From a Web Services Catalog to a Linked Ecosystem of Services

Fatma Slaimi[1], Sana Sellami[1], Omar Boucelma[1]
[1]Aix Marseille Université, CNRS, ENSAM, Université de Toulon,
LSIS UMR 7296,13397, Marseille, France
{fatma.slaimi,sana.sellami,omar.boucelma}@univ-amu.fr

Abstract—In this paper, we present a Linked ecosystem of Web services where both Web services, mashups and users are represented as a multigraph structure. For illustration and experimental purposes, a graph has been constructed, in gathering different data from ProgrammableWeb. The graph is stored in a Neo4J graph database and serves as a repository for real collection of data for achieving services/mashups discovery, selection and recommendation.

***Keywords*:** Web Services; Discovery, Recommendation; Linked Data; Graph Databases; Neo4J.

1 Introduction

The advent of service oriented approaches has led to plethora of available Web services (or APIs). However, and besides the availability of these services through directories/catalogues such as ProgrammableWeb[1] (PWeb) Web services' gates still remain non-well-structured in order to ease the discovery and recommendation processes. In addition, all these services/mashups are a bit scattered on the Web and there are no materialized links among them that could facilitate their discovery. Therefore, we need an explicit WSs representation in materializing different links accordingly with different semantics.

Indeed, with the growth of social networks and the emersion of social Web, we do not need to take leverage only from the user's previous usages, user's profile, or to consider him/her as an independent, isolated entity, but we may also consider other types of relationships as inputs, e.g. follow, confidence, etc. These relationships generate graphs of interconnected close or "similar" users, among which are those having common interests. In PWeb, users' relations are represented by track relationships (Fig. 1).

[1] https://www.programmableweb.com/

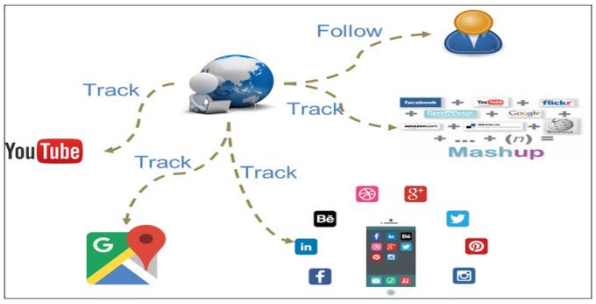

Fig. 1 Track relationships on ProgrammableWeb

Hence, the goal of this paper is to describe the design of a PWeb-like ecosystem model as a multigraph where intra-services/mashups, intra-users and inter services/users and mashups links are exhibited. The idea is to make leverage of the available information on objects (services, mashups and users) in order to discover and to recommend services of interest to a user, accordingly on his/her profile. From the implementation standpoint, the graph is stored in a graph database and a prototype has been implemented on top of a Neo4j graph database. For validation, an experimental evaluation has been performed with a real dataset[2].

The remainder of this paper is organized as follows. In section 2, we describe some related work. In section 3, we present the PWeb heterogeneous multigraph. Section 4 describes the graph exploration to recommend services/mashups. Section 5 details the implementation features and the results of our approach. Section 6 concludes the paper and presents some future work directions.

2 Related Work

Several research works have been proposed to model the Web services Network as a graph in order to discover or recommend APIs or mashups [3], [6], [9]. As described in [6], authors propose a three-level graph model for visualizing the Web service ecosystem: (1) An API graph that connects services when they are used in the same mashup; (2) A domain graph that provides information about services that are more likely to be connected to produce innovative applications; and (3) A tag graph which connects two tags if there exist two APIs which are labeled with those tags and appear in the same mashup. In addition, the authors in [3] propose a graph based recommendation approach to assign tags to unlabeled APIs by exploiting both graph structure information and semantic similarity.

