Mining Social Web Service repositories for social relationships to aid service discovery

Alejandro Corbellini, Daniela Godoy, Cristian Mateos, Alejandro Zunino, Ignacio Lizarralde
ISISTAN Research Institute, CONICET-UNCPBA
Campus Universitario, Tandil (B7001BBO), Buenos Aires, Argentina
E-Mail: alejandro.corbellini@isistan.unicen.edu.ar

***Abstract*—The Service Oriented Computing (SOC) paradigm promotes building new applications by discovering and then invoking *services*, i.e., software components accessible through the Internet. Discovering services means inspecting registries where textual descriptions of services functional capabilities are stored. To automate this, existing approaches index descriptions and associate users' queries to relevant services. However, the massive adoption of Web-exposed API development practices, specially in large service ecosystems such as the IoT, is leading to ever-growing registries which challenge the accuracy and speed of such approaches. The recent notion of Social Web Services (SWS), where registries not only store service information but also social-like relationships between users and services opens the door to new discovery schemes. We investigate an approach to discover SWSs that operates on graphs with user-service relationships and employs lightweight topological metrics to assess service similarity. Then, "socially" similar services, which are determined exploiting explicit relationships and mining implicit relationships in the graph, are clustered via exemplar-based clustering to ultimately aid discovery. Experiments performed with the ProgrammableWeb.com registry, which is at present the largest SWS repository with over 15k services and 140k user-service relationships, show that pure topology-based clustering may represent a promising complement to content-based approaches, which in fact are more time-consuming due to text processing operations.**

***Keywords*-Service discovery; Social Web Service; Social recommender systems; Exemplar-based clustering**

I. Introduction

The Service Oriented Computing (SOC) paradigm promotes creating new applications by invoking remote components called *services*, which are publicized via repositories accessible through the Internet known as *registries*. Furthermore, as social networks and Cloud Computing became popular, value-added applications –known as "Web mashups"– that combine several Web Services from different sources emerged [6], many of which are in turn exposed as new services.

Anatomically, a service is composed by its implementation and its description, this latter stored in one or more registries, which must later be discovered by consumers. The problem of discovering services has led to several efforts both for SOAP [3], [18] and REST services [13], [19], exploiting traditional techniques from Information Retrieval (IR) to match keywords (queries) against an indexed collection of service descriptions (documents). Two common indexing schemes are non-partitioned and partitioned. The former locates documents in the same search space (e.g. using the Vector Space Model technique) and finds the nearest neighbors to an input query. Partitioned approaches, alternatively, divide the search space into groups of highly-related documents, usually via clustering techniques [19], [8], and queries are solved by returning closely-associated services. In general, the partitioned scheme has better search time because it reduces the amount of documents to brute-force search.

Despite these advances, application developers still struggle to find relevant services [9], in part, due to the ever-growing volume of services that pushes the limits of current approaches accuracy. This growth stems from the popularity of Web-accessible APIs and mashup development [6], the advent of user-friendly online API repositories such as Mashape.com (http://www.mashape.com), Mashery.com (http://stackshare.io/mashery) and ProgrammableWeb.com (http://www.programmableweb.com/), the existence of Cloud models like IaaS/SaaS/PaaS where any computing resource is exposed as service, and new (and very large) service ecosystems such as the IoT. Here, clustering services based only on textual content [19] might a) scale poorly, as compute-intensive text-processing algorithms must be run over many service descriptions when creating or updating clusters, and b) lead to low-quality clusters since the high number of published services increases the likelihood of poor textual service descriptions (e.g. those with ambiguous terms, unknown acronyms, etc.).

Social Web Services (SWS) [9] brings the metaphor of social networks to service ecosystems by evolving service discovery from processing service descriptions to exploiting social-like networks with rich relationships between users and services. This opens the door to clustering schemes for service discovery where services are clustered based on topological similarity, under the assumption that e.g. services followed by users showing similar interests should be in the same cluster. In general, it is known that the cost of computing topological similarity in large networks (graphs) is negligible compared to that of content-based ones [11].

Then, we empirically assess the effectiveness of topological-based clustering for SWS discovery. To determine service similarity, we utilize local similarity indices –known to scale well for big graphs [4]– that assess the commonalities between two service neighborhoods in a graph. Moreover, we evaluate two relevant exemplar-based clustering algorithms. This type of algorithms are well-suited for generating partitions based on pairwise similarities as those measuring the resemblance of the social context of two given services.

