



Leveraging Track Relationships for Web Service Recommendation

Fatma Slaimi, Sana Sellami, Omar Boucelma, Ahlem Ben Hassine

► To cite this version:

Fatma Slaimi, Sana Sellami, Omar Boucelma, Ahlem Ben Hassine. Leveraging Track Relationships for Web Service Recommendation. 13th IEEE International Conference on e-Business Engineering (ICEBE), Nov 2016, Macau, China. hal-03538503

HAL Id: hal-03538503

<https://hal.science/hal-03538503>

Submitted on 21 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Leveraging Track Relationships for Web Service Recommendation

Fatma Slaimi, Sana Sellami, Omar Boucelma

Aix Marseille Université, CNRS, ENSAM, Université
de Toulon,

LSIS UMR 7296,13397, Marseille, France

{fatma.slaimi,sana.sellami,omar.boucelma}@univ-
amu.fr

Ahlem Ben Hassine

National School of Computer Science (ENSI),

University of Manouba, Tunisia

Ahlembh@gmail.com

Abstract—Existing Web services recommendation approaches are based on usage statistics or QoS properties, leaving aside the evolution of the services’ ecosystem. These approaches do not always capture new or more recent users’ preferences resulting in recommendations with possibly obsolete or less relevant services. In this paper, we describe a novel Web services recommendation approach where the services’ ecosystem is represented as a heterogeneous multi-graph, and edges may have different semantics. The recommendation process relies on data mining techniques to suggest services “of interest” to a user.

Keywords: *Service Recommendation; Web service; multi-Graph; Association rules; track relationship*

I. 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 [6] [10-11]. Most of those systems are based on both similarities between previously used services, and the assessments of the Quality-of-Service (QoS) combined with collaborative filtering (CF) [15]. These approaches rely on static data and hence cannot deal with a continuously evolving Web services ecosystem, which is composed of services, service providers, Mashups and (users) developers. In such eco-system, services can be updated, deleted making the whole ecosystem prone to obsolescence. In addition, most of those services do not provide QoS metadata.

Other approaches consider the exploitation of trusted relationships between services and users or between users themselves [1-3]. Trust relationships are established on the basis of similar service invocations or by similar evaluations of Web services by different users. However, in other settings, e.g. a social network, a user might be interested by a service recommended by a friend rather than by other users, even if these users have provided same rankings for a service, or did invoke the same service. In addition, users’ information feedback is often missing and user interests may change over time. Random evaluations provided by spam

users may be another limitation for these recommendation systems.

Now that the big picture is drawn, why would it be worthy to extend/improve/develop recommendation approaches in the Web services ecosystem?

In this paper, we describe WSR-Track, a novel service recommendation system where (1) the service ecosystem is represented as a heterogeneous multi-graph, (2) nodes represent users (resp. services), and edges represent either trust relationships between users, or usage relationships between a user and a service, and (3) nodes and edges properties are integrated in the service discovery process. Trust relationships are expressed by means of *ProgrammableWeb*¹ tracks. Usage relationships represent a service invocation. The recommendation process takes into account relationships, user records and preferences, invocations, watchlists and mashups.

The rest of the paper is organized as follows; related work is presented in Section 2 and Section 3 describes the WSR-Track recommendation system. Section 4 discusses experimental results and Section 5 concludes the paper.

II. RELATED WORK

Most of Web services RS proposed in the literature are based on Collaborative Filtering or CF [15][16]. A 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.

Other approaches were devoted to the use of QoS, ratings and relationships among Web services end-users [4].

Both approaches are reviewed below.

¹<http://www.programmableweb.com/>

A. QoS Aware approaches

QoS properties are often expressed by means of performance indicators such as price, response time, availability, accuracy, etc. Many researches on Web service selection, recommendation and composition use QoS in order to improve the performance of their systems [9]. Generally, RS combine QoS evaluations with CF to recommend services [4][10] and users [11].

In [4], the recommendation approach makes use of the QoS ratings given by a user to predict QoS evaluations of other services that she never used.

In [3], authors consider when the user is interested in the QoS evaluations of other users, they will be considered as trustful ones. Similarity between users is expressed as an (Euclidean) distance. First, authors choose a set of services to recommend, and then calculate the degree of user confidence in these services by taking into consideration his preferences in term of services' categories.

