Revista de Sistemas de Informação da FSMA
n. 13 (2014) pp. 21-29

SISTEMAS DE INFORMAÇÃO
http://www.fsma.edu.br/si/sistemas.html

TRILHA PRINCIPAL

Assessing the Stability and Robustness of Semantic Web Services Recommendation Algorithms Under Profile Injection Attacks

Pedro Henrique Grandin and Juan Manuel Adán-Coello

***Abstract* — Recommendation systems based on collaborative filtering are open by nature, what makes them vulnerable to profile injection attacks that insert biased ratings in the system database in order to manipulate recommendations. In this paper we evaluate the stability and robustness of collaborative filtering algorithms used to recommend semantic web services when subjected to random and segment profile injection attacks. We evaluated four algorithms: (1) IMEAN, that makes predictions using the average of the ratings received by the target item; (2) UMEAN, that makes predictions using the average of the rating made by the target user; (3) an algorithm based on the k-nearest neighbor (k-NN) method and (4), an algorithm based on the k-means clustering method. The experiments showed that the UMEAN algorithm is not affected by the attacks and that IMEAN is the most vulnerable of all algorithms tested. Nevertheless, both UMEAN and IMEAN have little practical application due to the low precision of their predictions. Among the algorithms with intermediate tolerance to attacks but with good prediction performance, the algorithm based on k-NN proved to be more robust and stable than the algorithm based on k-means.**

***Keywords* — Profile injection attack; collaborative filtering algorithms; semantic web services.**

I. Introduction

In SOA (*Service Oriented Architecture*) architectures, loosely coupled services allow for the creation of flexible and dynamic business processes and agile applications that can include different organizations and computational platforms [1]. Among the central problems for the creation of those processes, deriving from the composition of new services from existing ones, is the discovery of services that can fulfill the users' requirements and interests, whether they are persons or software agents.

The discovery of a service is made by matching algorithms that seek in descriptions repositories which of the announced services fulfill the requisites of the potential user. Service discovery architectures are usually based on the WSDL [2] and UDDI [3] standards and have a series of limitations that make it difficult to find relevant services for the users.

Those limitations derive mainly from the informal descriptions of services' functionalities and capacities, usually composed in natural language, generally without the usage of a vocabulary that is common to the provider and the consumer of the service. Semantic Web Services are a recent approach that tries to overcome those limitation combining the technology of Web Services with elements of the so called Semantic Web [4,5].

Research in the field of Web Services has focused on service matching algorithms, either semantic or not. Nevertheless, recent work showed that the most challenging issue when we want to provide the user with the desired service is service selection from a list of candidates instead of the matching process itself.

The main approaches for service selection include content based filtering, collaborative filtering, reputation systems, P2P systems and reference systems [6]. Among them, collaborative filtering methods, which are the focus of this paper, are one of the most relevant.

Collaborative filtering is based on the premise that users with profiles or interests in common usually seek similar items. The fundamental idea is to use the options of similar users made when choosing related items to recommend items to a user [7].

The recommendation problem can then be reduced to estimate ratings for items that were not accessed by a user and recommend those with the highest estimated ratings.

Even though recommendation systems can be seen as part of a search engine, they can also be used autonomously in

Pedro Henrique Grandin (e-mail: pedro.hg@puccamp.edu.br) and Juan Manuel Adán-Coello (juan@puc-campinas.edu.br) are with the Computer Engineering Faculty at Pontifical Catholic University of Campinas (PUC-Campinas).

order to suggest items to users as they become available, without the need of explicit and constant searches.

Collaborative filtering is widely used in Web based systems to recommend different types of items, including books, music and movies.

The development of new algorithms that increase the precision and efficacy of recommendation systems or create models that explain the reasons behind a recommendation is an issue that has increasing academic and commercial interest. An example of this interest was the *Netfilx prize*, created by Netflix, an online movie distribution company, that in 2006 announced a one million dollar award to those who developed a movie recommendation algorithm that was 10% more accurate than the Cinematch algorithm that was then used by the company. The prize was receive in 2009 by the BellKor's Pragmatic Chaos team, created during the contest by the merge of different competing groups that gathered their efforts to created the winning algorithm ensemble [8].

Recommendation systems based on collaborative filtering are by nature open, given that they are based on the ratings made by a user community, making them vulnerable to profile injection attacks. This type of attack consists in inserting biased profiles in the system database allowing the attackers to manipulate the recommendations. The goal of these attacks may also be to promote items (push) or obfuscate them (nuke), having as target a user or a group of users [9].

In this context, analyzing the robustness and stability of recommendation algorithms becomes a relevant issue. When analyzing the robustness, we measure the performance of the system before and after the attack to verify how the attack affected the system as a whole. In the stability analysis we intend to verify the deviation from the values predicted by the system for the attacked items.