Next we explain the problem of discovering traditional –i.e., non-social– services, and discuss works on SWS discovery and selection. Section III describes the considered approach to socially cluster services. An empirical evaluation with the ProgrammableWeb.com registry is reported in Section IV. Section V discusses the findings and prospective future works.

II. Background and Related Research

Although service registries are very good at retrieving services through well-defined keywords, users often use ambiguous and short natural language sentences, which leads to term mismatch, or the vocabulary problem [5]. Then, most works in the area focus on mitigating this problem. In [3] queries are enhanced with terms extracted from the client application. Another approach is enriching textual service descriptions with more meaningful/relevant terms [18], [2].

In software engineering terms, services are isolated components that may be combined to produce more complex applications once this isolated services are retrieved from a registry [6]. To evolve this idea, SWSs [9], [10] mix Social Web concepts (notably social networks) and service-oriented engineering. An SWS is a service that lives in an interlinked network of services and users, and where users interact, share, collaborate, and make recommendations both for accessing services and building compositions.

Indeed, the network structure and its services/users, and the information of service interactions and users' activity can be exploited in a variety of ways, but this is an incipient idea yet. In [21] a framework to build communities of services is proposed, where services belonging to the same community are those offering similar functionality. Likewise, LinkedWS [10] models competition, collaboration, and substitution interactions as part of the process of building a "network of contacts" for services, and describes a service discovery model that sits on top of the networks built.

Another line of work involves capitalizing Recommender Systems (RS) for individual service discovery. This is the cornerstone of works for trustworthy discovery of SWSs (e.g., [17]). Other works, such as WSRec [22], use collaborative filtering techniques to recommend services based on service QoS attributes.

All in all, we mainly aim at investigating and improving SWS discovery. Unlike works like [10], [21], which focus on the process of modelling and building social networks for existing services, we pursue an RS-inspired approach that can be applied to service registries already deployed and exhibiting social information and/or structure, such as ProgrammableWeb.com and Mashape.com. A key design driver of this approach is to utilize simple metrics to compute similarity among services to keep the search process as scalable as possible.

III. Topology-based Clustering of SWSs

Current service registries provide rich information for easing service discovery that goes beyond the usual syntactic structure of a service, as a consequence of the users' collective task of publishing, linking and annotating services [9]. User-generated text and tags provide meaningful content-related information regarding services [3]. Service repositories such as ProgrammableWeb.com or Mashape.com allow users to follow services according to their interests, establishing a certain relationship between users and the published services and, through other users in the community, also with the remaining services. Since a follower relationship denotes an interest of the user in a given service, it can be assumed that services followed by the same people respond to the same interests. From the perspective of an application consuming services, this means that two services might offer the same or similar functionality and hence they are interchangeable. Consequently, socially alike services respond to the same information need in terms of a classical information retrieval system (i.e., queries) or can be grouped together via a clustering process. Figure 1 depicts a canonical example of socially-related services in a repository. Based on the graph, services WS_1 and WS_2 could be clustered together, whereas WS_3 belongs to another cluster.

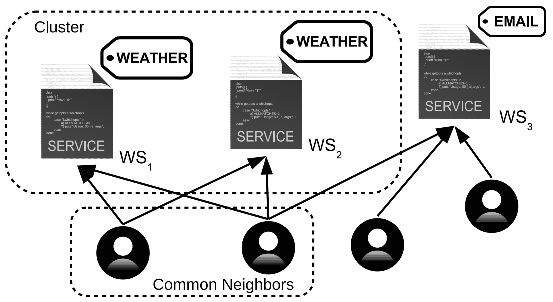

Fig. 1: Example of socially-related services

Particularly, to exploit the social context of services for improving clustering, we first measure social resemblance of services using topological similarity indices (Section III-A). Second, we use exemplar-based clustering methods based on the matrices resulting from assessing pairwise similarities between services (Section III-B). This lead to service partitions that can be then leveraged for service discovery under the clustering hypothesis.

A. Social similarity of services

Grouping services by similarity implies to determine their commonality in the associated graph. The structure of the graph itself can be taken as source for measuring similarity between two nodes so that they are deemed similar if they are structurally close in the graph. To this end, available metrics are local similarity indices, which consider the neighborhood of the two nodes of interest, while global indices involve the whole graph. We study local indices since they are easier to compute and scale well for large graphs [4].

