A Multigraph Approach for Web Services Recommendation

Fatma Slaimi[1(✉)], Sana Sellami[1], Omar Boucelma[1], and Ahlem Ben Hassine[2]

[1] Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, 13397 Marseille, France
{fatma.slaimi, sana.sellami, omar.boucelma}@univ-amu.fr
[2] National School of Computer Science (ENSI), University of Manouba, Manouba, Tunisia
ahlembh@gmail.com

Abstract. In this paper, we describe a Web services recommendation approach where the services' ecosystem is represented as a heterogeneous multigraph, and edges may have different semantics. The recommendation process relies on clustering techniques to suggest services "of interest" to a user. Our approach has been implemented as a tool called WesReG (Web services Recommendation with Graphs) on top of Neo4j and its cypher query language. We present the system implementation details and present the results of experiments on a collection of real Web services.

Keywords: Web services · Recommendation · Graph database · Neo4j

1 Introduction

Recommendation Systems (RS for short) have proved useful to exploit Web consumer behavior in order to recommend an item of interest to a client e.g., a product, a document or a book to cite a few. During the last decade, several Web service recommenders have been proposed in the literature [21–23]. Most of those systems are based on similarities between previously used services, as well on the assessments of Quality-of-Service (QoS) combined with collaborative filtering (CF).

With the availability of advanced service registries such as *ProgrammableWeb*[1] (*PWeb* for short), the services ecosystem may be represented as a linked data structure where both users and services are members of a graph. Within this setting, web services' consumers may leverage of the graph structure and semantics in receiving (or participating to) a service recommendation.

The goal of this paper is a novel Web service recommendation system based on users/services relationships. The idea is to make leverage of the available information on objects (services and users) in order to build a heterogeneous multigraph where nodes (services and users) are connected by labeled edges having different semantics,

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

C. Debruyne et al. (Eds.): OTM 2016 Conferences, LNCS 10033, pp. 282–299, 2016.
DOI: 10.1007/978-3-319-48472-3_16

i.e., similarity, popularity, follow and track relations, etc. A service may be recommended to a given user either as a response to his/her request or based on his/her profile.

The contribution of this work is twofold: (i) the design of a multigraph model where intra-services, intra-users and inter services/users links are exhibited; (ii) a novel recommendation approach based multigraph search is proposed. In addition, a prototype has been implemented on top of a Neo4j graph database, and an experimental evaluation has been performed with a real dataset.

The rest of the paper is organized as follows; Sect. 2 reviews some Web services recommendation approaches. Section 3 describes the proposed recommendation approach. Section 4 discusses experimental results and Sect. 5 concludes the paper.

2 Related Work

Most of Web services Recommendation Systems proposed in the literature are based on Filtering techniques on users' data and items (i.e. services). We distinguish two categories of filtering: content based filtering and collaborative filtering. Recommendation based content [2] exploits the user's profile which includes the user concerns defined from the evaluations that he/she gave to the item, i.e. comments, opinion, rating, preferences, use, etc. It analyses content similarities [3] among items, previously used by the underlying user. This approach establishes the profile of a user while investigating his/her concerns and provides him with items similar to those he/she already used. It is noteworthy that the main benefit of recommendation based content approaches resides in connecting items to a user's profile. Also users are independent of each other; hence a user may take advantages of the recommendation process even if he/she is the unique user of this system. However, a user's concerns may change over time making the list of items to be recommended very tight [18]. Actually, the recommended items are often similar and identical (compared to their contents) to those previously referred by the user.

The collaborative filtering (CF) system proceeds as follows: (1) a user expresses his preferences by rating items. These ratings are considered as user preferences; (2) the system matches some user ratings against other users' ratings and detects users with the most "similar" preferences; (3) the system recommends the highly-rated items for a user that are not forcibly rated by him. These systems are based on the idea that users with similar rates may have similar interests. Nevertheless, CF systems are not able to recommend any item for a new user, i.e. a user without use history for items. This problem is known as a cold start problem. In addition, in such filtering system, users should provide evaluations, opinions on items they already used in order to build their profile and to figure out their preferences. Nevertheless, the contents of items to recommend were not parsed. In order to overcome this latter limit, hybrid recommendation approaches were proposed. Their idea is to merge content based approaches with collaborative filtering based approaches [5] in order to cope with aforementioned problems, i.e. problems of both cold start and lack of data for the new users.

In the following, we will focus on both hybrid approaches that are based on trust relationships called "social recommendation" approaches and linked data approaches.

2.1 Social Recommendation Approaches

With the growth of social networks and the emersion of social Web, several recommendation systems that make use of users' relationships, e.g. Twitter follow, etc. were proposed [7, 8, 17]. Most of them depend on trust relationships.

