describe them, further described with proper features and relationships between elements.

Service Perspective. This perspective focuses on data services, according to the following definition.

Definition 1. *We define a data service s (hereafter, service) as an operation/method/query to access data of a web source, whose underlying data schema might be unknown to those who use the service. Within the scope of this chapter, we model a service s as* $\langle n_s, \mathcal{F}_s, \{t_s\}\rangle$*, where:* n_s *is the service name;* $\mathcal{F}_s$ *is an array of elements, where each element* $\mathcal{F}_s^X$ *represents the technical feature X (e.g., protocols, data formats, authentication mechanisms) and is modeled as a set of allowed values for that feature (e.g., XML or JSON among data formats);* $\{t_s\}$ *is a set of tags. We denote with* $\mathcal{S}$ *the overall set of available services.*

A *tag* in $\mathcal{T}_s$ may be: (a) a category, taken from a top-down classification imposed within the repository where the data service is stored and advertised[1]; (b) a user tag, that is, a term assigned by developers, aimed at classifying the data service in a folksonomy-like style; (c) a keyword, that is, a recurrent term extracted from the service name and textual description using common IR techniques.
In Figure 2 data services taken from `ProgrammableWeb.com` for the running example are listed.

Aggregation Perspective. Concerning modern application development, to implement a web application starting from available data services, developer has to explore the set of available services, select the most suitable ones, integrate and compose them, in order to deploy the final application. Within the scope of this paper, we focus on the first step, i.e., service exploration for selection purposes, and we talk about service aggregations, instead of web applications that are the final product of the development process. We model aggregations according to the following definition.

Definition 2. *A service aggregation represents a set of services that can be composed to deploy a Web application. An aggregation g is modeled as a triple* $\langle n_g, \mathcal{S}(g), d\rangle$*, where:* n_g *is the aggregation name;* $\mathcal{S}(g) = \{s_1, \ldots, s_n\}$ *is the set of data services used in g;* $d \in \mathcal{D}$ *is the developer who designed the web application by*

[1] See, for instance, the list of `ProgrammableWeb.com` categories at `http://www.programmableweb.com/category-api`.

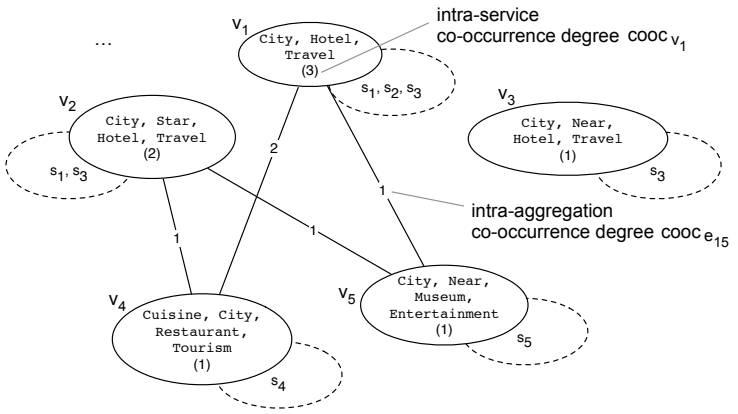

Fig. 3. A portion of the term graph used in the running example; for each node, data services annotated with the node terms are also specified.

composing services in g. We denote with $\mathcal{G}$ the overall set of service aggregations, that is, $g \in \mathcal{G}$, and with $\mathcal{G}(s)$ the set of aggregations where s has been included.

Fictious examples of aggregations are listed in the following.

$g_1 \Rightarrow \langle$ `TravelPlan`, $\mathcal{S}_{g_1} = \{s_1, s_3\}$, d_{g_1} $\rangle$
$g_2 \Rightarrow \langle$ `Stay&Fun`, $\mathcal{S}_{g_2} = \{s_2, s_3\}$, d_{g_2} $\rangle$

Terminological Perspective. The Terminological Perspective collects and organises for supporting service exploration the terminological items used to describe data services. The aim is to provide a *term graph*, described as shown in this section, to start from in order to support data service exploration (see Section 4). We will explain the structure of the graph with the help of the example shown in Figure 3. Formally, the graph is represented as $\langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ is the set of *nodes* and $\mathcal{E}$ is the set of *edges*. In particular, each node $v_i \in \mathcal{V}$ is formally described as $v_i = \langle \mathcal{T}_{v_i}, cooc_{v_i} \rangle$, where $\mathcal{T}_{v_i}$ is a set of tags jointly used to describe a number $cooc_{v_i}$ of data services (*intra-service term co-occurrence degree*). Each edge $e_{ij} \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathbb{N}$ is formally described as $e_{ij} = \langle v_i, v_j, cooc_{e_{ij}} \rangle$, where v_i and v_j are nodes (with corresponding sets of tags $\mathcal{T}_{v_i}$ and $\mathcal{T}_{v_j}$, respectively), such that tags in $\mathcal{T}_{v_i} \cup \mathcal{T}_{v_j}$ have been jointly used within a number $cooc_{e_{ij}}$ of aggregations (*intra-aggregation term co-occurrence degree*). For example in Figure 3, tags in {`City`, `Hotel`, `Travel`} have been used to describe three data services (namely, s_1, s_2 and s_3). The same tags, together with {`Cuisine`,`City`,`Restaurant`,`Tourism`}, are associated with two aggregations (namely, `TravelPlan` and `Stay&Fun`). Tags aim at grouping services that are close from the data viewpoint.