Table I summarizes the local similarity indices used in this work. For a node x in a graph G, let $\Gamma(x)$ denote the set of neighbors of x in G and k_x its degree. We employ neighbor-based metrics, which assume that two vertices x and y are more likely to form a link in the future if $\Gamma(x)$ and $\Gamma(y)$ have

Local Similarity Index	Formula
Common Neighbors (CN)	$s_{xy}^{CN} = \lvert\Gamma(x) \cap \Gamma(y)\rvert$
Salton Index	$s_{xy}^{Salton} = \frac{\lvert\Gamma(x)\cap\Gamma(y)\rvert}{\sqrt{k_x \times k_y}}$
Jaccard Index	$s_{xy}^{Jaccard} = \frac{\lvert\Gamma(x)\cap\Gamma(y)\rvert}{\lvert\Gamma(x)\cup\Gamma(y)\rvert}$
Sørensen Index	$s_{xy}^{Sørensen} = \frac{2\lvert\Gamma(x)\cap\Gamma(y)\rvert}{k_x+k_y}$
Hub Depressed Index (HDI)	$s_{xy}^{HDI} = \frac{\lvert\Gamma(x)\cap\Gamma(y)\rvert}{\max\{k_x,k_y\}}$
Adamic-Adar Index (AA)	$s_{xy}^{AA} = \sum_{z\in\Gamma(x)\cap\Gamma(y)} \frac{1}{\log k_z}$

TABLE I: Definitions of local similarity indices employed

a large overlap. The simplest one is Common Neighbors (CN), which measures the overlap of the ego-centric networks of two nodes, or services in our case. CN represents the intersection between the neighborhood of each service as illustrated in Figure 1. Salton [15] and Jaccard [7] indices normalize the size of common neighbors. Sørensen index [16] penalizes it with the sum of the neighborhoods sizes. Hub Depressed Index (HDI) [12] measures the number of shared neighbors and penalize it by the maximum neighborhood size. Lastly, the Adamic-Adar index (AA) [1] refines the simple counting of common neighbors by assigning the less-connected neighbors higher weights.

B. *Exemplar-based service clustering*

In this type of clustering, the pairwise similarity of services is used to form clusters of services. A classical clustering algorithm is k-Means, a prototype-based algorithm that builds a user-specified number of clusters k, each represented by a centroid. Indeed, k-Means has been used to cluster services on the grounds of its simplicity and fastness [19], [20]. For instance, in [19] services are clustered using both service descriptions and tags, adding a tag recommending service that employs tag co-occurrence, tag mining, and semantic relevance measurement for mitigating the problem of uneven tag distribution and noisy tags. In [20] k-Means is used for service recommendation in automatic mashup creation.

A representative exemplar-based algorithm is k-Medoids. This algorithm returns the k medoids as exemplars (i.e., clusters), it works with any similarity metric, and only the $n \times n$ similarity matrix is required, where n is the number of services. The major drawback of k-Medoids is the need to establish a desired number of clusters, which is difficult in today's service registries where users are regularly publishing new services. Therefore, we also consider Affinity Propagation (AP), which automatically determines the number of clusters.

Unlike k-Medoids, AP initially considers all data points as potential exemplars, and therefore is deterministic. Based on the input similarity matrix, AP exchanges real-valued messages ("affinities") along the edges of the network, regarding each data point as a node, until a good set of exemplars and the clusters gradually emerge. AP sends two types of messages between data points i to j, namely responsibilities and availabilities. A responsibility message reflects how well-suited j is to serve as the exemplar for i. An availability message tells how appropriate it is for i to choose j as its exemplar. Interestingly, AP can be applied with non-metric similarities.

IV. Experimental Evaluation

We crawled the services from the ProgrammableWeb.com registry, which resulted in 15,076 services from diverse domains. Each service has a textual description, one main category and a list of secondary categories, both specified by service providers. There are 474 unique service categories, 142,065 links representing users following services, and 80,464 registered users. The experimental data is available upon request.

Service main categories were used as ground truth to validate the clustering results. A small proportion of services have no followers (9.03%), a 16.96% of services has only 1 follower, and a 1.16% of services have more than 100 followers. As in most social networks, a reduced number of services concentrate a high number of followers: in fact, the 6 most popular services –*Facebook*, *Google Maps*, *Twitter*, *YouTube*, *AccuWeather* and *LinkedIn*– are the only ones having more than 1,000 followers.

With respect to quantifying the quality of our clustering solutions, we relied on external quality metrics, which measure how well the clustering results match some prior knowledge (main categories in our case) about the data. Although some categories are general, they are assigned manually by the service provider, and thus, they represent a hint on what the exposed functionality is w.r.t. other Web Services.

Given the knowledge of the ground truth category assignments, a conditional entropy analysis helps to define external validation metrics. A perfect clustering solution contains clusters with examples from a single category, while an entropy close to 1.0 implies that the cluster contains a uniform mixture of categories. However, entropy can not be used in isolation as the entropy is maximized when each cluster contains exactly one single instance.