Moreover, the convergence towards a Web of linked data[3] leads to a novel services' representation, known as "services linked by their URIs" or linked services allowing the definition of interconnected services based on their functionalities. In [2], authors propose a framework for defining a linked view over multiple repositories and for

[2] http://www.lsis.org/sellamis/Projects.html#WeS-ReG
[3] http://linkeddata.org/

searching their content. The objective is to publish repository contents and identify semantic links across them in order to exploit complementary API descriptions. The motivation of this framework are threefold: (i) the overlapping of the storage of APIs, (ii) the same API is registered multiple times within different repositories, (iii) similarities between APIs and mashups across different repositories cannot be exploited to enrich search results. Authors also identified a link (denoted simAs link) between mashups of resources which is calculated based on the comparison of their terminological items. In [5] an approach for providing new services based on service composition is proposed. Authors define the term of composite web services for a web service that does not provide requested service but delegates parts of the execution to other web services and receives the results from them to perform the whole service. However, all these models do not consider the users' interactions.

Indeed, social networks change the nature of "useful" metadata in the discovery and recommendation processes [4]. Consequently, we do not need to profit only from the user's previous uses, user's profile, or to consider him/her as an independent, isolated entity, but we can take as input other types of relationships, e.g. follow, confidence, etc. These relationships are used to generate graphs of interconnected users, which help in determining closest users or those having common interests.

The works of Deng.et al. [7] [8] are based on the analysis of social networks' contents and users' relationships. Authors propose a Web service recommendation system based on trust relationships modeled by a graph and established either i) explicitly when a user specifies his/her list of trustful connections, from the beginning, or ii) implicitly when the same QoS evaluation is given by two different users. Note that this approach exploits the preferences similarities among users involved in the network, but still suffers from the lack of data especially for new users, e.g. preferences, previous uses, etc. In [12], authors proposed a discovery/composition model (called LinkedWS) based on social networks. This model is able to detect the interactions among Web services. The relations used by LinkedWS are: Recommendation/Partners, Recommendation/Robustness and Collaboration. The composition process implements both Recommendation/Partners and Collaboration relationships. LinkedWS can be updated by adding new nodes/relations or by modifying them.

Within this context, we propose to model the programmable Web ecosystem as a heterogeneous multi-graph. The graph considers the interest relationships between system's users and published Web services/mashups. The goal is to streamline the search for a service and also to allow the detection of users' neighborhoods. In addition, the use of such graph representation makes the system able to propose a service to a requester and to personalize a recommendation for a new user.

3 Heterogeneous Multigraph of the Programmable Web

In this section, we describe a the PWeb "social network". This network is an extension of a previous work [11] on Linked APIs graph database while considering more nodes/relationships.

3.1 Multigraph Overview

The network is modeled as a heterogeneous multigraph by means of *users/services* and *mashups relationships*. This multigraph is depicted in (Fig. 2) and consists on four levels according to the different nodes: Categories level, Services level, Mashups and Users levels. A service may belong to one or several categories (membership relationship); A user can follow services/mashups used by other users even if he/she is not directly connected to these users.

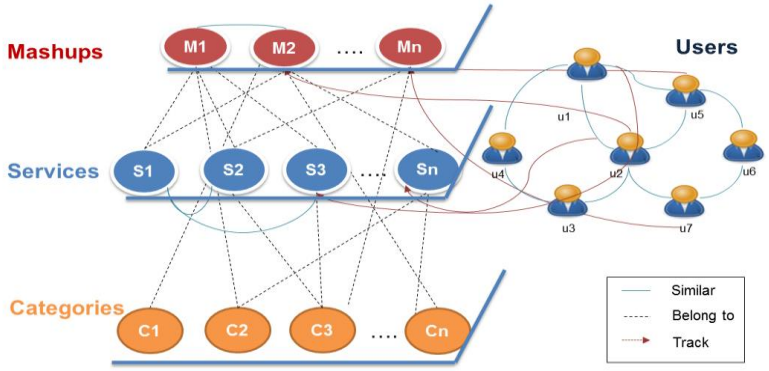

Fig. 2. Heterogeneous multigraph of the Programmable Web

The multigraph exhibits four different types of relationships among both users and services/mashups. Relationships are the edges of the multigraph. First, knowing that each service/mashup may belong to a category, e.g. twitter service belongs to social category, we came up a *Belongs_To link*. Next, services (resp. mashups) need to be related to each other. Based on the services (resp. mashups) properties, we proposed the *similar link*. We identify the *Belongs_To_mashup link* between services and mashups when a service is part of a mashup. Finally, each user can follow services/mashups and other users. We define then a *track link* between these entities.