Previous works proposed algorithms for the recommendation of Web Services with semantic markup [10, 11]. The evaluations performed, without considering the possibilities of attacks, showed that the algorithms have good performance when evaluated for precision of the recommendation, especially in situations where the user-item matrix that stores the recommendations made by the user community is sparse. Given that, the goal of the research we report here is to analyze the stability and robustness of those algorithms when subject to profile injection attacks.

The next sections of this paper are organized as follows. In section II we present the algorithms for collaborative filtering of semantic web services whose stability we intend to study. In section III we will characterize in greater detail the profile injection attacks that are the subject of this work. In section IV we present and discuss the methods used to verify the robustness and stability of the algorithms under consideration and the experiments performed. In section V we present some related work and in section VI we wrap up the paper making some final considerations.

II. Collaborative Filtering of Semantic Web Services

A. *Semantic Web Services*

The algorithms discussed in this paper intend to recommend Web Services semantically annotated according to the service annotation ontology OWL-S, which is based on the ontology description language OWL, a standard from W3C which is based on first-order logic [12]. The OWL-S ontology is made of three main parts: the service profile, used to advertise and discover services, the process model, which provides an accurate description of the workings of the services and a grounding, which provides details on how to interoperate with a service through messages.

Most matching systems for services described with OWL-S use only the service profile, which defines the semantic of the service signature, that is, the required inputs and the produced outputs. The profile also allows describing the preconditions to be satisfied, so that the service can be executed, and the results expected from its execution. This information is usually known by the acronym IOPE (*inputs, outputs, preconditions, effects*). With this information, we can use logic reasoning methods to determine the similarity degree between two services.

The algorithms used in this work to verify the degree of similarity between two services use only service signatures, that is, the descriptions of inputs and outputs. This certainly allows for false positives in some matching attempts, but this has not shown itself as a relevant problem. The input and output parameters are associated with concepts from domain ontologies, such as Person, Doctor, Vehicle, Motor-Vehicle and not with basic types (int, char, real) or mere character sequences. Hence, even though we cannot guarantee that we are comparing services that perform very different things on inputs that are semantically equal, this possibility is greatly reduced. Naturally, we would prefer to have matching algorithms that are not limited to the comparison of inputs and outputs, but this is not a concern for this research and should not change a lot the experimental results.

OWL-S, just like OWL, is based on description logic. We know that formal logic has limited expression abilities. For instance, description logic does not allow representing precisely structured objects arbitrarily interconnected. In order to partially overcome this limited representation ability of OWL-S, this work used the hybrid semantic Web Services matching mechanism implemented in the OWLS-MX system [13]. This mechanism allows comparing between two semantic Web services and establishes four

degrees of semantic similarity using logic reasoning and a value of syntactic similarity using information retrieval methods.

B. Collaborative Filtering

Collaborative filtering algorithms can be used to predict the value a certain user (the target user) will attribute to an item (the target item) he has not yet evaluated. The prediction is performed based on the rating history of the user community for all items under consideration.

We can use some very simple algorithms such as IMEAN, that evaluates the unknown rating values as equal to the average of the ratings performed by all users for the target item, or UMEAN, that estimates the rating of an item as equal to the average of the ratings performed by the target user. These algorithms are easily implemented and have low execution cost, but are not very precise, serving, in general, only as a comparison benchmark for the rating of other more elaborate algorithms, like the two other evaluated in this paper: the first one is based on the k-nearest neighbors algorithm (k-NN) and the second on the k-means clustering algorithm.

C. Rating Predictions using K-nn

The k-NN algorithm is an algorithm based on memory and as such it predicts the rating an user will give to an item directly from the ratings of the k users more similar to this user.

In the algorithms described in this paper, the similarity between two users u and v, $s_{u,v}$ is calculated using the Pearson correlation coefficient (PCC), extended to consider also similar services, according to equation (1).

$$s_{u,v} = \frac{\sum_{s \in S_u, t \in S_v} (f_{u,s} - \overline{f}_u)(f_{v,t} - \overline{f}_v)}{\sqrt{\sum_{s \in S_u} (f_{u,s} - \overline{f}_u)^2} \sqrt{\sum_{t \in S_v} (f_{v,t} - \overline{f}_v)^2}} \quad (1)$$

In equation (1), S_u is the set of services evaluated by u and S_v is the set of services accessed by v. $\overline{f}_u$ and $\overline{f}_v$ are the average values of the ratings performed by the users u and v, respectively; among the services evaluated by v, t is the services that is most similar semantically or syntactically to s (service accessed and evaluated by u), given a minimum degree of similarity. When both services evaluate the same service, then s and t are the same.