In [6], a recommendation system with semantic matching is described. The approach is based on CF and users' feedbacks, and helps a user to select a Web service from a list of similar ones. When a user invokes a service, he is asked to evaluate it. But in most cases, users do not provide explicit ratings [18].

Xi et al. [19] studied the impact of user's location in the prediction of QoS. They proposed a novel approach where users are hierarchically grouped according to their location and their QoS evaluations. Consequently, users from the same region are considered as similar. Given a user, the system searches similar users in the same location (region) instead of exploring the set of all users. However, belonging to the same region does not necessary mean having the same preferences.

Authors in [20] consider the impact of personalization of Web service items when computing the degree of similarity between users. That is, more popular services or services with more stable QoS should contribute less to user similarity measurement. Zhang et al. [21] propose to combine users QoS experiences, environment and user input (query) to predict Web services QoS values. But the environment and the user-input can hardly represent user's opinions/tastes.

B. Trust based approaches

The number of available services and users is increasing and so, some fake feedbacks of malicious users may be introduced as inputs to the RS: as a result, this may affect the quality of recommended services. This issue has been addressed in [2] [4-5] [7] [12-14] [22-23], where authors proposed trust-enhanced recommendation methods. Based on trust relations, trust-based approaches refer only to ratings given by trusted users.

SoCo [7] 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 a 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. Services preferred by trusted workers are recommended.

The work of Deng et al. [2] is based on the analysis of social networks' contents and users' relationships. Authors propose a Web service recommendation system based on trust relationships that are 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. TSR [2] the service recommendation algorithm proposed is based on trusts and calculates the similarity between users via their assessments of public services. TSR also calculates the confidence of the user in a service. This confidence depends on the assessments of similar users. The most confident Web services are recommended.

In [5], authors exploit the implicit users' ratings to build a CF-based RS that takes advantages of the user *WatchList* and preferences in terms of tags. The idea is to transform implicit evaluations into explicit ones to improve the recommendation accuracy. Empirical experiments based on *ProgrammableWeb* show that compared to CF, the recommendation system (with time preference and tags) provides results that are more accurate and more precise.

As mentioned above, most of RS inputs are users' relationships and similarities among users. In our work, the similarity between users is based on the following assumptions: 1) user A tracks user B, this means that A have similar interests as B in terms of Web services; 2) users tracking a user in common may have similar interests.

Moreover, unlike existing approaches which consider explicit user evaluations of services and QoS rating, we propose to exploit explicit users' relations, established via the track function to detect their services preferences. This information is obtained from their previous tracks. We assume the existence of track relations in *ProgrammableWeb* that play the role of explicit trust relationships. We can identify the most trusted users and then exploit their (track) records to recommend Web services. This simplifies the work needed to detect trust relations based on similar invocations or evaluations and minimizes the risk of using inconsistent information from spam users.

III. WSR-TRACK RECOMMENDATION SYSTEM

WSR-Track, the system we propose, consists of two main components:

- **Graph generation process:** Builds a heterogeneous multi-graph where the nodes represent users (resp.

services) and the edges illustrate track relationships among these users and between users and services.

- **Web services recommendation process:** Provides the most adequate Web services for a target user based on his/her history and his/her tracked relations.

In the following we will detail both processes.

A. Graph generation process

The process builds a heterogeneous users/services multi-graph by means of *ProgrammableWeb* track functions. A user can track (follow) either: Web services, mashups, searches or others users (see Figure 1). Hence, in following a user on *ProgrammableWeb*, one may infer Web services and mashups of interest. Based on these tracking functions, we can connect users and services with track relationships. Users (resp. services) represent the nodes while the track relations are the edges. We refer to this graph as the (*users/services'*) *track graph (TG)* (cf. Definition 1).