3.2 Similarity Relationships

In this section, we describe the different relationships that may exist between users (resp. services/mashups) in order to construct the multigraph. These relationships are defined on the basis of similarities among entities. Similarities are based on follow/ track relationships and properties similarities. Track relationships are expressed by means of PWeb tracks. A user can track (follow): web services, mashups, or other users.

Services Relationships (*Similar link***).** Currently, several Web services (e.g. google Maps, big Maps) that are exposed on existing platforms such as *ProgrammableWeb* are functionally equivalent. A relationship can be defined between each pair of functionally similar services. This similarity is determined by comparing services' description items,

i.e., category, name, description and tags. The similarity between two services (formula.1) is calculated using the average of Information Retrieval (IR) metrics [14] as described in the Table 1.

Table 1 IR similarity measures

Service Properties	**Measures**	**Formula**
Category	cosine	$sim_{Cat}(C1,C2)=\frac{\lvert VC1.VC2\rvert}{\lvert\lvert VC1\rvert\rvert.\lvert\lvert VC2\rvert\rvert}$ Where C1 and C2 are the categories of services. VC1 and C2 are the vectors of words of C1 and resp. C2.
Name	Ngram	$sim_{Name}(n1,n2)=\frac{C(w_{n-1}w_n)}{C(w_{n-1})}$ Where W is the set of words in n1 and n2, $C(w_{n-1})$ represents the count of bigrams starting with w_{n-1}
Description	cosine	$sim_{Desc}=\frac{\lvert VD1.VD2\rvert}{\lvert\lvert VD1\rvert\rvert.\lvert\lvert VD2\rvert\rvert}$
Tag	Jaccard	$sim_{Tag}(T1,T2)=\frac{\lvert T1\cap T2\rvert}{\lvert T1\cup T2\rvert}$

$$Sim\ (S_i,S_j)= \text{Average}\ (sim_{Cat}, sim_{Name}, sim_{Desc}, sim_{Tag}) \quad (1)$$

Mashups Relationships (*Similar link***).** We establish a link between mashups if they share one or more services. For example, twitter API belongs to BBC Browser and Soccer Shots mashups Mashups similarities are measured based on the following Formula (2):

$$Sim_{Mashups}\ (M1,M2)=\frac{\lvert SM1\cap SM2\rvert}{\lvert SM1\cup SM2\rvert} \qquad (2)$$

Where SM1 and SM2 represent services composing M1 and M2.

Users Relationships (*Track link***).** We consider two types of relationships between users: follow and similarity relationships. The *follow relationship* represents a relation of interest between two users. The *similarity* relationship between users is determined if two users follow the same services, mashups and eventually other users, then these users may have similar interests. A relation is then inferred between these two users, and it is labeled as a similarity relationship.

The *similarity* relationship between users is determined according to the number of services and mashups that users have in common in their watchlist. Users deploying several common services may have similar interests and could be considered as similar. The similarity between two users u_i and u_j is measured using the following function (3) [1].

$$Sim_u\ (u_i,u_j)=\frac{\lvert H_{ui}\cap H_{uj}\rvert}{\lvert H_{ui}\rvert} \qquad (3)$$

Where Hu_i and Hu_j are the recent histories of users u_i and u_j respectively.

Follows we will describe services/mashups recommendation approach based on this multigraph.

4 MultiGraph based Recommendation Approach

We propose a hybrid recommendation approach [15] based on previously described relationships in the multigraph, user's record and preferences, invocations and *watch-lists* in order to recommend relevant services/mashups.

We consider a scenario when the user searches for a Web service. The recommendation process described in Fig. 3 involves: i) Web services search in order to find the most relevant services in the services graph according to the user's request; ii) recommending relevant services according to the user's history, iii) recommending the mashups which share the same relevant services according to the mashups relationships in the graph and the mashups tracked by the user's neighbors and iv) ranking the resulted Web services/mashups list in terms of the services/mashups' according to theirs popularities to recommend the most relevant ones.