Trust relationships based recommendation approaches differ from traditional collaborative filtering approaches (history based similarity) in two points. First, the concept of neighborhood (expressed by similar users) is replaced by the concept of trust and the user can himself choose his/her own friends (trustful users). Second, the recommendation system ensures the spreading of the trust relationships: if a user u trusts another user v, and if v trusts a third user w, then u trusts w. Trust spreading allows a user to take advantage from the experiences of the friends of his/her friends [9, 15, 18]. Trust concept was applied as a type of relationship between two persons (local trust), or an overview of the community's view regarding a user (global trust). The former is more frequent in the recommendation systems.

SoCo [13] is a Web service recommendation framework based on social networks analysis. SoCo enables the discovery and the composition process by transforming the interactions between users and services into social network interactions among users represented as a graph. This graph links users according to their common interests; it describes their profiles including their preferences and their previous system usages. The trust relationships are established between two users when a user uses a Web service offered by the other. Services preferred by trusted users are recommended. However, trust relationships do not consider similar users' behaviors or functionally similar Web services.

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.

The recommendation process starts by identifying the current user's preferences using his/her previous uses and the uses of his/her related trustful users. Then, it measures the similarity between them in order to identify the most similar ones. The degree of trust of a user for a service is predicted by combining his/her preferences with the previous uses of similar 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.

Social recommendation approaches are especially based on trust relationships modeled by a user graph. This graph will be exploited to recommend relevant Web services. Unlike traditional approaches, the use of a user graph allows an efficient identification of neighborhoods users and provides more accurate results.

2.2 Linked Data Oriented Recommendation Approaches

Linked data[2] "*is about using the web to connect related data that wasn't previously linked*". The Linked Data principles aim to facilitate the navigation from a document to another. Recently, the linked services network has drawn attention among the web services research community. For example, in [20], authors propose a framework for defining a linked view over multiple repositories and for searching their content. Linked services aim to enhance services discovery and recommendation. For example, the recommendation approach described in [11], considers services as resources that can be identified by their URIs (Uniform Resource Identifier). Authors in this research propose a resources-users graph, where entities resources/users are linked by concern/interest relationships. A matrix indicating the users' services usage, deduced from this graph, is provided as input for a collaborative filter to determine the top K services to recommend.

In [12], authors proposed a new 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.

In [14], authors propose a new recommendation approach for services and things using an objects-users-services graph, i.e., a graph interconnecting things, users and services. A user is connected to every object with which he/she interacts and to every service he/she used. Their proposed system uses the services that are from one side used by the user and on the other side related to intelligent objects, e.g. home, PDA, Smartphone, etc. Their idea is to identify the top five relevant services ranked according to their popularity. The popularity of a service is defined by the number of objects in its neighborhoods.

2.3 Discussion

Most of recommendation systems mentioned above may be considered as hybrid ones, because they use both items' content and relationships among users. The advent of social networks changes the nature of "useful" metadata in the recommendation process [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. In addition, the social approach has solved the cold start problem. Moreover, the convergence towards a Web of linked data 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.

Within this context, we propose a novel hybrid approach that combines both social recommendation techniques and linked data. The idea of our approach is to consider

[2] http://linkeddata.org/.

explicit information (users' follow relationships) with users' preferences resulting from observance of previous follow relationships of Web services and users. Note that, the interest relationships between system's users and published Web services are represented by a multi graph of users/services. 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 Web Service Recommendation with Graphs (WesReG)

The multigraph based recommendation approach (*aka* WesReG) makes explicit different types of relationships among both users and services in order to recommend relevant Web services to users with common interests. Relationships are the edges of the users/services multigraph (cf. Definition 1).

Definition 1. Users/services multigraph. ***MG* = <*U, S, E* >** is an oriented heterogeneous multigraph where

$U = \{u_1, u_2, \ldots, u_N\}$ is a set of N users (**vertices**), and
$S = \{s_1, s_2, \ldots, s_M\}$ is a set of M services (**vertices**), and

$$E = \Big\{ (u_i, u_j) \mid \text{user } u_i \text{ tracks (or is similar to) a user } v_j \Big\}$$
$$\cup \{ (u_i, s_k) \mid \text{user } u_i \text{ tracks a service } s_k \}$$
$$\cup \{ (s_k, s_1) \mid \text{service } s_k \text{ is similar to service } s_1 \}$$

is the set **edges**, that materialize oriented relations between vertices.

WesReG involves three steps as illustrated in Fig. 1:

1. Modeling the users/services relationships according to their similarities.
2. Building a heterogeneous multigraph (MG) where the nodes represent users (resp. services) and the edges illustrate the different relationships among these users (resp. services) and between users and services.
3. Recommending the most relevant Web services for a target user based on the resulting graph.