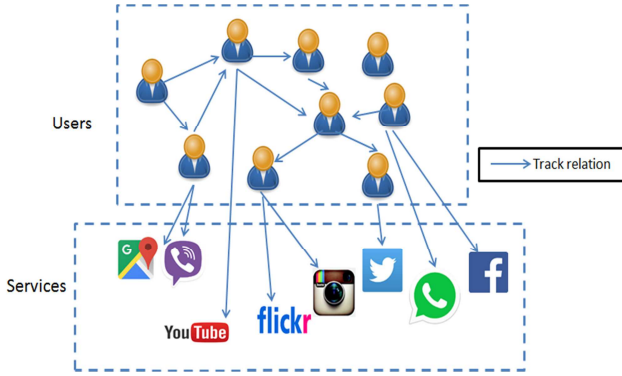


Figure 1. Users/services' track graph

Definition 1. Track Graph. $TG = \langle V, E \rangle$ is an oriented heterogeneous multi-graph where $V = \{v_1, v_2, \dots, v_m\}$ is a set of m registered users and services, and $E = \{(v_i, v_j) / \text{user } v_i \text{ tracks user } v_j \text{ and/or user } v_i \text{ tracks service } v_j\}$ is the set of oriented relations.

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.

B. Web services recommendation process

The output of this process is the recommendation of a set of relevant Web services to a target user based on his/her history and his/her tracked relations. This process is divided in two steps: *i*) The first step retrieves data from the track graph by exploiting the users' relationships; *ii*) The second step merges this data with other information such as the user's history and the quality of services (popularity, usage, etc.) to select the most relevant services.

The first step, considered as a mining step, involves two actions:

- *Neighborhoods detection*, which consists of detecting user's neighbors in the track users graph to extract their recent histories, i.e. previous Web services usages.
- *Association rules generation*, based on *Apriori* [8] to mine "valid" association rules. These rules express correlation amongst used services.

The second step, called recommendation step, consists of exploiting these association rules to select relevant Web services and to rank them according to their usage/popularity.

Figure 2 illustrates the main steps of WSR-Track recommendation system.

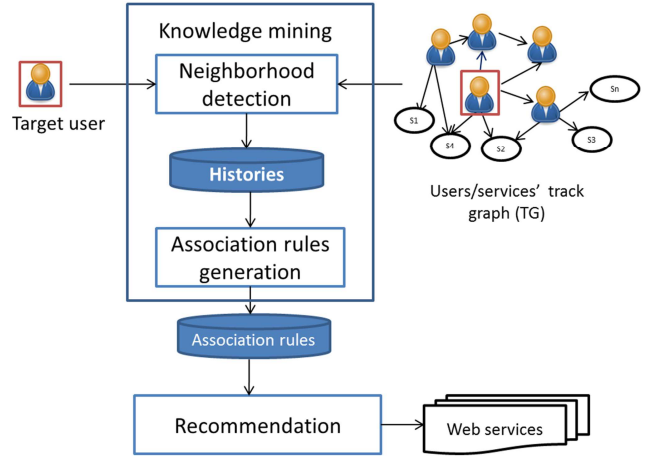


Figure 2. WSR-Track Recommendation Processes

In the following, more details will be given for the above mentioned steps.

1) Knowledge mining

This step includes mainly two actions: *neighborhood detection* and *association rules generation*.

a) Neighborhood Detection

The recommendation system takes into account relations between users in order to recommend services. If a user u_i follows a user u_j , it means that u_i is interested in services/mashups of u_j . Therefore, we recommend services to a user u_i , based on the history of his/her neighbors in the track graph (TG).

Once the list of neighbors is obtained from the TG, it is filtered in order to acquire the k most similar users. The similarity between users is measured according to the number of common services in their previous usages. 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).

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

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

b) Association rules generation

Association rules (AR) (cf. definition 2) are useful for analyzing and predicting users' behaviors. They play an important role in data analysis, product clustering, product recommendation, etc.

In order to determine the most used and the closest Web services to a user u_i , we will integrate the association rules mining method [8] into our approach. This method aims at discovering the relations of interest between two or more stored items. The idea is to find frequent items, associations, correlations, or causal structures from a data collection.

For sake of readability, we recall the definition of the main concepts used in the paper.

Definition 2. Association rules [8]. Given: (1) a list of items I , and (2) a set of transactions T where each transaction T is composed of a set of items ($T \subseteq I$), the goal is to find all association rules that express correlation between the presence of an item with the presence of a set of items.

For each association rule $R_i: X, Y \rightarrow Z$, we apply two metrics which are the *support*(R_i) and the *confidence*(R_i). The *support* is used to measure the quantity of population concerned by this rule while the *confidence* is used to evaluate its statistical significance.