With reference to same figure, $\mathcal{T}_{v_1}$ and $\mathcal{T}_{v_5}$ can be used to suggest developers to aggregate services s_1, s_2 and s_3 with s_5. The framework suggests firstly sets corresponding to an existing aggregation already deployed and tested (let us suppose, $\{s_1, s_5\}$). Other solutions are suggested as well although they do not correspond to existing aggregations (for example $\{s_2, s_5\}$ and $\{s_3, s_5\}$). Therefore, the intra-aggregation term co-occurrence enables developers to explore services that have not been aggregated yet, but can be considered for aggregation because tagged with tags forming patterns used in some existing aggregation. This enables a greater coverage of proposed solutions, at the cost of a lower precision, that however can be acceptable in an explorative process. The term graph can be built and maintained in a fully automatic way.

3.2 Experience Framework

We integrate the Data Service Framework with methods and techniques designed to exploit the experience of developers in selecting data services, thus enabling their ranked recommendation. Such a framework, that in this paper we refer to as *Experience Framework* (EF), has been investigated in our previous work [4, 11]. For the sake of completeness, we report here only few details that are useful to understand the rest of this paper.

The Experience Framework is focused on the set $\mathcal{D}$ of developers. Given a data service $s \in \mathcal{S}$, we denote with $\mu(s, g, d) \in [0, 1]$ the vote assigned to s by a developer $d \in \mathcal{D}$ with reference to the aggregation $g \in \mathcal{G}$ in which s has been used (*aggregation-contextual rating*). Votes are assigned according to the NHLBI 9-point Scoring System[2]. Furthermore, in [11] we included credibility assessment techniques, inspired by the ones defined in [8], with respect to which we introduced the notion of aggregation-contextual rating. In the Experience Framework, a developer is defined according to the following definition.

Definition 3. *A developer represents an actor that is in charge of exploring services and using them to design new aggregations. A developer might also assign aggregation-contextual votes to services. A developer d is modeled as $\langle n_d, c(d) \rangle$, where: (i) n_d is the developer nickname in the considered repository; (ii) $c(d) \in [0, 1]$ is the estimated developer's credibility.*

4 Data service exploration

We envision the service exploration process as a sequence of exploration steps between the developer and the system, used to search for services. The developer starts the exploration by specifying: (i) the set $\mathcal{T}^r$ of terms used within the search request, that provide some initial hints about developer's interests; (ii) the set $\mathcal{F}^r$ of required technical features, for further refining requester's search constraints. Sets $\mathcal{T}^r$ and $\mathcal{F}^r$ compose the service request $\mathcal{R}$, that is completed with the set g^r of services, representing the current composition of the aggregation that is being designed. The framework is equipped with proper wizards (described in [4]), that guide the developer in formulating the request. The system suggests services by computing similarity, filtering and ranking techniques such as the ones introduced in [11] and summarised in the following.

Service similarity evaluation and ranking. A set of similarity metrics have been designed to compare each service description $s \in \mathcal{S}$ extracted from `ProgrammableWeb.com` and the corresponding elements of the request $\mathcal{R}$. The rationale behind these metrics is that the more tags set $\mathcal{T}_s$ of s is similar to $\mathcal{T}^r$, the more technical features $\mathcal{F}_s$ of s are similar to $\mathcal{F}^r$ and the more aggregations where s has been used are similar to the aggregation g^r that is being designed, the more description of service $s \in \mathcal{S}$ fits the request $\mathcal{R}$. The building

[2] `http://www.nhlbi.nih.gov/funding/policies/nine_point_scoring_system_and program_project_review.htm`.

blocks are the *term similarity* ($TermSim()\in[0,1]$), the *technical feature similarity* ($TechSim()\in[0,1]$) and the *aggregation similarity* ($AggSim()\in[0,1]$) metrics that are combined into a overall similarity $Sim(\mathcal{R},s)\in[0,1]$. We denote the set $\mathcal{S}^e\subseteq\mathcal{S}$ of search results such that $\mathcal{S}^e=\{s_i\in\mathcal{S}|Sim(\mathcal{R},s_i)\geq\gamma\}$, where $\gamma\in[0,1]$ is a threshold set by the developer.