It is important to notice that the PCC does not weight differently similar users that evaluated either a small or a very large set of items in an equivalent way. In the same way, equation (1) does not differentiate between users whose similarity is due to the rating of the same items and those who evaluated similar items. These are issues not dealt with by this research but are interesting enough to merit further attention in future work.

TABLE I
EXAMPLE OF AN ITEM-USER MATRIX

User	Item 1	2	3	4	5	6	Similarity with user 1
1	5	2	3	3		?	1.00
2	2		4		4	1	- 1.00
3	3	1	3		1	2	0.76
4	4	2	3	1		1	0.72
5	3	3	2	1	3	1	0.21
6	4	3		3	3	2	0.94

Table I shows an example of a user-item matrix for generic items (not necessarily services), without displaying information that allow to verify the semantic or syntactic similarity between two items. In this table, the ratings vary from 1 to 5. An empty position indicates that the user who is referred to by that line did not evaluate the service referred by that column. The matrix stores the values of the ratings performed by six users to six generic items (books, movies, Web Services, etc). The column to the right of the matrix shows the similarity between user 1 and all other users, calculated using the Pearson correlation coefficient.

If we only consider the closest user to user 1 (k=1), the algorithm based on the k-NN method would predict that user 1 would attribute to the target item (item 6) a value close to the one attributed by the user 6 (the most similar to user 1). Knowing the similarity among users, we can estimate the rating a user would make of a specific target item if similar users have evaluated this service or similar ones. For that, we define the neighborhood V or the user u with respect to the service s as being made of the k users most similar to u that accessed service s or services similar to s. Once we have built the neighborhood, the prediction of the rating that the user u would make of the service s, $f_{u,s}$, is given as the weighted average of all ratings of service s, or similar services, given by the users in V, as seen in equation (2).

$$f_{u,s} = \overline{f}_u + \frac{\sum_{v \in V} s_{u,v} (f_{v,t} - \overline{f}_v)}{\sum_{v \in V} |s_{u,v}|} \quad (2)$$

In equation (2), as in equation (1), t is the service accessed by v that is most similar to the service s (accessed by u), given a minimum similarity threshold. If V is empty, $f_{u,s}$ is predicted as equal to $\overline{f}_u$ (the average of all ratings made by u).

Applying equation (2) to the data in Table I, and considering k=1, the value predicted for the rating of item 6

by user 1 would be:

$f_{1,6}$ = 13/4 + (0.94 * (2 - 15/5))/ 0.94 = 2.25

D. Rating Predictions Using K-means

Literature points out that memory-based collaborative filtering algorithms such as k-NN tend to have high precision but low scalability given that their predictions are made directly from the available data. This takes a high processing time for each prediction to identify the neighborhood of the target user for each item (service) under consideration. As an alternative, in model-based algorithms the predictions are not made directly from the users closer to the target user, but based on a model built previously based on available data.

In this paper we also analyze a collaborative filtering algorithm based on the k-means clustering method. This algorithm clusters similar user profiles into groups and makes the predictions based on their centroids. In this context, the user profile is defined by the ratings a user made of the available items (a line in the user-item matrix). For instance, in the user-item matrix given by Table I, the profile of user 1 is given by the n-uple (5,2,3,3, Ø, Ø), where Ø indicates that the corresponding item was not evaluated.

The algorithm works as follows: initially k points (user profiles as defined by the n-uples that contain the ratings attributed by the users to the items) are chosen as centroids of k groups, where k is a previously defined parameter. Next, there is an assignment and an update step until the algorithm converges. In the assignment step each point (profile) is associated with the groups with the closest centroid, while in the update step the group centroids are updated to the average of the points associated to the group. The algorithm converges when the centroids become stable, that is, they do not change in the update step.

In the implementation under study, equation (1) is used to calculate the distance between a user and a group centroid. Defined the groups, equation (2) is then used to predict the rating a user would make of an item, using a neighborhood made of the centroids of the groups closer to the target user, instead of the ratings (n-uples) of the closer users. Since the number of group centroids in the considered neighborhood is usually a lot smaller than the number of users that make up a neighborhood in the algorithm based on k-NN, the algorithm based on k-means has a much small computational time, as proven in experiments non reported in this paper.

III. PROFILE INJECTION ATTACKS

In a profile injection attack, the attacker inserts biased profiles in the user-item matrix of the recommendation system.

Table II shows the user-item matrix of Table I with the insertion of a fake profile, user 7. Once this insertion was made, the closer user to user 1 is the attacker, with a similarity degree between them of 0.98.