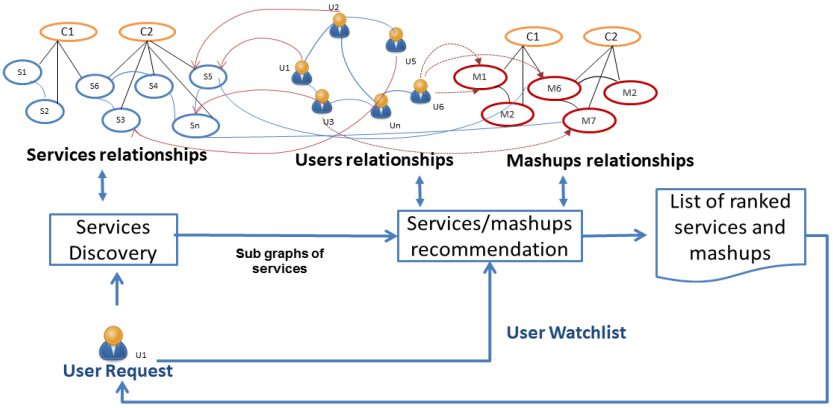

Fig. 3. Recommendation Process

4.1 Web Services Discovery

Given a user query, the search process consists in retrieving the most relevant services. The process returns the top k-similar Web services that are relevant. The user query is keyword-based and may include one (or more) category (ies), a service name, tags, service's protocol (SOAP, REST, etc.). For the similarity between a user request R and each service in the graph, we combine several string similarity functions [14], such as: *Ngram* for services names, *cosinus* for categories, *Jaccard* for tags and *cosinus* for textual descriptions.

4.2 Web Services/mashups Recommendation

Services Recommendation Process. The recommendation process is based on the previously generated multigraph. To recommend Web services, we need to rank them based on their popularity scores. The popularity of a service (*Pop(s)*) denotes the number of previously recorded usages (Hu) (of the user and his/her neighbors) this service has been involved in. The neighbors of a user are those related to him/her in the graph. We choose to Top K most similar users and we compute the popularity of a service in using correspondence matrix M (users are the lines, and services are the columns):

$$M[ui, sj] = \begin{cases} 1 & if\ ui\ tracks\ sj \\ 0 & else \end{cases}$$

$$Pop(s_j) = \frac{\sum_{i \in |U|} M[u_i, s_j]}{|U|} \qquad (4)$$

Mashup recommendation Process. The mashups recommendation is based on the relationships between services, mashups and users in the multigraph. The recommendation algorithm will return the popular mashups that share similar services and the mashups tracked by the user's neighbors. The popularity of mashups is computed according to the tracks relations between users and mashups.

5 Experimental Results

In this section, we describe the generation of a proposed multigraph model and the experimental obtained results following the implementation of our recommender system[4] with a real dataset crawled from ProgrammableWeb.

5.1 Implementation

Fig. 4 illustrates the main tasks involved to generate the graph database.

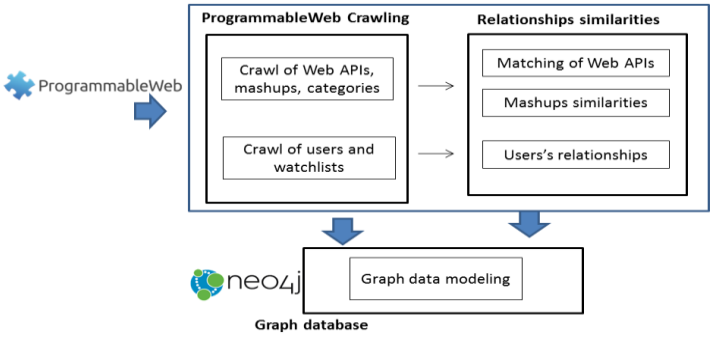

Fig. 4. Workflow of the multigraph model

[4] http://www.lsis.org/sellamis/Projects.html#WeS-ReG

For evaluation, we crawled 828 users, 700 Web services, 300 mashups and 344 watchlists (users' previous tracks). The relationships similarities module computes similarities among services, mashups and users in order to build the graphs. We built a Neo4J graph database consisting of 1460 nodes (116 categories, 700 services, and 344 users having a watchlist, 300 mashups) and 1756 relationships. The users' level consists of nodes (users) and edges that are labeled as *FOLLOW* relationships between users. At the services (respect. Mashups) level, services (mashups) are grouped by categories as illustrated in Fig. 5. We have defined four types of relationship between nodes: 1) *BELONGS_TO* is established between a service(s) (or mashups) and a category, 2) *SIMILAR* is the similarity relationship between services (resp. mashups), 3) *BELONGS_To_Mashups* denotes a link between services and mashups, and 4) *TRACK* is the link between users and services /mashups.