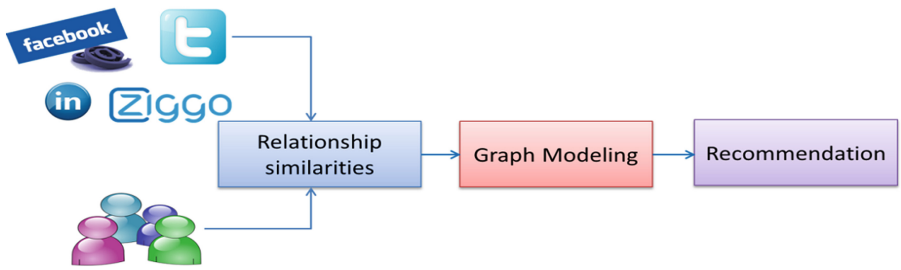

Fig. 1. Recommendation approach

3.1 Users/Services Relationships Similarities

In this section, we will describe the different relationships that may exist between users (resp. services) in order to build the users/services multigraph. These relationships are defined based on the similarities among entities. Similarities are based on follow relationships, services track relationships and properties similarities. Track relationships are expressed by means of *PWeb* tracks. A user can track (follow) either: Web services, mashups, searches or others users. The recommendation process will take in account these relationships, user's record and preferences, invocations and watchlists in order to recommend relevant services.

Users Relationships. We consider two types of relationships between users: follow and similarity relationships.

The *follow (track) relationship* represents a relation of interest and can be inferred from users' (PWeb) *watchlist* (i.e. history). Since a user may track Web services and mashups, or create mashups, her/his history is defined by means of her/his followees or the mashups she/he created.

Actually, if two users follow the same services, mashups and eventually other users, then these users may have similar interests (see Fig. 2). A relation is then inferred between these two users, and it is labeled as a similarity relationship.

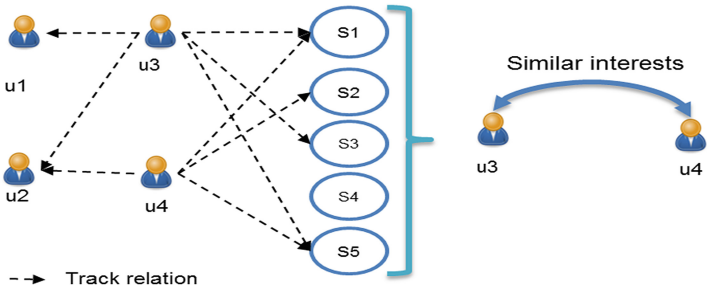

Fig. 2. Relationships among users

The *similarity* relationship between users is determined according to the number of services 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 (1) [1].

$$Sim(u_i, u_j) = \frac{|H_{ui} \cap H_{uj}|}{|H_{ui}|} \tag{1}$$

Where H_{ui} and H_{uj} are the recent histories of users u_i and u_j respectively.

Services Relationships. Currently, several Web services (e.g. google Maps, big Maps) that are exposed on existing platforms such as *PWeb* are functionally equivalent. A relationship can be defined between each pair of functionally similar services. This similarity is determined by comparing services' description items (Fig. 3), i.e., categories, name, tags and description. The similarity between two services may be calculated using a matchmaker such as SR-REST [16].

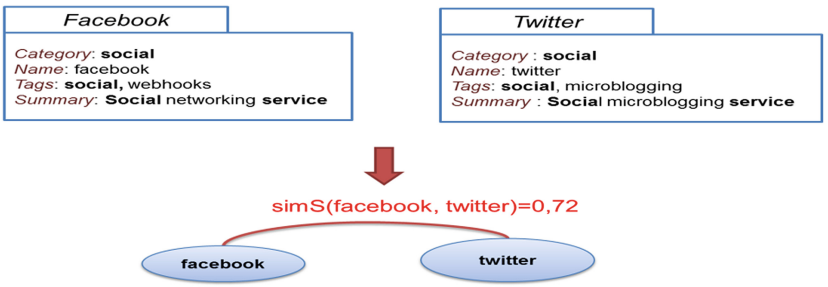

Fig. 3. Similarity relationship between two services

Graph Generation Process. This process builds a heterogeneous users/services multigraph by means of *users/services relationships* previously described. The resulting multigraph is depicted on several levels: Categories level, Services level and Users level. A service may belong to one or several categories (membership relationship). For example, as given in Fig. 4, *facebook* and *twitter* services are considered