A ranking function over the set of services in $\mathcal{S}^e$, denoted with $\rho:\mathcal{S}^e\mapsto[0,1]$, is defined as follows:

$$\rho(s_i)=\frac{1}{N}\sum_{N}[\mu(s_i,g_k,d_i)\cdot c(d_i)\cdot AggSim(g^r,g_k)] \tag{1}$$

where N votes have been assigned to s_i, each vote $\mu(s_i,g_k,d_i)$ is weighted with the credibility $c(d_i)\in[0,1]$ of $d_i\in\mathcal{D}$ as computed in [11] and with the aggregation similarity $AggSim(\cdot)$ of g_k with respect to the aggregation g^r that is being developed. The rationale behind Equation (1) is that a service $s_i\in\mathcal{S}$ is ranked better if it received better votes by more credible designers in the context of more similar aggregations. The system reacts to developer's actions by supporting exploration according to the following three modalities.

Exploration by simple search. The system also looks for nodes $v_i\in\mathcal{V}$ such that $\mathcal{T}^r\subseteq\mathcal{T}_{v_i}$. If multiple nodes are found, for each $v_i\in\mathcal{V}$ the system will suggest to the developer additional terms to be included within the set $\mathcal{T}^r$ considering the set $\mathcal{T}_{v_i}\backslash\mathcal{T}^r$. A suggestion is given for each $v_i\in\mathcal{V}$, ranked in decreasing order with respect to the $cooc_{v_i}$ value. The developer can explore these suggestions in order to consider services alternative to $\mathcal{S}^e$ and to formulate a different request. For instance, with reference to Figure 3, if $\mathcal{T}^r=\{$`City`, `Hotel`, `Travel`$\}$, the system might also suggests as additional terminological item the term $\{$`Star`$\}$ first ($cooc_{v_2}=2$), and $\{$`Near`$\}$ as second option ($cooc_{v_3}=1$). In this way, the developer might realize that hotels can be searched either based on the number of stars or based on the proximity to a given location and he/she might refine the request by choosing one of the two options.

Exploration by proactive completion. The developer selects a subset $\overline{\mathcal{S}^e}\subseteq\mathcal{S}^e$ of services he/she is interested in. The system suggests services that could be used together with services in g^r, by updating the set $\mathcal{S}^e$, according to the *intra-aggregation co-occurrence*. Let's consider the example shown in Figure 4. After performing a search based on $\mathcal{T}^r=\{$`City`, `Hotel`, `Travel`$\}$, thus obtaining $\mathcal{S}^e=\{s_1,s_2,s_3\}$ as results, the developer chooses s_1 to be included in g^r. With reference to Figure 3, s_1 is associated with v_1 and v_2 nodes. Considering node v_1, other nodes connected to v_1 by graph edges are v_4 (associated with s_4, $cooc_{e_{14}}=2$) and v_5 (associated with s_5, $cooc_{e_{15}}=1$). Similarly, considering node v_2, $cooc_{e_{24}}=1$ and $cooc_{e_{25}}=1$. Therefore, the system ranks better the service s_4 than s_5, since $cooc_{e_{14}}+cooc_{e_{24}}>cooc_{e_{15}}+cooc_{e_{25}}$. The developer can accept one of these results. If more than one service is included in g^r, the step of retrieving services is repeated for each service in g^r.

Exploration by hybrid completion. This explorative modality is a combination of proactive completion and simple search. After $\mathcal{S}^e$ has been updated, the developer selects a subset $\overline{\mathcal{S}^e}\subseteq\mathcal{S}^e$ of services he/she is interested in, as well as

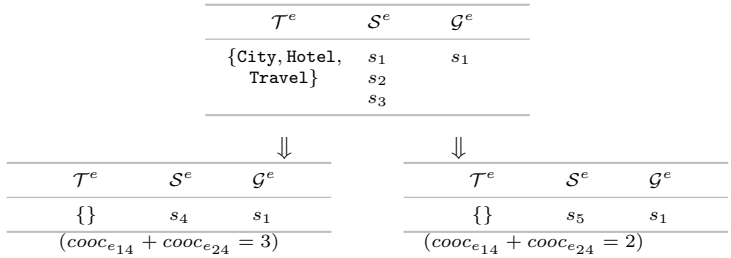

Fig. 4. Example of exploration by completion (see also Figure 3 for the intra-aggregation co-occurrences).

he/she specifies a new set $\mathcal{T}^r$ of terms. The system suggests services that could be used together with services in g^r, by updating the set $\mathcal{S}^e$. In order to obtain this set, a proactive completion step on g^r retrieves some services as explained before.

5 Preliminary validation of the framework

The exploration process described in Section 4 calls for a quantitative evaluation of the scalability of the exploration activities and the execution of experiments with developers, to test the effectiveness of the approach in supporting data service exploration. In this section, we present preliminary experiments on scalability, performed on a dataset of 1317 services extracted from `ProgrammableWeb`.