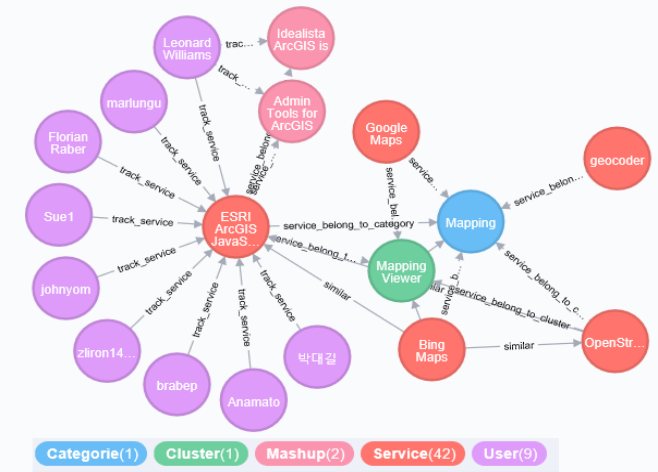

Fig. 5. Excerpt of Users/Services Multigraph

5.2 Experimental Evaluation

The goal of the experimentations is to evaluate the user's satisfaction with recommended services based on graph database exploration. The recommendation system performs graph analytics to produce a set of recommended services and mashups according to a user's request. Our assumption is that the user belongs to the user's graph. A user's query consists of a category and a set of keywords. Filtering by category is first carried out and then the similarity between user's query and services' (mashups') components is computed to find the most similar services or mashups. The recommendation system returns a set of ranked services (mashups) which may be of interest to the user. These services are retrieved from the list of discovered services and then ranked according to the service popularity.

Two types of experiments were conducted: first, we assess the performance of our system in term of CPU time. Second, we evaluate the quality of the recommendations.

Our experiments have been conducted on a dual Core i 7@2.50G PC with 8G RAM, under Windows 10.

CPU time Evaluation.

We compared the execution time of our recommendation system called WReG with that obtained by known recommendation approaches in the literature using the librec framework[5] namely, i) *TrustSVD*[10] which is a trust-based matrix factorization technique recommendation system that analyzes the social trust data from real-world data sets ; ii) *Recommendation of popular services* which is the applied strategy by ProgrammableWeb. It recommends the most popular services on the basis of use in mashups. The most popular service is the most used in mashups.

We carried out 10 times our algorithm with random inputs (the users' services previous uses for WReG and QoS of these same services for other approaches). Fig. 6 illustrates the results in terms of the obtained CPU time in seconds for the different approaches.

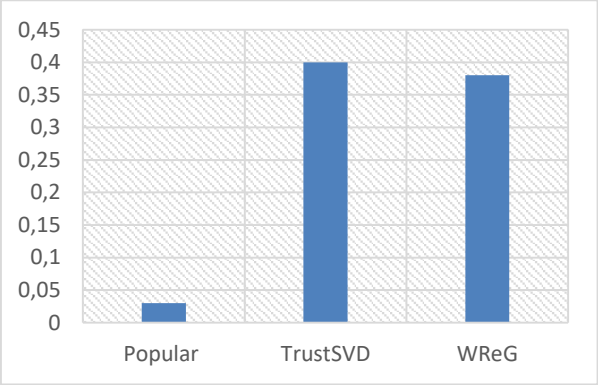

Fig. 6. Execution time (ms) of the recommendation approaches

We notice that the *recommendation of popular services* is more efficient than the other approaches. This can be vindicated by the fact that this approach finds out only the most frequent services used in mashups. WReG and TrustSVD approaches are more costly due to the data filtering process which explore the users' neighbors' history identified based on users relations in the multigraph. However, WReG is more efficient since it does not perform any other similarity measurements between users. These measurements are computed during the graph generation.