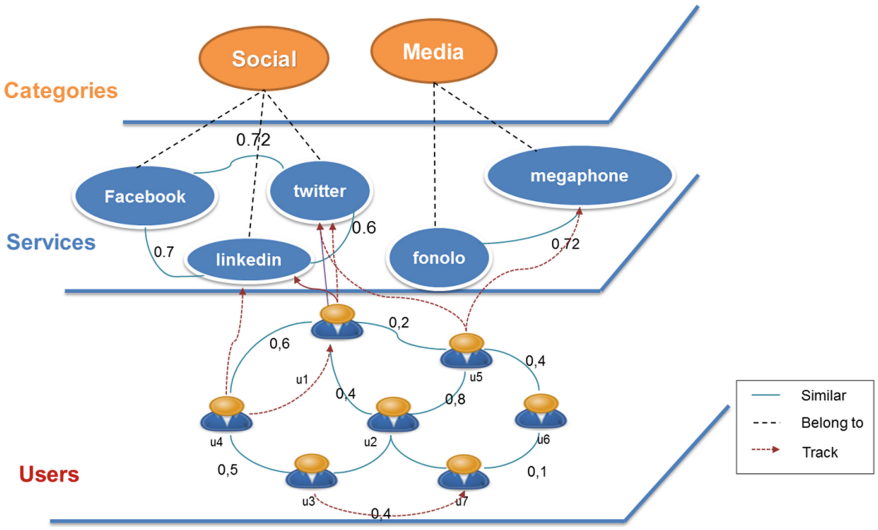

Fig. 4. Users/services multigraph

from the same category *Social* services. Consequently, a user can follow services used by other users even if he/she is not directly connected to these users.

Since our main objective is to recommend relevant Web services for any requester, follows we will describe how this multigraph will be explored to find out such set.

3.2 Web Services Recommendation Process

Our objective is to perform an online recommendation where a user expresses his/her request and the system provides him/her by a set of Web services worthy of interest (see WesReG Algorithm). The idea is to leverage the users/services multigraph by exploring the track/similarity relationships and to come up with a set of Web services that would interest the requester. However, as proved by several studies [6], exploring the multigraph is very costly especially when we have to deal with a huge amount of data. Therefore, and in order to optimize and to ease the search process, we propose to integrate an offline preprocessing step that consists in splitting up the multigraph into several clusters of similar Web services.

As illustrated in Fig. 5, the recommendation approach involves: (*i*) clustering services graph to get the clusters of similar services; (*ii*) Web services search in order to find the most relevant services in the services' clusters according to the user's request and history; (*iii*) ranking the resulted Web services list in term of the services' popularities to recommend the most relevant ones.

Clustering Process. The main of the clustering process is to bring together a set of items having the same criteria. In order to reduce the search space and consequently improve the quality of the recommendation results, we propose to cluster the Web services graph into similar Web services groups according to the services relationships. Therefore, we use an agglomerative, hierarchical and incremental clustering algorithm [6] for clustering services. Similar services are grouped by categories and are merged as

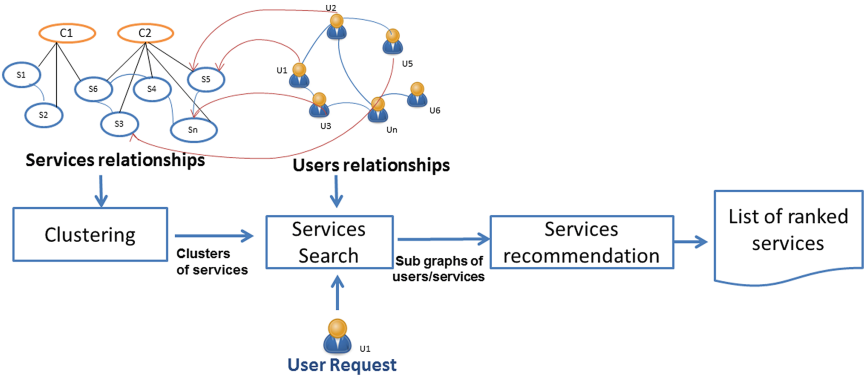

Fig. 5. Recommendation process

one move up the hierarchy. The output of this process is set of clusters of services. Clusters are labeled by the most used keywords in services' descriptions.

Services Search Process. Given a user query, the search process consists in retrieving the most relevant service in the clusters of 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.). Two types of filtering process are during the search process: by category and by cluster. For each cluster, we will extract the most used keywords in services' descriptions. For the similarity between a user request R and every cluster C (of keywords), we will use the Cosine function as given in Eq. 2.

$$Cosinus(C, R) = \frac{VC \cdot VR}{||VC|| \cdot ||VR||} \tag{2}$$

with VC represents the terms characterizing cluster C while VR represents the user request's keywords.