TABLE II
EXAMPLE OF AN ITEM-USER MATRIX WITH AN ATTACK PROFILE (USER 7)

Usuários	*Itens*						*Similaridade com o usuário 1*
	1	*2*	*3*	*4*	*5*	*6*	
1	5	2	3	3		?	*1.00*
2	2		4		4	1	*- 1.00*
3	3	1	3		1	2	*0.76*
4	4	2	3	1		1	*0.72*
5	3	3	2	1	3	1	*0.21*
6	4	3		3	3	2	*0.94*
7	**4**	**2**		**3**	**3**	**5**	***0.98***

In these conditions, if we use an algorithm such as k-NN, with k=1, the estimated rating value for item 6 by user 1 would be equal to 4.85. This can be verified by considering that using equation (2) we have the following values:

- $f_{u,s} = f_{1,6}$
- $\overline{f_u}$ = *(5+2+3+3)/4 = 13/4* ; average of all ratings performed by user *u=1*
- $s_{u,v}$ = 0.98 ; degree of similarity between users u=1 and v=7, according to the table;
- $f_{v,t}$ = 5 ; *v=7* and *t = 6,* because user 7 accessed service 6;
- $\overline{f_v}$ = (4+2+3+3+5)/5= 17/5 ; the average rating performed by user v=7.

Hence, *f1,6 = 13/4 + (0.98 * (5 - 17/5))/ 0.98 = 4.85*

Even though it is not a current practice to use such a small neighborhood, the example above intends to emphasize the possible negative impact a successful attack can achieve.

A. Types of Profile Injection Attacks

Profile injection attacks can be characterized by for sets of items [14]:

- A unit set, containing the target item i_t;
- A set of selected items with characteristics determined by the attacker, I_S;
- A set of items for filling, I_F, usually chosen randomly, and
- A set of non evaluated items, $I_\emptyset$.

Types of profile injection attacks are defined by the methods used to identify the selected items, I_S, the proportion of filling items, I_F, and the way to determine associated ratings to each one of these set of items and to

the target item.

In this paper we present experiments seeking to reproduce two of the most representative types of profile injection attacks, the random attack and the segment attack.

In the random attack, I_S is empty and the items in I_F are filled randomly according to a normal distribution around the average rating of all items in the database and i_t is filled with the maximum value that can be attributed in a rating. The knowledge required to create a random attack is very small, given that the general average of the ratings of a recommendation system usually can be determined empirically without great difficulty by an external observer and, in many cases, is directly available in the system. Nevertheless, the cost of executing this attack can be high, because it is necessary to calculate specific random ratings for each item in the attack profile.

In the segment attack, the items in I_S define a category or segment. These items receive maximum rating when trying to promote an item (push) or minimal when trying to obfuscate it (nuke). If those items are, for instance, travel services, the items in I_S would correspond to services in this category. This attack is usually remarkably effective if its target belongs to the segment under consideration. The values in I_F are filled with the minimal rating and i_t with the maximum one. The literature highlights that this attack is quite effective and requires little knowledge to be executed.

IV. STABILITY AND ROBUSTNESS OF THE RECOMMENDATION ALGORITHMS

As stated in the introduction, the goal of this research is to analyze the stability and the robustness of algorithms for recommending semantically marked Web Services, when subjected to profile injection attacks. In order to reach this goal, we performed an experimental research that executed the following steps:

i. Selection of a database of semantically marked Web Services;
ii. Creation of a dataset for training and testing the algorithms including user profiles characterized by the ratings performed to a selection of the services included in the previous step;
iii. Selection of users and items to be targets of the attack;
iv. Application of the recommendation algorithms without the presence of attack profiles and measurement of the normalized mean absolute error (NMAE) of the predictions made by the algorithms;
v. Injection of the attack profiles;
vi. New application of the recommendation algorithms and measurement of NMAE and other metrics that measure the success of the attacks;
vii. Analysis of the obtained results;

These steps and its results are presented and discussed with more details in this section. Initially, subsection A will present the metrics used to measure the success of the attacks. Afterwards, subsection B will describe the experiments performed and subsection C will present and analyze the results obtained in the experiments.

A. Rating Metrics

The efficacy of the profile injection attacks on specific items can be evaluated by an examination of the prediction shift induced by the attack, the hit ratio of the attack and the influence of the attack over the mean absolute error of the predictions performed.

Given a user u and an item i, the prediction shift, $\Delta_{u,i}$, is calculated using equation (3). In this equation, $P'(r_{u,i})$ is the predicted value for the rating that user u would make of the item i after the attack and $P(r_{u,i})$ represents the value predicted before the attack.

$$\Delta_{u,i} = P'(r_{u,i}) - P(r_{u,i}) \quad (3)$$

A positive deviation indicates that the attack was successful in improving the item rating. Nevertheless, even a high increase in the prediction shift does not guarantee its recommendation. It is possible that other items are also affected by the attack or that the item had initially a very low prediction, so that even a high deviation does not include it in the top recommended items.