Then, the following two objectives for any cluster assignment are desirable [14]: homogeneity h (whether each cluster contains only members of a single class) and completeness c (whether all members of a given class are assigned to the same cluster). Both metrics are bounded between 0 and 1, with higher scores being better. If C is the ground truth class assignment and K the clustering, $h = 1 - \frac{H(C|K)}{H(C)}$ and $c = 1 - \frac{H(K|C)}{H(K)}$. $H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n_k}\right)$ is the entropy of the classes given the cluster assignments and $H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n} \cdot \log\left(\frac{n_c}{n}\right)$ is the entropy of the classes. Here, n is the total number of instances, n_c and n_k are the number of instances of class c and cluster k respectively, and $n_{c,k}$ counts the instances of c assigned to k. $H(K|C)$ and $H(K)$ are defined analogously. Finally, we also consider V-measure, computed as the harmonic mean of homogeneity and completeness: $v = 2 \cdot \frac{h \cdot c}{h+c}$.

	Content-based	Social-based					
	Cosine	CN	AA	Jacard	Salton	Sørensen	HDI
# clusters	1,540	802	1,469	2,509	2,604	2,291	2,012
Entropy	2.033	3.889	3.065	2.510	2.450	2.693	2.632
Homogeneity	0.686	0.399	0.527	0.612	0.621	0.584	0.593
Completeness	0.444	0.298	0.346	0.377	0.376	0.355	0.365
V-measure	0.539	0.341	0.418	0.467	0.468	0.442	0.452

TABLE II: Affinity propagation results

	Content-based	Social-based					
	Cosine	CN	AA	Jaccard	Salton	Sørensen	HDI
# clusters	1,500	1,043	1,098	205	218	176	203
Entropy	2.247	4.139	4.064	5.391	5.354	5.207	5.435
Homogeneity	0.653	0.361	0.372	0.167	0.173	0.196	0.160
Completeness	0.433	0.334	0.337	0.275	0.277	0.280	0.276
V-measure	0.521	0.347	0.354	0.208	0.213	0.230	0.203

TABLE III: k-Medoids with 1,500 clusters

Traditional partitioned discovery schemes are often only based on indexing and clustering of service descriptions. Then, we used the description of services for comparing this social clustering approach against the standard content-based clustering approach. A common text processing pipeline was applied to service descriptions: stop-words removal, stemming and *tf-idf* weighting.

Tables II, III and IV show the results from running the AP and k-Medoids algorithms using the considered metrics. Given the number of categories in the dataset (ground-truth), experiments were run for $k = 470$ (the classes in the dataset), and $k = 1500$, the number of clusters computed by the AP algorithm.

In general, the cosine similarity metric between the *tf-idf* vectors of the services' descriptions performed better than most of the social-based metrics for both algorithms. Social-based metrics using the k-Medoids algorithm yielded low values on most metrics, even failing to find the requested amount of clusters. This might be caused by k-Medoids inability to handle the low scores of social-based metrics, resulting in some medoids being discarded.

	Content-based	Social-based					
	Cosine	CN	AA	Jaccard	Salton	Sørensen	HDI
# clusters	470	422	423	1	1	2	1
Entropy	3.133	4.911	4.807	6.473	6.473	6.472	6.473
Homogeneity	0.516	0.241	0.257	0	0	0	0
Completeness	0.403	0.293	0.290	1	1	0.352	1
V-measure	0.453	0.265	0.273	0	0	0	0

TABLE IV: k-Medoids with 470 clusters

However, AP performed very well for both social and content-based metrics. The V-measure scores for HDI, Salton and Jaccard, which are 15%, 16% and 19% lower than the scores from the cosine metric, respectively, which is a promising result given the inherent computational cost difference in social and content-based approaches. The Salton metric also provided the lowest entropy score of the social-based metrics, although it had 70% more clusters than for the content-based approach. The AA metric provides a good balance between clustering performance and number of clusters, resulting in 1,469 clusters and a V-measure of 0.418, i.e., 28% lower than the cosine metric.

V. Conclusions

Although purely topology-based clustering achieves partitions of slightly inferior quality than a content-based approach, they could be a suitable alternative/complement to measure service similarity in large, fast-growing service registries. Topology-based metrics explore vertices that are directly connected to the target vertices, which means that only those users that are currently following the target services are considered for metric computation. Moreover, updating the scores of topology-based metrics can be performed on-the-fly after a change is detected on the social graph, for example, when a user follows/stops following an API. This is useful in fast-evolving registries. For example, ProgrammableWeb.com and Mashape.com show a clear exponential growth from 2005-2013 and 2010-2015, respectively.