Services Recommendation Process. The recommendation process is based on the previously generated users/services subgraphs. However, the set of generated services may include several functionally equivalent Web services. To recommend a Web service, 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 (services are the lines, and users are the columns):

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

$$Pop(s_i) = \frac{\sum_{j \in |U|} M[s_i, u_j]}{|U|} \tag{3}$$

WesReG Algorithm

```
var: MG:users/services multigraph, u :target user, R : user's request
Output: RS, the set of recommended services
Begin
   RS ←∅;
   U ←∅;// neighbors of the user u
  // services search
  FilterCatgories(MG, R.categories);
  C←FilterClusters(MG, R.tags);
  RS ← Similarity(C,R);
  // neighbors search
  For each v in MG Do
    If ∃ rel(v, u) Then
      U ← U ⋃ v
    end if
  end for each
  For each  si in RS Do
    p[si] ←  popularity(si,U); // services popularity
  end for each
  Rank(RS, p); // ranking of services according to theirs popularity scores
  // select a service s from RS
  s ← select(RS);
  For each  si in MG Do
    If ∃rel(si, s) Then
      SR ← SR ⋃ si
    end if
  end For each
  For each si in RS Do
  // services` popularities
  p[si] ←  popularity(si, U);
  end For each
  Rank (RS, p);
  Return RS;
End
```

4 Experimental Results

In this section, we describe the WesReG recommendation system and the experimental obtained results using a real dataset retrieved from ProgrammableWeb.

4.1 Implementation

Figure 6 illustrates the main components of WesReG. The input is a set of *PWeb* users and APIs data stored in SQL Server and released as public dataset file on the Internet[3]. This database contains 10050 users, 69384 Web services and watchlists (users' previous uses). Among the used data set, only 4568 users have watchlists, as for Web services, 6362 of them (API and mashups) are tracked by the users [18].

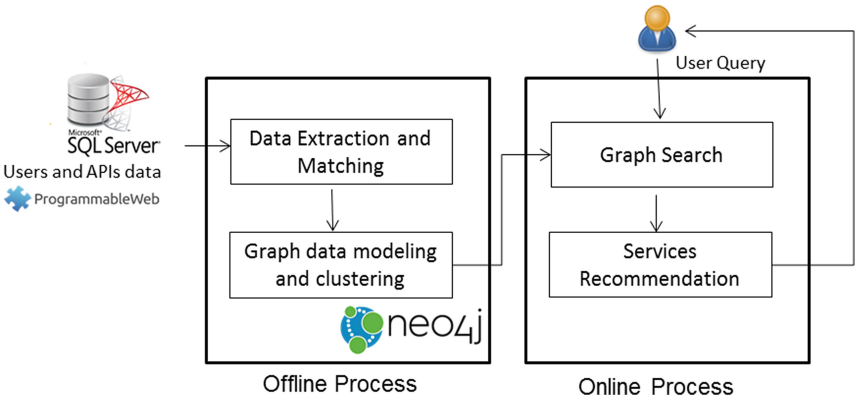

Fig. 6. WesReG architecture

Processing is performed in two steps: the offline process ingests raw data (users and services) and generates a Neo4j graph of clustered services and users. The online process performs graph analytics to produce a set of recommended services according to a user's request.

Data Extraction and Matching. This module computes similarities among services and users in order to build the graphs. Similarities are based on follow relationships and services track relationships available. We evaluate the similarities by matching services properties as described in our previous work [16].

Graph Data Modeling and Clustering. We built a Neo4J graph database consisting of 671 nodes (3 categories, 540 services, 28 clusters and 100 users) (Fig. 7) and 281 relationships. The users' level consists of nodes (users) and edges that are labeled as *FOLLOW* relationships between users. At the services level, services are grouped by

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

categories and clusters as illustrated in Fig. 8. We have defined 5 types of relationship between nodes: (1) *BELONGS_TO* is established between a service(s) and a category, (2) *SIMILAR* is the similarity relationship between services, (3) *BelongsTo_CAT* relies each cluster to the corresponding category, (4) *Cluster_LINK* denotes a link between services and clusters, and (5) *TRACK* is the link between users and services.

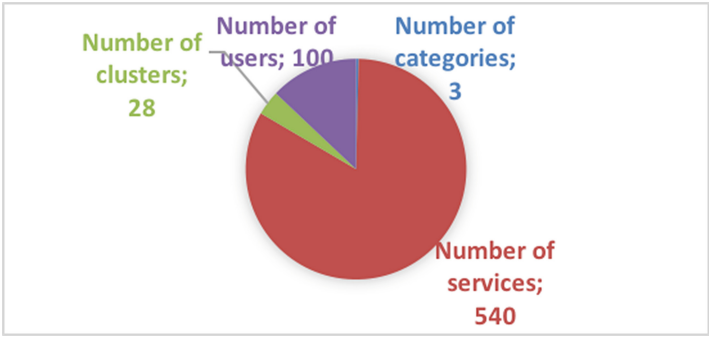

Fig. 7. Graph instance statistics

Graph Search. This component returns the most relevant services in the graph according to the user's query. 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 query is compared to the clusters of services in the graph by performing a function similarity. Cosine function (see formula 2) is used to calculate the similarity between the labels of clusters of services and the user's query.