Since our recommendation process should only rely on significant and "valid" associations rules, two thresholds, *minsupport* and *minconfidence*, will be used respectively for the support and the confidence of each rule. Hence, each association rule R_i with *support*(R_i) \geq *minsupport* and *confidence*(R_i) \geq *minconfidence* is extracted, while other associations will be ignored.

These two thresholds are given by the user according to the nature, type of the underlying data, and the objective of the analysis.

The association rules generation consists of two steps: i) frequent itemsets deduction, and ii) AR mining using *Apriori* [8]. As a first step, we define a frequent itemset with a minimum support \geq *minsupport*. All the items of an association rule should be included in the generated itemset i.e., if $\{AB\}$ a frequent itemset then $\{A\}$ and $\{B\}$ are both frequent itemsets.

In a second step, we will apply *Apriori* on the obtained frequent itemsets. This algorithm uses breadth-first search and a tree data structure to extract candidate itemsets efficiently. It generates candidate itemsets of length k from the itemsets of length $k - 1$. Then it prunes the candidates that have an infrequent subpattern. It determines all frequent itemsets among the candidates set of k -length itemsets and generates the association rules. Most confident rules will be applied to recommend services for users.

The confidence of each rule $R_i: S_1 \rightarrow S_2$ is measured as follows:

$$Confidence(R_i) = \frac{support(\{S_1, S_2\})}{support(\{S_1\})} \quad (2)$$

Where S_1 and S_2 are two Web services, and $\{S_1, S_2\}$ a frequent itemset that contains S_1 and S_2 , while $\{S_1\}$ is the frequent itemset containing only S_1 .


Neighbors' histories (transactions)

facebook, googleMaps, twitter

facebook,twitter

facebook,youtube

googleMaps,linkedIn

 For min support = 50% = **2 trans**,
and min confidence = 50%

Frequent Itemset

Support

{facebook}

75%

{googleMaps}

50%

{twitter}

50%

{facebook, twitter}

50%

- Support = Sup({facebook, twitter})=50%

- Confidence = $\frac{50}{70} = 66.6\%$

Twitter \rightarrow facebook has 50% support and 100% confidence

Figure 3. Example of association rules generation

Figure 3 describes the main steps for generating an association rule. Given four users' previous transactions, with a *minsupport* set to 50% and a *minconfidence* set to 100%, *Apriori* generated a valid association rule **$R_1: Twitter \rightarrow facebook$** .

2) Recommendation

The recommendation process (cf. Algorithm 1.) is based on the previously generated association rules. These rules are built up on frequent itemsets of services used by the target user and by his/her neighbors. To recommend a Web service, we need first to process these rules and then to rank them.

a) Association rules Processing

Starting from the list of valid rules being generated, a set of candidate Web services to recommend will be selected. This

set contains services involved in the consequences of each rule satisfying a *friendship* condition. A rule is considered to be *friendship* if and only if it encloses in its antecedents *at least* one service already used by the target user.

b) Ranking

The final step in the recommendation process consists on ranking the resulted services based on their popularity scores (cf. definition 3).

Definition 3. Popularity. The popularity of a service (Pops) denotes the number of previously recorded usages this service has been involved in. A popular service will have a high popularity.

Algorithm 1. WSR-Track recommendation

Input: TG: Track graph
 u_i : the target user
 Hu : the set of services histories
 S : set of services that can be recommended
Output: recommended services for u_i

```

Begin
Neigh( $u_i$ )  $\leftarrow$  Neighborhood_detection(TG,  $u_i$ )
 $Hu \leftarrow UH_{u_i}$  where  $i \in [1..|Neigh(u_i)|]$ 
 $AR \leftarrow$  Apriori ( $Hu$ )
For each association rule  $R_i (a \rightarrow b)$  in  $AR$ 
  If  $b$  is not in  $Hu$  then
     $S = S \cup b$ 
  End if
End for each
For each service  $s$  in  $S$ 
  Pops = popularity ( $s$ , Neigh( $u_i$ ))
End for each
Sort services in  $S$  according to Pops values
Return services in  $S$ 
End

```

IV. EVALUATIONS

The main goal of our experimentations is to evaluate the user's satisfaction with recommended services. These recommendations are made based on the tracks relationships between users (resp. services).