In order to evaluate if items attacked were effectively recommended we can use the Hit Ratio metric (HR) that measures the efficacy of the attack oven an item. The hit ratio for item i is calculated by equation (4), where $H_{u,i}$ will be equal to 1 if an item i appears in the list of the N items recommended to the user u (*top-N recommendations*). If it does not appear, its value will be 0; U_T is the set of target users.

$$HR_i = \sum_{u \in U_T} H_{u,i} \Big/ U_T \quad (4)$$

The effect of the attack considering all the predicted ratings can be evaluated by the Mean Absolute Error *(MAE)* or by the Normalized Mean Absolute Error *(NMAE)*, metrics that are generally used to evaluate the precision of predictions made by recommendation algorithms. The MAE is obtained by the differences between the predicted and actual values of the ratings, as shown in equation (5), where $r_{u,i}$ is the real rating given by user u to item i, $r'_{u,i}$ is the algorithm predicted rating for this item and N the number of predictions made by the algorithm.

$$MAE = \frac{\sum_{u,i} \left| r_{u,i} - r'_{u,i} \right|}{N} \quad (5)$$

MAE can be normalized in order to make it independent of the rating scale used, giving origin to the Normalized Mean Absolute Error (NMAE), calculated by the equation (6).

$$NMAE = \frac{MAE}{\sum_{u,i} r_{u,i} \Big/ N} \quad (6)$$

B. Experiments Description

The experiments performed intended to evaluate the impact of profile injection attacks of the random and segment type over the collaborative filtering algorithms IMEAN, UMEAN, *k-NN* and *k-means* presented in section II. The performance of those algorithms without attacks, particularly when the user-item matrix is sparse, is analyzed in [11].

One of the main difficulties to evaluate algorithms for the recommendation of Semantic Web Services is the need for a public base of services with semantic markings. The closest we have to this is the OWLS-TC[1] collection, which was used in this work. It was created to evaluate the performance of matchmaking algorithms for Semantic Web Services described according to the OWL-S 1.1 service annotation ontology. In the experiments descrived we used version 2.2 of this base which includes 1004 Web Services of various domains.

In order to perform the experiments we created 50 users, 20 with the profile "tourist" and 30 with the profile "student". The items evaluated included 108 services, related to the subjects "cars", "food", "books", "hotels", "cameras", "publications", "surf" and "movies". Each user evaluated 18 services that would be characteristic to his profile. A tourist, for instance, would be more interested in hotels than students and, therefore, there is a possibility of finding more hotel ratings in the tourists group than in the students' one.

We used parameters similar to those used in [14] to create the attacks. This way, we inserted in the user-item matrix an amount of attack profiles equal to 15% of the number of users, resulting in the injection of eight attackers whose goal was to promote an item (push). We chose five users and four services randomly as targets, corresponding approximately to 5% of the users and 3% of the total of items. In the attack profiles, the filling items (I_F) corresponding to 6% of the total of services. Filling items are chosen randomly after specifying the amount of profiles to be created. In the segment attack, we defined a segment (I_S) characterized by three items. The segment chosen for the attacks was "food", because it is presumed that both students and tourists would be interested in this type of service, so that this choice should affect both group of users.

[1] http://projects.semwebcentral.org/frs/?group_id=89

C. RESULTS AND DISCUSSION

The experiments described in this section were repeated 10 times, so that the values presented for the NMAE, hit ratio and prediction shift are the average of the values obtained in those predictions.

Influence of the attacks on NMAE

Table III presents the NMAE for the algorithms under study before and after the attacks of the segment and random types.

TABLE III
NMAE OF THE EVALUATED ALGORITHMS

	No attack	*Random attack*	*Segment attack*
UMEAN	*0.33111*	*0.33111*	*0.33111*
IMEAN	*0.33064*	*0.48232*	*0.47835*
k-NN	*0.27781*	*0.28239*	*0.28415*
k-médias	*0.29305*	*0.30015*	*0.30193*

As we can see in Table III, UMEAN is not affected by the attacks. This makes sense, given that the addition of profiles does not affect this algorithm, because an item rating is simply the average of the ratings made by the user himself, and no consideration is given to the ratings made by the attackers.

For the IMEAN algorithm, we expected the attacks to have a large impact on the NMAE and the hit ratio, given that its predictions are based on the ratings of the whole user community. Table III and Fig. 1 show that this expectation was fulfilled. In Table III we can observe that for IMEAN the random attack increased the NMAE in 45.9% (from 0.33064 to 0.48232) and the segment attack in 44.7% (from 0.33064 to 0.47835).