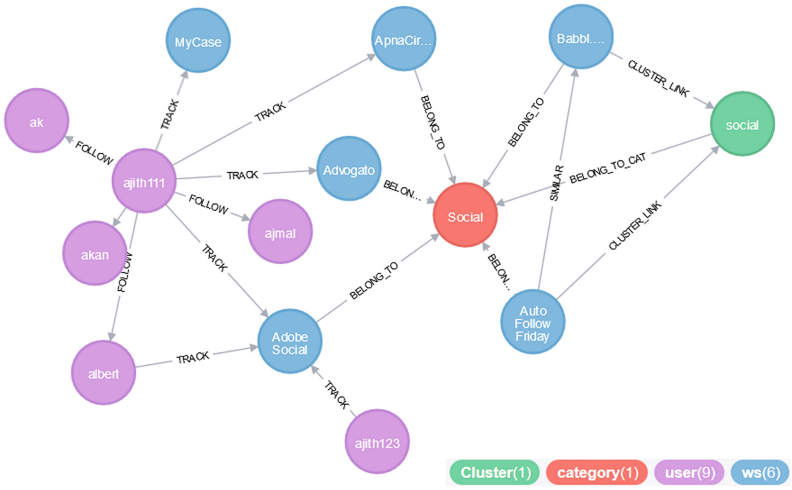

Fig. 8. Excerpt of users/services multi-graph

Services Recommendation. This module returns a set of ranked services which may be of interest to the user. These services are retrieved from the list of discovered services generated by the graph search module and then ranked according to the service popularity.

4.2 Experimental Evaluation

The goal of the experimentations is to evaluate the user's satisfaction with recommended services. 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 cpu@2.20G PC with 4G RAM, under Windows 7.

CPU time Evaluation. We computed the required CPU time for our proposed WesReG system with and without clustering process while varying the number of services in the graph.

As illustrated in Fig. 9, the clustering process has an important impact on the WesReG consumed CPU time when the number of services is greater than 200 nodes. WesReG with clustering is much more efficient than without clustering. Indeed, the number of comparisons has been reduced by using the clustering of web services.

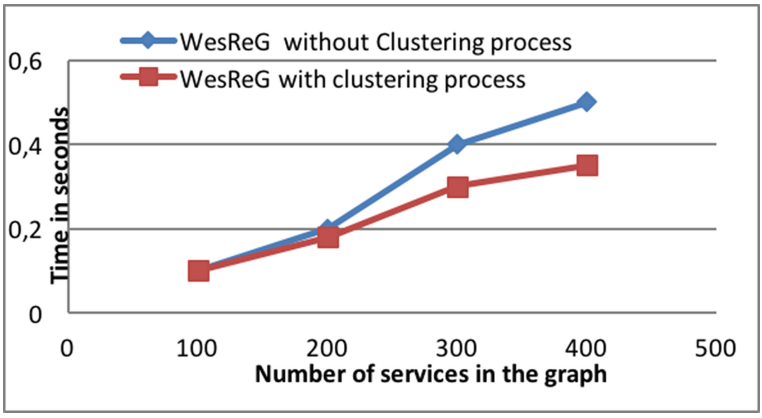

Fig. 9. Impact of the services number on WesReG

We then compared the WesReG execution time (with clustering) with that obtained by known recommendation approaches in the literature using the librec framework[4] 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.

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

This approach considers both explicit and implicit ratings to recommend; (ii) *Association Rules (AR)* [19] which only applies association rules in order to recommend services and (iii) *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 WesReG and QoS of these same services for other approaches). Figure 10 illustrates the results in terms of the obtained CPU time in seconds for the fourth approaches.

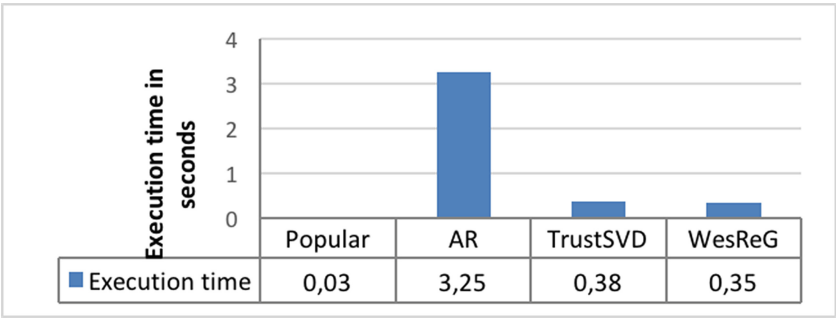

Fig. 10. Comparison of the four approaches

We notice that the *popular recommendation* is more efficient than the other approaches. This can be vindicated by the fact that this approach finds out the most frequent services in the users' previous uses. While AR recommendation approach is costly in terms of CPU time due to the fact that frequent services detection algorithms and services correlation processes require several exhaustive investigations of the users' previous uses.

WesReG and TrustSVD approaches are less costly than AR approach due to the data filtering process which only explore the users' neighbors' history. However, WesReG 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. To evaluate the quality and the performance of our approach, we use precision, recall measures and hit-rank.