Our experiments have been conducted on a real-world dataset. We used Tanagra platform² for the generation of Association rules, java with SQL Server 2005. We used a dual Core cpu@2.20G PC with 4G RAM, under Windows 7.

A. Dataset

Firstly we give a description of our dataset. The experimental dataset comes from the public Web service registry where Web services, mashups, member profiles can be searched

through its own services. All the data are stored in a database of SQL Server 2005 and released as public dataset file on the Internet³.

From the all open data, we selected 1000 users who have more than 1 service in their histories of tracks, and 3138 services (include APIs and mashups) are involved by these users.

Our dataset is divided into two parts: a testing set covers 20% of most recent published services in the history of each user, and the remaining 80% is a training set.

B. Evaluation metrics

To evaluate the performance and the effectiveness of our recommendation approach, several statistic metrics measures have been applied. Let's note that PR is the set of pertinent recommended services, R the set of recommended services and P the set of pertinent services.

1) Precision

Recommendation precision refers to a ratio of correctly predicted (satisfying user) services to the number of all recommended services.

$$\text{Precision} = \frac{|PR|}{|R|} \quad (3)$$

2) Recall

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

$$\text{Recall} = \frac{|PR|}{|P|} \quad (4)$$

The recall and precision are standard metrics to discuss the relevance of the recommended services according to the user's satisfaction.

3) RMSE

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

$$\text{RMSE} = \sqrt{\frac{\sum_{u,s} (1 - r_s)^2}{N}}$$

Where N is the number of recommended services.

4) Hit-rank

The hit-rank [17] metric is considered to take into account the ranks of returned services.

$$\text{hit-rank} = \frac{1}{m \cdot |L|} \sum_{u \in U} \sum_{i=1}^h \frac{1}{p_i} \quad (5)$$

Where h is the number of relevant services occurring at the positions p_1, p_2, \dots, p_h within the recommendation list; m is the total number of the users and L is the length of the recommendation list ($|L| = 10$).

Hence, services that occur earlier in the top-10 list are weighted higher than the services that occur later in the list. In order to ensure that our results are not sensitive to the particular training test partitioning of each dataset, we performed 10 different runs for each of the experiments,

²<http://eric.univ-lyon2.fr/~ricco/tanagra/>

³<http://www.lsis.org/sellamis/Projects.html#WeS-ReG>

each time using a different random partitioning into training and test sets.

C. Experimental Setup

Our proposed system is mainly based on the *Apriori* where the number of generated rules depends on the values of both support and confidence parameters. In order to find the “best” parameters values for WSR-Track performance evaluation, we set the support and confidence configurations as follow:



Figure 4. Variation of the number of generated rules according to the confidence (support=0.5)

- **Support:** The number of generated association rules is inversely proportional to the support values, e.g. support=0.5 means that for a service to be considered as common, it must be used by 50% of the neighbors of the user. The higher is the support value, the more relevant rules will be generated. According to our experiments, all the support values beyond 0.6 can generate no more than one or two rules. For example, *Apriori* generates 200 association rules with a minimum support of 0.3. For the below experiments, we set the support value at 0.5.
- **Confidence :** The number of generated rules is inversely proportional to the confidence values (Figure 4).

For our experiments we set the value of the *support* at 0.5 and the *confidence* value at 0.9. We used a sample of 30 randomly selected users among 1000 initially selected ones, having 1-30 services in their histories. Each test is repeated 5 times to ensure that results do not depend on the used dataset.

D. Experimental results

Four groups of experiments were conducted to evaluate the performance of WSR-Track system. The first one intends to find out the impact of the neighborhood sizes on both precision and recall. The second one assesses the impact of the number of tracked services in users’ histories. The third group evaluates the effect of service popularity and the last

one compares our system to other recommendation approaches.

1) Neighborhood size effect

To assess the impact of the target user's neighborhood size on the performance of our system and the precision of the generated recommendations, we randomly selected 30 users that tracked between 1 and 30 services. We repeated our tests using different neighborhood size $\in [1,200]$ by an interval of 10. As illustrated in Figure 5 and

Figure 6, the number of neighbors affects the recommendations accuracy in terms of precision, RMSE and recall. When the neighborhood cardinality size increases, the recommendations provided by our system are more precise and the error rate decreases.