In the case of the k-NN and k-means algorithms, the attacks resulted in visible increases in the NMAE, eventhough they are much less expressive that those verified for IMEAN. The random attack increased the NMAE in 1.6% for k-NN (from 0.27781 to 0.28239) and 2.4% for k-means (from 0.29305 to 0.30015); on the other hand, the segment attack increased the NMAE by 2.3% for *k-NN* (from 0.27781 to 0.28415) and 3.0% for k-means (from 0.29305 to 0.30193).

Hit Ratio and Prediction Error

Fig. 1 presents the hit ratio of the attacks for the IMEAN algorithm. The x-axis indicates the number of items to

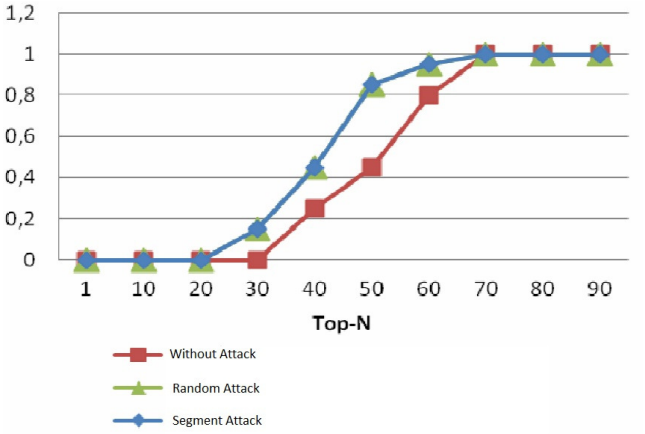

Fig. 1. Hit ratio when using the IMEAN algorithm

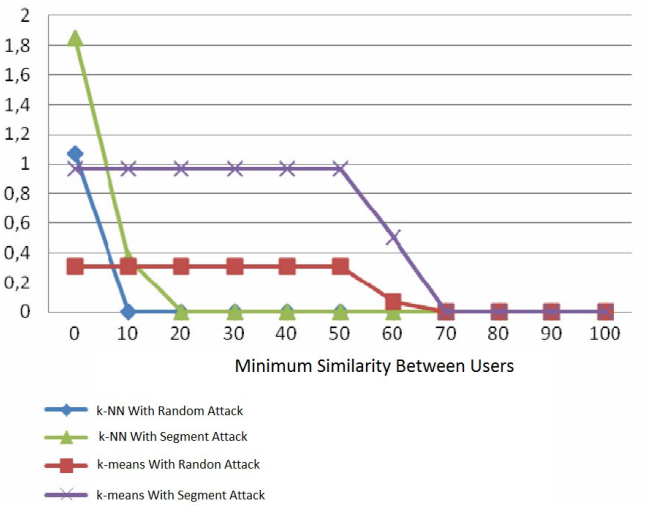

Fig. 2. Prediction error of the k-nn and k-means algorithms as a function of the minimum similarity between users

recommend for the user, the top-N (the N items with the highest predicted ratings), while the y-axis represents the hit ratio. The closest to 1 the hit ratio, the highest the percent of attacked items that will be recommended to the users. A hit ratio equals to one indicates that all target items were recommended.

We can see in Fig. 1 that both the random and the segment attack presented similar results for IMEAN. This happens because this algorithm uses the ratings of all users for the target item, making both attacks achieve similar effects. For top-N up to 20 items and above 70, the effects of the attacks were not perceptible, but for top-N between 20 and 70, the hit ratio was quite high, implying in high success rates in the inclusion of the attacked items in the users' recommendation lists.

Both attacks generated a prediction shift equals to 2.70966 (not shown in the figures), which is quite high when compared to the values observed for the k-NN and k-means algorithms, presented next.

In order to measure the prediction shift for the k-NN and k-means algorithms, we varied the minimum similarity value among users (k-NN) and among users and group centroids (k-means), used in the moment of choosing the k closer neighbors and then applying equation (2). Besides, we established as 2 the number of clusters used during the tests with k-means (this value of k was chosen after a series of preliminary adjustment experiments).

Fig. 2 presents the results obtained. It can be seen that the prediction shifts for the k-NN and k-means algorithms are perceptibly lower than those verified for UMEAN and IMEAN. It can also be noticed that starting at a 20% of similarity between users for the k-NN algorithm and 70% for k-means, the prediction shift remains constant at zero, which means that starting from those values, the similarity required was enough to exclude attack profiles from the item rating estimation.

In the analysis of the hit ratio of the attacks over k-NN and k-means, we used similarity values that resulted in a high prediction shift and that were close to the value where the prediction shift falls to zero. The values adopted were 5% to the k-NN algorithm and 50% for the k-means algorithm.

It can be seen in Fig. 3, which shows the hit ratio for k-NN, that the random and segment attack had some success on the top 60 and 70, but the segment attack had an elevated hit ratio for top 20 and 50.