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:

$$\text{Precision} = \frac{|\text{PR}|}{|\text{R}|}$$

Recall: refers to the ratio of correctly predicted services to the number of all the services satisfying the user in the testing set:

$$\text{Recall} = \frac{|\text{PR}|}{|\text{P}|}$$

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

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

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 only 5, 10 and 15 first resulting recommended services (Top 5, Top 10, Top 15). Figure 11 shows the recall, precision and Hit-rank measures of WesReG w.r.t. the number of recommended services.

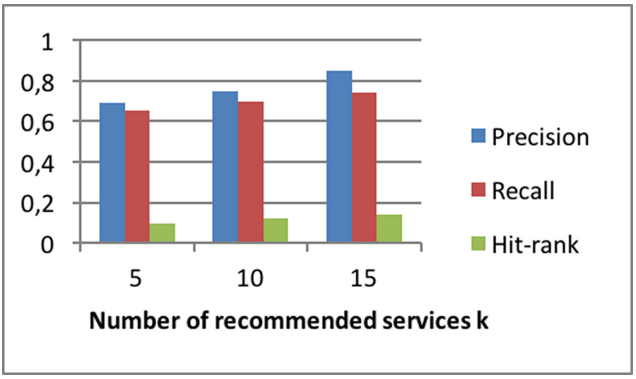

Fig. 11. Recall, precision and hit-rank numbers (w.r.t the number of recommended services)

Figures 12 and 13 show the performance comparison (in terms of precision and recall) for our approach versus approaches developed with LibRec. Table 1 illustrates the recall and precision obtained values (for Top5 and Top 10).

We note that WesReG performs better than the other approaches. This can be justified by the fact that, unlike existing approaches, WesReG recommendation algorithm is not based only on intra-services and intra-users relationships but also on users-services 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 results 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

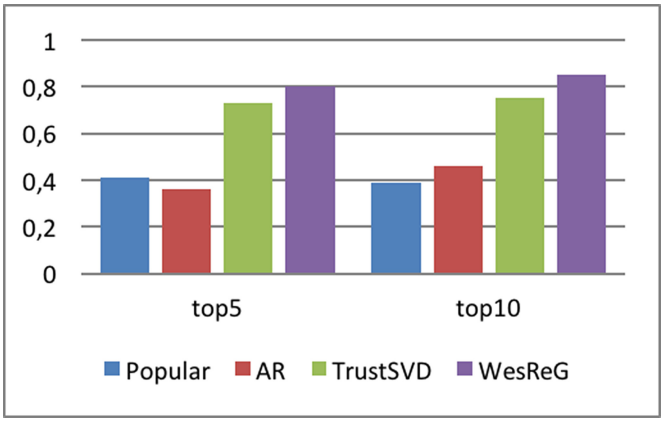

Fig. 12. Precision variations

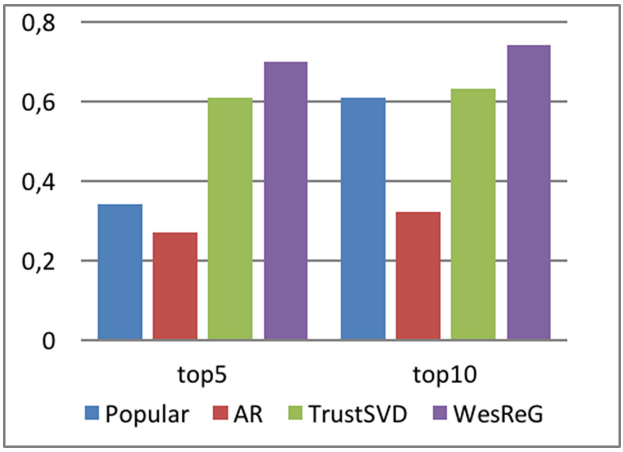

Fig. 13. Recall variations

Table 1. Comparison of the four recommendation approaches

Approaches	Precision Top5	Precision Top10	Recall Top5	Recall Top10
TrustSVD	0.73	0.75	0.61	0.63
WesReG	**0.80**	**0.85**	**0.70**	**0.74**
Popular	0.41	0.39	0.34	0.61
Ass. Rules	0.36	0.46	0.27	0.32

compared to TrustSVD and WesReG. 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.

To summarize, our recommender system is based on intra-relations (users, services) and relationships between users and services. 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.

5 Conclusion

Since the advent of services oriented approaches, the ever-growing number of available Web services (or APIs) leads to a real expansion of this phenomenon especially on the cloud. However, and besides the availability of these services through existing directories/catalogues, e.g. ProgrammableWeb, Web services' gates still remain non well-structured to facilitate the discovery and composition processes.