We will take advantage of other relationships present in the social network to identify service similarity and perform clustering to increase accuracy. Particularly, ProgrammableWeb.com allows users to publish not only individual APIs but also compositions of individual services (mashups), which involves service-service links in the graph. Roughly, at present ProgrammableWeb.com contains around 7,500 mashups. We hypothesize that these links may also contribute to determine similarity of (simple) individual services.

We will also improve the ground truth. At present, there are some quite general categories for services (e.g. "Tools") in the dataset. Another problem is that the metrics used do not consider "overlapping" service similarity, since each service is assigned only one (main) category. To mitigate these problems, we will exploit service secondary categories. Lastly, we will exploit the clustering hypothesis plus our results in the context of a recommender system for SWS, i.e., evaluating the impact of our approach w.r.t. performance metrics from the IR area, such as Precision, Recall, NDCG and F1 score. To this end, we will build an extended dataset with keyword-based queries associated to *functionally relevant* services –as judged by human discoverers–.

Acknowledgment

This work has been partially funded by CONICET (Argentina) under Grant PIP No. 112-201201-00185 and ANPCyT (Argentina) under Grant PICT 2013 No. 0464.

References

[1] L. A. Adamic and E. Adar. Friends and neighbors on the Web. *Social Networks*, 25(3):211–230, 2003.

[2] Z. Azmeh, J.-R. Falleri, M. Huchard, and C. Tibermacine. Automatic web service tagging using machine learning and wordnet synsets. In *International Conference on Web Information Systems and Technologies*, pages 46–59. Springer, 2010.

[3] M. Crasso, A. Zunino, and M. Campo. Combining query-by-example and query expansion for simplifying web service discovery. *Information Systems Frontiers*, 13(3):407–428, 2011.

[4] F. Fouss, M. Saerens, and M. Shimbo. *Algorithms and Models for Network Data and Link Analysis*. Cambridge University Press, 2016. Chapter 2.

[5] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[6] M. Garriga, C. Mateos, A. Flores, A. Cechich, and A. Zunino. Restful service composition at a glance: a survey. *Journal of Network and Computer Applications*, 60:32–53, 2016.

[7] P. Jaccard. Etude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Societe Vaudoise des Science Naturelles*, 37(142):547–579, 1901.

[8] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information storage and retrieval*, 7(5):217–240, 1971.

[9] Z. Maamar, P. Santos, L. Wives, Y. Badr, N. Faci, and J. P. M. de Oliveira. Using social networks for Web services discovery. *IEEE internet computing*, 15(4):48–54, 2011.

[10] Z. Maamar, L. K. Wives, Y. Badr, S. Elnaffar, K. Boukadi, and N. Faci. Linkedws: A novel web services discovery model based on the metaphor of 'social networks'. *Simulation Modelling Practice and Theory*, 19(1):121–132, 2011.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[12] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002.

[13] J. M. Rodriguez, A. Zunino, C. Mateos, F. O. Segura, and E. Rodriguez. Improving rest service discovery with unsupervised learning techniques. In *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 97–104, 2015.

[14] A. Rosenberg and J. Hirschberg. V-Measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

[15] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[16] T. Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Kongelige Danske Videnskabernes Selskab*, 5(4):1–34, 1948.

[17] S. Wang, L. Huang, C.-H. Hsu, and F. Yang. Collaboration reputation for trustworthy web service selection in social networks. *Journal of Computer and System Sciences*, 82(1):130–143, 2016.

[18] C. Wu. WSDL term tokenization methods for ir-style Web Services discovery. *Science of Computer Programming*, 77(3):355–374, March 2012.

[19] J. Wu, L. Chen, Z. Zheng, M. R. Lyu, and Z. Wu. Clustering Web services to facilitate service discovery. *Knowledge and Information Systems*, 38(1):207–229, 2014.

[20] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu. Category-aware API clustering and distributed recommendation for automatic mashup creation. *IEEE Transactions on Services Computing*, 8(5):674–687, 2015.

[21] H. Yahyaoui, Z. Maamar, E. Lim, and P. Thiran. Towards a community-based, social network-driven framework for web services management. *Future Generation Computer Systems*, 29(6):1363–1377, 2013.

[22] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Qos-aware web service recommendation by collaborative filtering. *IEEE Transactions on Services Computing*, 4(2):140–152, 2011.