Fig. 4 shows that for k-means we only observe a remarkable hit ratio for top 50 and 60 and once again there is some higher importance for the segment attack.

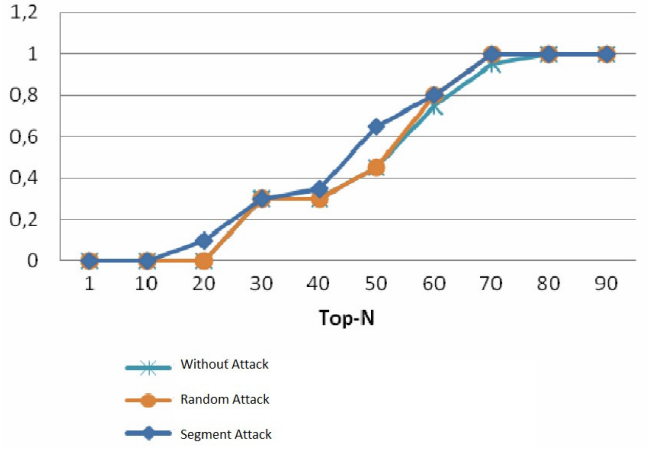

Fig. 3. Hit ratio for *k-NN*

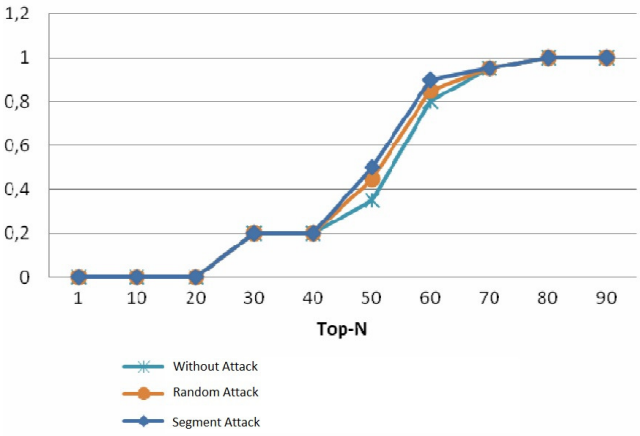

Fig. 4. Hit ratio for *k-means*

V. RELATED WORK

The random attack was originally proposed by Lam and J. Riedl [25]. The knowledge required to perpetrate this attack is small, but its execution cost can be quite high, given that it is necessary to attribute ratings for each item in the attack profile. On the other hand, as these authors showed and our results confirmed, this attack is not very effective.

The segment attack was introduced by Mobasher et al. [14]. The authors showed and our experiments confirmed that it is possible to execute successful attacks of this type against recommendation systems based on collaborative filtering without the need of having a substantial knowledge on the system or on the users.

Several papers, including [14][17][20][21], showed that profile injection attacks can damage a lot the robustness of recommendation systems. This lead several authors to search for recommendation systems more robust and stable using a variety of mechanisms, including attacker influence thresholds [18], dynamics rating sequences instead of static sets of rating profiles [22] and event to offer monetary incentives for other evaluators to correct the system distortions, whether or not they were provoked by attacks [23]. Several authors propose using strategies to detect attacks, using, among others, unsupervised or semi-supervised learning mechanisms [16, 24] and statistical models [19].

In this paper, the term stability is associated to measuring the dynamic of the system predictions when it is under external attack. When measuring stability, we evaluate the change in the system predictions for the items under attack. Some authors, such as Adomavicius and Zhang [15], studied a different aspect of stability, called internal consistency, which represents the consequences of internal inconsistencies of the algorithms of the recommendation systems. For those authors, a stable recommendation algorithm offers consistent prediction as time goes by, assuming that the new ratings that become available are according to the previous system predictions.

VI. CONCLUSION

The experiments performed to evaluate the stability and robustness of the algorithms IMEAN, UMEAN, k-NN and k-means for the recommendation of semantic Web Services, when subject to profile injection attacks, showed that the UMEAN algorithm is not affected by attacks and the IMEAN algorithm is the most vulnerable to those attacks. Nevertheless, those two algorithms are used only to provide a performance baseline for the analysis of the other algorithms, since both present low precision, particularly when the user-item matrix is sparse, as shown in [10] and [11].

For the other two studied algorithms, we concluded that k-NN arrives at a prediction shift close to zero at quite low similarity rates between users (20%) while in k-means this is only possible for higher values (70%).

On the other hand, in the experiments presented to analyze the hit ratio, k-means presents better results when compared to k-NN when k-NN employs a base user similarity value much lower than the one used in k-means. With high similarity values between users, k-NN presents better results also in this issue. In spite of those results, when choosing an algorithm to use in a concrete application, one must consider that, according to the literature, k-means tends to be more scalable than k-NN. In terms of attack efficacy, experiments showed that the segment attack is more effective than the random attack.