Quality of the Recommendation

In order to estimate the quality of recommendations generated by our system, we divide the dataset (users' tracks of services and mashups extracted from watchlists) into a training set and a test set. To evaluate the quality and the performance of the recommendation based graph database system, we use precision, recall, RMSE and hit-rank measures.

[5] http://www.librec.net/

Let's PR denote the set of relevant recommended services, R the set of recommended services and P the set of relevant services.

Precision: refers to a ratio of correctly predicted (satisfying user) services to the number of all recommended services: $Precision=\frac{|PR|}{|R|}$

Recall: refers to the ratio of correctly predicted services to the number of all the services satisfying the user in the testing set*:* $Recall=\frac{|PR|}{|P|}$

RMSE: The Root Mean Squared Error (RMSE) is used in evaluating accuracy of predicted rating (r_s). Let $r_s=\begin{cases}1 \text{ if } s \in P \\ 0 \text{ else}\end{cases}$.

$$RMSE=\sqrt{\frac{\sum_{u,s}(1-r_s)}{N}}$$

Where N is the number of recommended services.

Hit-rank: takes into account the ranks of returned services.

$$Hit\text{-}rank=\frac{1}{m*|R|}\sum_{u\in U}\sum_{i=1}^{h}\frac{1}{pi}$$

where h is the number of relevant services occurring at the positions $p_1,p_2, \ldots,p_h$ within the recommendation list; m is the total number of the users.

To compute the recall, precision and Hit-rank, we used 5 and 10 first resulting recommended services (Fig.7) and mashups (Top 5, Top 10) (Fig.8). We note that the accuracy of the recommendations increases proportionally with the number of returned services/mashups. The likelihood of providing a relevant service is greater when the number of recommended services increases. Indeed, it may be difficult for recommendation algorithms to identify all intersting services for users, while it is relatively easier to recommend a subset of relevant services.

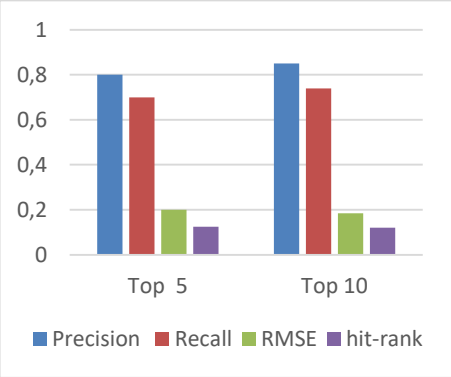

Fig. 7. Recall, Precision, RMSE and Hit-rank numbers (w.r.t the number of recommended services)

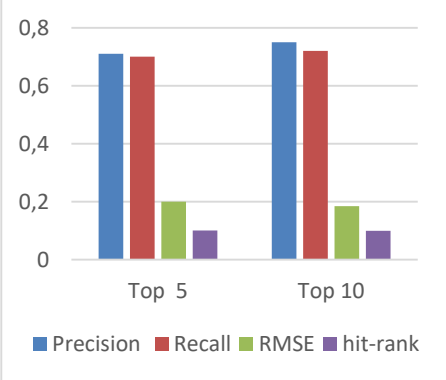

Fig. 8. Recall, Precision, RMSE and Hit-rank numbers (w.r.t the number of recommended mashups)

Table 2 shows the performance comparison in terms of precision@5, precision@10 recall@5, recall@10, hit-rank@5, hit-rank@10 and RMSE@5, RMSE@10 obtained values.
We note that WReG performs better than the other approaches. This can be justified by the fact that, unlike existing approaches, WReG recommendation algorithm is not based only on intra-services and intra-users relationships but also on users-services and users-mashups relationships. TrustSVD that is the closest to our system gives good precision values since it takes account of trust relations between users and services. But as all rating based approaches, it is not able to recommend services in the lack of rating values or invocation histories. Unlike TrustSVD, our approach provides good recommendations of services and mashups even when users do not have services in their histories. Furthermore, one may notice that *Popular*, which is the *ProgrammableWeb* approach, returns the lowest results compared to TrustSVD and WReG. This is due to the fact that *Popular* does not take into account users' interests and recommends the same services to all users. The results are not personalized.