In this paper, we proposed a novel approach and a novel Web services recommendation system that explores a graph oriented DB, which we elaborated to represent Web services ecosystem. The proposed approach was implemented using a Neo4J DB with real dataset. We mainly explored explicit track relationships extracted from ProgrammableWeb. Obtained experimental results are promising despite the non-availability of relevant and specific benchmarks.

Although using graphs to model the Web services ecosystem is not a new idea, from our perspective, this work is a first step towards building a real graph database, along the line of the Linked Data initiative. In future work, we plan to propose other graph structures, especially by exhibiting other semantic links, offering other appropriate recommendation methods based on processing huge graphs to be able to deal with larger amount of Web services.

References

1. Berry, M.W., Drmac, Z., Jessup, E.R.: Matrices, vector spaces, and information retrieval. SIAM Rev. **41**, 335–362 (1999)
2. Bobadilla, J., Ortega, F., Hernando, A., GutiéRrez, A.: Recommender systems survey. Know. Based Syst. **46**, 109–132 (2013)
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann Publishers Inc. (1998)
4. Chen, W., Paik, I., Hung, P.C.K.: Constructing a global social service network for better quality of web service discovery. IEEE Trans. Serv. Comput. **8**, 284–298 (2015)
5. Chen, Z., Jiang, Y., Zhao, Y.: A collaborative filtering recommendation algorithm based on user interest change and trust evaluation. JDCTA **4**, 106–113 (2010)
6. Choi, K., Yoo, D., Kim, G., Suh, Y.: A hybrid online-product recommendation system: combining implicit rating-based collaborative filtering and sequential pattern analysis. Electron. Commer. Res. Appl. **11**, 309–317 (2012)
7. Deng, S., Huang, L., Yin, Y., Tang, W.: Trust-based service recommendation in social network. Appl. Math. **9**, 1567–1574 (2015)

8. Deng, S., Huang, L., Xu, G.: Social network-based service recommendation with trust enhancement. Expert Syst. Appl. **41**, 8075–8084 (2014)
9. Golbeck, J., Hendler, J.: Filmtrust: movie recommendations using trust in web-based social networks. In: Proceedings of the IEEE Consumer Communications and Networking Conference, vol. 96, pp. 282–286 (2006)
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. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 123–129 (2015)
11. Heitmann, B., Hayes, C.: Using linked data to build open, collaborative recommender systems. In: AAAI Spring Symposium: Linked Data Meets Artificial Intelligence, pp. 76–81 (2010)
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. Simul. Model. Pract. Theor. **19**, 121–132 (2011)
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. In: IEEE International Conference on Web Services (ICWS 2011), pp. 700–701 (2011)
14. Mashal, I., Chung, T.-Y., Alsaryrah, O.: Toward service recommendation in internet of things. In: 2015 Seventh International Conference on Ubiquitous and Future Networks (ICUFN), pp. 328–331. IEEE (2015)
15. Massa, P., Avesani, P.: Trust-aware collaborative filtering for recommender systems. In: Meersman, R. (ed.) CoopIS/DOA/ODBASE 2004. LNCS, vol. 3290, pp. 492–508. Springer, Heidelberg (2004)
16. Slaimi, F., Sellami, S., Boucelma, O., Ben Hassine, A.: Flexible matchmaking for restful web services. In: Meersman, R., et al. (eds.) OTM 2013. LNCS, vol. 8185, pp. 542–554. Springer, Heidelberg (2013)
17. Walter, F.E., Battiston, S., Schweitzer, F.: A model of a trust-based recommendation system on a social network. Auton. Agent. Multi-Agent Syst. **16**, 57–74 (2008)
18. Zhang, X., He, K., Wang, J., Wang, C., Tian, G., Liu, J.: Web service recommendation based on watchlist via temporal and tag preference fusion. In: 2014 IEEE International Conference on Web Services (ICWS), pp. 281–288. IEEE (2014)
19. Kim, C., Kim, J.: A recommendation algorithm using multi-level association rules. In: IEEE/WIC International Conference on Web Intelligence (WI 2003), pp. 524–527 (2003)
20. Bianchini, D., De Antonellis, V., Melchiori, M.: Link-based viewing of multiple web API repositories. In: Decker, H., Lhotská, L., Link, S., Spies, M., Wagner, Roland R. (eds.) DEXA 2014, Part I. LNCS, vol. 8644, pp. 362–376. Springer, Heidelberg (2014)
21. Zheng, Z.B., Ma, H., Lyu, M.R., King, I.: QoS-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**, 140–152 (2011)
22. Cao, J., Wu, Z., Wang, Y., Zhuang, Y.: Hybrid collaborative filtering algorithm for bidirectional Web service recommendation. Knowl. Inf. Syst. **36**(3), 607–627 (2013)
23. Manikrao, U.S., Prabhakar, T.V.: Dynamic selection of web services with recommendation system. In: International Conference on Next Generation Web Services Practices (NWeSP 2005) (2005)