The algorithms were also evaluated considering the usage of semantic and syntactic similarity between Web Services when computing the similarity between users with equation (1) and when predicting the value of a rating with equation (2). The experiments showed that this has not influenced in a meaningful way the results, which is coherent with the results found in [10], where it was verified that using semantic similarity between services affect in a relevant way the precision of the algorithms only when the user-item matrix is sparse, which does not occur in the experiments describe in this paper.

Future works should include experiments to evaluate the behavior of the algorithms when subject to attacks considering data sets of higher dimensions and user-item matrices with sparse data.

ACKNOWLEGMENTS

We would like to thank the anonymous reviewers and the journal editor, Prof. Dr. Ricardo Linden, for the careful reviews and valuable corrections, recommendations and comments that have contributed to improve this paper's clarity and precision.

REFERENCES

[1] H. Chesbrough and J. Spohrer, "A research manifesto for services science," Commun. ACM, vol. 49, no. 7, pp. 35–40, 2006.

[2] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1, 2001," At http://www. w3. org/TR/2001/NOTE-wsdl-20010315, 2001.

[3] L. Clement, A. Hately, C. von Riegen, and T. Rogers, "UDDI Version 3.0." OASIS, 19-Oct-2004.

[4] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," Intelligent Systems, IEEE, vol. 16, no. 2, pp. 46–53, 2005.

[5] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic Web," Scientific American, vol. 284, no. 5, pp. 28–37, 2001.

[6] R. M. Sreenath and M. P. Singh, "Agent-based service selection," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 1, no. 3, pp. 261–279, 2004.

[7] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in Artificial Intelligence, vol. 2009, pp. 1–19, 2009.

[8] S. Lohr, "The contest that shaped careers and inspired research papers," CHANCE, vol. 23, no. 1, pp. 25–29, 2010.

[9] J. J. Sandvig, B. Mobasher, and R. Burke, "A survey of collaborative recommendation and the robust-ness of model-based algorithms," IEEE Data Engineering Bulletin, vol. 31, no. 2, pp. 3–13, 2008.

[10] J. M. Adán-Coello, Y. Yuming and C. M. Tobar, "A Memory-based Collaborative Filtering Algorithm for Recommending Semantic Web Services. Revista IEEE América Latina, v. 11, p. 795-801, 2013.

[11] J. M. Adán-Coello, C. M. Tobar and Y. Yuming , "Improving the Performance of Web Service Recommenders Using Semantic Similarity". Submetido.

[12] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, and others, "OWL-S: Semantic Markup for Web Services," 2004.

[13] M. Klusch, B. Fries, and K. Sycara, "OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, no. 2, pp. 121–133, Apr. 2009.

[14] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," ACM Transactions on Internet Technology (TOIT), vol. 7, no. 4, p. 23, 2007.

[15] G. Adomavicius and J. Zhang, "Stability of recommendation algorithms," ACM Transactions on Information Systems (TOIS), vol. 30, no. 4, p. 23, 2012.

[16] R. Bhaumik, B. Mobasher, and R. D. Burke, "A clustering approach to unsupervised attack detection in collaborative recommender systems," in Proceedings of the 7th IEEE international conference on data mining, Las Vegas, NV, USA, 2011, pp. 181–187.

[17] G. Shani and A. Gunawardana, "Evaluating recommendation systems," Recommender Systems Handbook, pp. 257–297, 2011.

[18] P. Resnick and R. Sami, "The influence limiter: provably manipulation-resistant recommender systems," in Proceedings of the 2007 ACM conference on Recommender systems, 2007, pp. 25–32.

[19] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in Proceedings of the third ACM conference on Recommender systems, 2009, pp. 149–156.

[20] S. Ray and A. Mahanti, "Strategies for effective shilling attacks against recommender systems," in Privacy, Security, and Trust in KDD, Springer, 2009, pp. 111–125.

[21] B. Van Roy and X. Yan, "Manipulation robustness of collaborative filtering," Management Science, vol. 56, no. 11, pp. 1911–1929, 2010.

[22] B. Van Roy and X. Yan, "Manipulation-resistant collaborative filtering systems," in Proceedings of the third ACM conference on Recommender systems, 2009, pp. 165–172.

[23] R. Bhattacharjee and A. Goel, "Algorithms and incentives for robust ranking," in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007, pp. 425–433.

[24] J. Cao, Z. Wu, B. Mao, and Y. Zhang, "Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system," World Wide Web, vol. 16, no. 5–6, pp. 729–748, 2013.

[25] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in Proceedings of the 13th international conference on World Wide Web, 2004, pp. 393–402.