Table 2 Comparison of the recommendation approaches

Approaches	Precision @5	Precision @10	Recall @5	Recall @10	RMSE @5	RMSE @10
TrustSVD	0.73	0.75	0.61	0.63	0.211	0.2
WReG	**0.80**	**0.85**	**0.70**	**0.74**	**0.2**	**0.185**
Popular	0.41	0.39	0.34	0.61	0.31	0.3

To summarize, our graph based recommender system is based on intra-relations (users, services, and mashups) and relationships between users and services/mashups. Compared to existing service recommendation systems, ours performs better in most cases. The neighborhood size and the users' histories affect positively the accuracy of our approach.

6 Conclusion

Since the advent of services oriented approaches and besides the availability of services through existing directories/catalogues such as ProgrammableWeb, Web services' gates still remain not well-structured to facilitate the discovery and composition processes.

In this paper, we describe an approach where a graph-based database model is used to structure the space of APIs, mashups and users. In order to show the extent of the graph, we propose a web services and mashups recommendation system. The proposed approach was implemented using Neo4J, and experiments have been conducted with real dataset crawled from ProgrammableWeb. Despite the lack of a benchmarking system, promising experimental results are despite the non-availability of relevant and specific benchmarks.

In the future, we plan to extend this work to service management for IoT in order to perform IoT services discovery.

References

1. Berry, M. W. Drmac, Z., Jessup, E. R.: Matrices, vector spaces, and information retrieval. SIAM review, vol.41 (1999) 335-362
2. Bianchini,D., Antonellis, V. D., Melchiori, M.: Link-Based Viewing of Multiple Web API Repositories. In Database and Expert Systems Applications - 25th International Conference, DEXA 2014, Munich, Germany, (2014) 362–376
3. Liang, T., Chen, L., Wu, J., Bouguettaya, A.: Exploiting Heterogeneous Information for Tag Recommendation in API Management. IEEE International Conference on Web Services ICWS 2016 (2016) 436-443
4. Chen, W., Paik, I., Hung, d P. C.: Constructing a global social service network for better quality of web service discovery. IEEE Trans. Services Computing, vol.8 (2015) 284-298
5. Chen, W., Paik, I.: Improving efficiency of service discovery using Linked databased service publication. Inf. Syst. Front., vol. 15, no. 4, (2013) 613–625
6. Lyu, S., Liu, J., Tang, M., Kang, G., Cao, B., Duan, Y.: Three-Level Views of the Web Service Network: An Empirical Study Based on ProgrammableWeb. IEEE International Congress on Big Data (BigData Congress 2014) 374-381
7. Deng, S., Huang, L., Yin, Y., Tang, W.: Trust-based service recommendation in social network. Appl. Math, vol.9 (2015) 1567-1574
8. Deng, S., Huang, L. Xu, G.: Social network-based service recommendation with trust enhancement. Expert Systems with Applications. vol.(41) (2014) 8075-8084
9. Liang, T., Chen, L., Wu, J., Dong, H., Bouguettaya, A.: Meta-Path Based Service Recommendation in Heterogeneous Information Networks. 14th International Conference (ICSOC) Service-Oriented Computing, vol.9936, (2016) 371-386
10. Guo, G., Zhang, J., Yorke-Smith., N.: TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, (2015) 123-129
11. Aljalbout, S., Boucelma, O., Sellami, S.: Modeling and Retrieving Linked RESTful APIs: A Graph Database Approach. On the Move to Meaningful Internet Systems: {OTM} 2015 Conferences Confederated International Conferences: CoopIS, ODBASE, and CTC, (2015) 443-450
12. 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, vol.19 (2011) 121-132
13. Maaradji, A., Hacid, H., Skraba, R., Lateef, A., Daigremont, J., Crespi, N.: Social-based web services discovery and composition for step-by-step mashup completion. IEEE International Conference on Web Services (ICWS 2011) (2011) 700-701
14. Jackson, D.A., M Somers, K., Harvey, H.: Similarity coefficients : measures of co-occurrence and association or simply measures of occurrence?. In : The American Naturalist (1989) 436–453
15. Bobadilla, J., Ortega, F., Hernando, A, GutiéRrez, A.: Recommender systems survey. Know.-Based Syst., vol. 46 (2013) 109-132