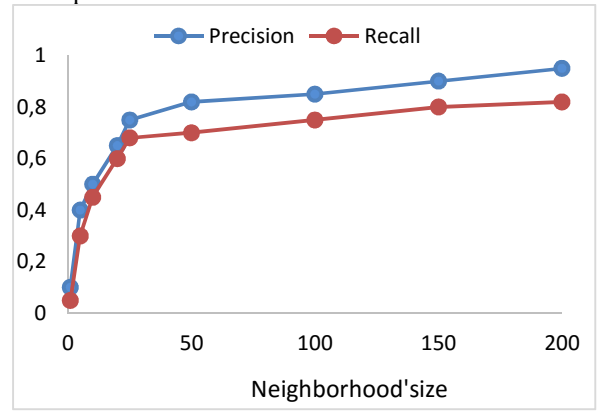


Figure 5. Impact of neighborhood' size

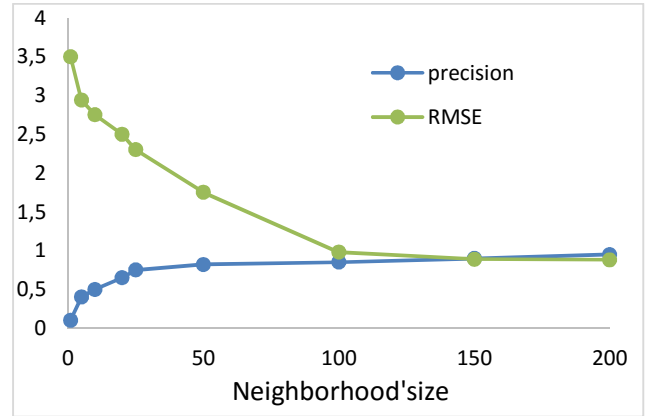


Figure 6. Variation of RMSE and precision

For example, more than 60% of recommended services are relevant for a user with 20 neighbors, while this number exceeds 80% of recommended services per 30 neighbors. It was observed that when the number of neighbors is less than 20, the precision value decreases. This result can be vindicated by the fact that with a high number of neighbors, *Apriori* mines several association rules leading to the improvement of the performance of the recommendation

process. Thus, neighborhoods' size may have a positive influence on the recommended services precision. We can conclude that users need to follow at least 20 users in order to get relevant recommended services.

Figure 7 shows that the neighborhood size brings a significant positive impact on the hit-rank metric values. From this, we concluded that we obtain more accurate Web services between the top-10 returned services of recommendation lists when the neighborhoods' sizes increase.

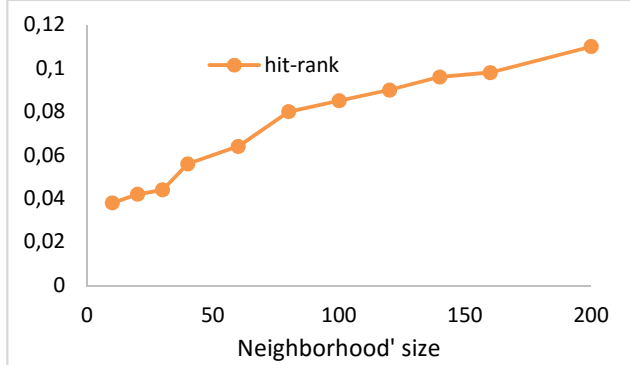


Figure 7. Impact of neighborhood size (hit-rank)

2) Histories size effect

We evaluated the effect of the users' histories size on the performance of our system. Figure 8 illustrates the precision and the recall values while varying the size of users' histories. It can be noticed that the variation of precision values is less important from 20 neighbors; therefore we can conclude that, users have to follow at least 20 users to get better recommended services.