We initially considered a set of service pairs $\langle s_1, s_2 \rangle$ in the dataset and we manually compared them according to tags, technical feature and aggregation similarity, considering aggregations where they have been used. We run the similarity evaluator to compute $Sim(s_1, s_2)$ by varying the threshold γ from 0.0 to 1.0 and we chose the value of γ that maximized the F-measure.

Then, experiments have been performed ten times using different requests. To this purpose we randomly retrieved aggregations from the repository and we considered as relevant the services included in the aggregations. We then issued the requests using the features of the services in the selected aggregations and we calculated the precision and recall of search results given by our system. The aim of these preliminary experiments is to confirm the advantages for service search brought by our approach, that considers the elements from the multiple perspectives described in the model. For these reasons, we compared our approach against: (a) the keyword-based search facilities made available within the `ProgrammableWeb` repository; (b) a partial implementation of our system, where we excluded the aggregation similarity from $Sim(\mathcal{R}, s)$ computation. Table 1 shows the precision, recall and F-measure results, the standard deviation and the variance of F-measure for the compared systems. As expected, the complete implementation of the system presents the best F-measure value. Although the exploitation of term and technical feature similarity bring significant improvements compared to the basic searching facilities of the `ProgrammableWeb` repository, it is quite evident as the highest enhancement in F-measure value is due to the integration also of aggregation similarity, that mainly relies on service co-occurrence.

System	F-measure	Precision	Recall	Std deviation	Variance
`ProgrammableWeb`	0,0414	0,03105	0,0621	0,0363	0,0013
WISeR (no $AggSim$)	0,4247	0,5880	0,3324	0,3562	0,1269
WISeR	0,5993	0,7451	0,5012	0,2159	0,0466

Table 1. Results of the experimental evaluation.

6 Concluding remarks

In this paper, we proposed an approach for data service explorative search, based on a multi-perspective model for data services and specific collective intelligence on them. The approach includes proactive search facilities and enables developers to iteratively increase their knowledge on available web data services. Future work will be devoted to the study of techniques for including latent factors (e.g., related to the perceived QoS) in the exploration process. Further open research includes the definition of the visualization interface to further increase the exploration experience of developers.

References

1. S. Idreos, O. Papaemmanouil, S. Chaudhuri, Overview of data exploration techniques, in: ACM Conference on Management of Data (SIGMOD), 2015.
2. W. Gao, L. Chen, J. Wu, A. Bouguettaya, Joint Modeling Users, Services, Mashups and Topics for Service Recommendation, in: Proc. of 23rd International Conference on Web Services (ICWS 2016), 2016.
3. R. Hu, J. Liu, Y. Wen, Y. Mao, USER: A usage-based service recommendation approach, in: Proc. of 23rd International Conference on Web Services (ICWS 2016), 2016.
4. D. Bianchini, V. De Antonellis, M. Melchiori, A Multi-perspective Framework for Web API Search in Enterprise Mashup Design (Best Paper), in: Proc. of 25th Int. Conference on Advanced Information Systems Engineering (CAiSE), Vol. LNCS 7908, 2013, pp. 353–368.
5. B. Cao, X. Liu, B. Li, J. Liu, M. Tang, T. Zhang, Mashup Service Clustering Based on an Integration of Service Content and Network via Exploiting a Two-level Topic Model, in: Proc. of 23rd International Conference on Web Services (ICWS 2016), 2016.
6. W. Xiong, Z. Wu, B. Li, Q. Gu, L. Yuan, B. Hang, Inferring service recommendation from natural language api description, in: Proc. of 23rd International Conference on Web Services (ICWS 2016), 2016.
7. X. Liu, I. Fulia, Incorporating User, Topic, and Service Related Latent Factors into Web Service Recommendation, in: Proc. of IEEE International Conference on Web Services (ICWS 2015), 2015, pp. 185–192.
8. Z. Malik, A. Bouguettaya, RATEWeb: Reputation Assessment for Trust Establishment among Web Services, VLBD Journal 18 (2009) 885–911.
9. S. Pakari, E. Kheirkhah, M. Jalali, Web Service Discovery Methods and Techniques: A Review, Int. Journal of Computer Science, Engineering and Information Technology 4 (1) (2014) 1–14.
10. H. Wang, N. Gibbins, T. Payne, A. Patelli, Y. Wang, A survey of Semantic Web Services formalisms, Concurrency and Computation Practice and Experience.
11. D. Bianchini, V. De Antonellis, M. Melchiori, Capitalizing the Designers' Experience for Improving Web API Selection, in: On the Move to Meaningful Internet Systems: OTM 2014 Conferences, Vol. LNCS 8841, 2014, pp. 364–381.