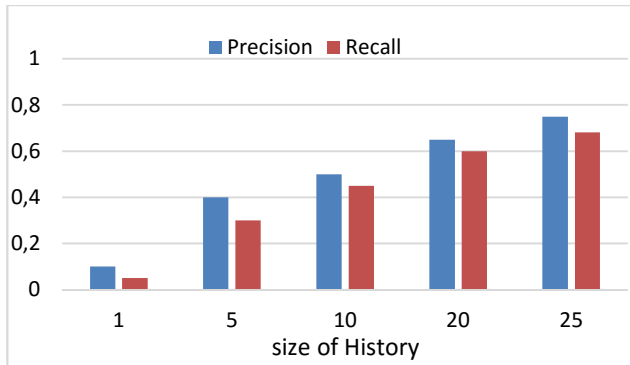


Figure 8. History size impact

3) Service popularity effect

We evaluated the impact of service's popularity on our recommendation system performance. Service's popularity is measured by its number of occurrences in the neighbors' histories.

For each user u , we identified the 5 most popular services in his/her neighborhood. We evaluated his satisfaction when he/she receives these services as recommendations. Figure 9

shows that a user's satisfaction increases when the recommended service is a popular one for his neighborhood. We conclude that the popularity of the service may have a great effect on the quality of recommendations.

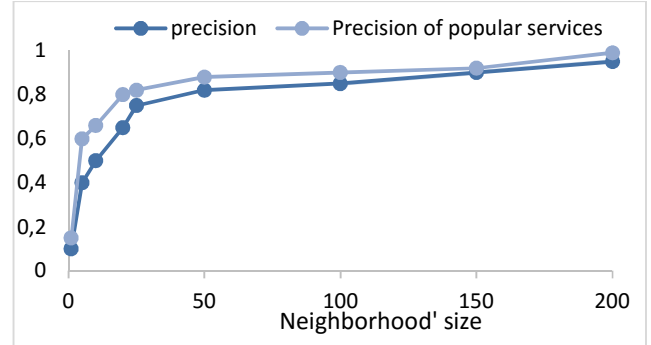


Figure 9. Service popularity effects on Precision and Recall

4) Comparison with others recommendations approaches

We compared the proposed approach WSR-Track with known recommendation approaches in the literature using the *librec* framework⁴ namely, i) *TrustSVD*[23] which is a trust-based matrix factorization technique recommendation system that analyzes the social trust data from real-world data sets. This approach considers both explicit and implicit ratings to recommend; ii) *Association Rules* [8](AR) 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.

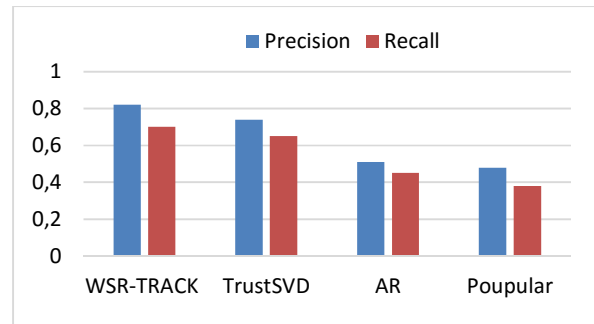


Figure 10. Comparison with other recommendations approaches

We fixed the neighborhood size to 20 neighbors and the users' histories size to 30. As illustrated in Figure 10, our recommendation algorithm, which is based on Track relationships, provides the best accuracy values compared to others approaches. TrustSVD that is the closest to our system gives good precision values due to the inclusion of trust relations between users and services. But as all rating

⁴ <http://www.librec.net/>

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 compared to TrustSVD and WSR-Track. 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, in our recommender system, the relations between users are established offline (track users graph) and updated periodically. Compared to existing CF methods for service recommendation, ours performs better in most cases. The neighborhood size and the histories size affect positively the accuracy of our approach.

V. CONCLUSION

Web service recommendation is still a challenging research issue due to the proliferation of available Web services in a dynamic environment and a variety of user needs. In this paper, we propose a graph-based recommendation method, which takes into account both users' history of service invocation as well as relationships among users (resp. services) based on track relations extracted from a subset of *ProgrammableWeb* services. We define a heterogeneous multi-graph of users and services that is exploited as follows: detection of neighbors, association rules generation and finally service recommendation. Experimental results show that our approach recommends more new Web services to the users.

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 weight edges in using similarity values between users based on their profiles in order to detect the most similar tracked users. We will improve our approach in taking into account users that have no tracking relationships and we will address the cold start problem as well.

REFERENCES

- [1] M. Eirinaki, M. D. Louta and I. Varlamis, "A trust-aware system for personalized user recommendations in social networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 4, pp. 409-421, 2014.
- [2] S. Deng, L. Huang, and G. Xu, "Social network-based service recommendation with trust enhancement," *Expert Systems with Applications*, 41(18), pp.8075-8084, 2014.
- [3] S. Deng, L. Huang, Y. Yin, and W. Tang, "Trust-based Service Recommendation in Social Network," *Appl. Math.*, 9(3), pp.1567-1574, 2015.
- [4] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "A Privacy-Preserving QoS Prediction Framework for Web Service Recommendation," In *Web services (ICWS)*, 2015 IEEE International Conference on pp. 241-248, 2015.
- [5] X. Zhang, K. He, J. Wang, C. Wang, G. Tian, and J. Liu, "Web Service Recommendation Based on Watchlist via Temporal and Tag Preference Fusion," In *Web Services* In 2014 IEEE international conference on Web services (ICWS), pp. 281-288. IEEE, 2014.
- [6] U. S. Manikrao, and T. V. Prabhakar, "Dynamic Selection of Web Services with Recommendation System," *International Conference on Next Generation Web Services Practices*, August 2005, Seoul, Korea.
- [7] A. Maaradji, H. Hacid, R. Skraba, A. Lateef, J. Daigremont, and N. Crespi, "Social-Based Web Services Discovery and Composition for Step-by-Step Mashup Completion," *ICWS*, pp. 700-701. IEEE Computer Society, 2011.
- [8] G. Piatetsky-Shapiro, "Discovery, analysis, and presentation of strong rules", in G. Piatetsky-Shapiro & W. J. Frawley, eds, 'Knowledge Discovery in Databases', AAAI/MIT Press, Cambridge, MA 1991.
- [9] X. Wang, Z. Wang, and X. Xu, "An Improved Artificial Bee Colony Approach to QoS-Aware Service Selection," 2013 IEEE 20th International Conference on Web Services, pp. 395-402, 2013.
- [10] Z. B. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-Aware Web Service Recommendation by Collaborative Filtering," *IEEE Transactions on Services Computing*, vol. 4, pp. 140-152, Apr-Jun 2011.
- [11] J. Cao, Z. Wu, Y. Wang, and Y. Zhuang, "Hybrid Collaborative Filtering algorithm for bidirectional Web service recommendation," *Knowledge and Information Systems*, pp. 1-21, 2012.
- [12] R. Andersen, C. Borgs, and J. Chayes, "Trust-based recommendation systems: an axiomatic approach," In *International conference on world wide Web*, pp. 199-208, 2008.
- [13] J. O'Donovan, and B. Smyth, "Trust in recommender systems", In *International conference on intelligent user interfaces*, pp. 167-174, 2005.
- [14] F. E. Walter, S. Battiston, and F. Schweitzer, "A model of a trust-based recommendation system on a social network," *Autonomous Agents and Multi-Agent Systems*, 16(1), pp. 57-74, 2008.
- [15] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43-52, 1998.
- [16] X. Su, and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques", *Advances in Artificial Intelligence archive*, 2009.
- [17] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, pp. 143-177, 2004.
- [18] M. Claypool, M. P. Le, M. Wased, and D. Brown, "Implicit interest indicators," In: *International Conference on Intelligent User Interfaces*, ACM Press pp. 33-40, 2010.
- [19] X. Chen, X. Liu, Z. Huang, and H. Sun, "RegionKNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation," *Web Services (ICWS)*, 2010 IEEE International Conference on , vol., no., pp.9,16, 5-10 July 2010.
- [20] Y. Jiang, J. Liu, M. Tang, and X. Liu, "An Effective Web Service Recommendation Method Based on Personalized Collaborative Filtering," *Web Services (ICWS)*, 2011 IEEE International Conference on , vol., no., pp.211,218, July 2011.
- [21] Q. Zhang, C. Ding, and C. Chi, "Collaborative Filtering Based Service Ranking Using Invocation Histories," *Web Services (ICWS)*, 2011 IEEE International Conference on , vol., no., pp.195,202, July 2011.
- [22] B. Yang, Y. Lei, D. Liu, and J. Liu, "Social collaborative filtering by trust," In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2747-2753, 2013.
- [23] G. Guo, J. Zhang, and N. Yorke-Smith, "Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 123-129, 2015.