Improving the Performance of Web Service Recommenders Using Semantic Similarity

Juan Manuel Adán-Coello, Carlos Miguel Tobar, Yang Yuming

Faculdade de Engenharia de Computação, Pontifícia Universidade Católica de Campinas (PUC-Campinas)
Campinas, SP, Brasil

{juan,tobar}@puc-campinas.edu.br; lyonbrcn@gmail.com

Abstract: This paper addresses issues related to recommending Semantic Web Services (SWS) using collaborative filtering (CF). The focus is on reducing the problems arising from data sparsity, one of the main difficulties for CF algorithms. Two CF algorithms are presented and discussed: a memory-based algorithm, using the k-NN method, and a model-based algorithm, using the k-means method. In both algorithms, similarity between users is computed using the Pearson Correlation Coefficient (PCC). One of the limitations of using the PCC in this context is that in those instances where users have not rated items in common it is not possible to compute their similarity. In addition, when the number of common items that were rated is low, the reliability of the computed similarity degree may also be low. To overcome these limitations, the presented algorithms compute the similarity between two users taking into account services that both users accessed and also semantically similar services. Likewise, to predict the rating for a not yet accessed target service, the algorithms consider the ratings that neighbor users assigned to the target service, as is normally the case, while also considering the ratings assigned to services that are semantically similar to the target service. The experiments described in the paper show that this approach has a significantly positive impact on prediction accuracy, particularly when the user-item matrix is sparse.

Keywords: Collaborative filtering, Recommender systems, Semantic similarity, Semantic Web Services, Sparse data.

1. INTRODUCTION

Service-Oriented Computing (SOC) is a new computing paradigm that uses services as building blocks to accelerate the development of distributed applications in heterogeneous computer environments. SOC promises a world of cooperating services where application components are combined with little effort into a network of loosely coupled services for creating flexible and dynamic business processes that can spread over many organizations and computing platforms [1]

Among the key challenges for the effective use of Web services is the discovery of services that meet the functional and non-functional requirements of its users and that take into account their preferences [2]. In Web service discovery systems, three entities can typically be distinguished: the service requester (a user or a program), the service provider and the service registry. Entities seeking services make service requests to the registry. In the registry, the description of the service requested is compared with the descriptions of services advertised by service providers, using a matching algorithm, to identify whether there are services that meet the request. If the matching is successful, the registry provides the description of identified service instances to the requester, including the necessary details for their invocation.

Architectures for service discovery, usually based on the WSDL specification [3], have serious limitations arising from the service description technology and matching algorithms used. These limitations are due, in part, to the use of informal descriptions of service functionality and capability, written in natural language, usually lacking a common vocabulary for the service requester and provider. Semantic Web Services (SWS) and Linked Services are recent approaches that try to overcome these limitations by combining Web services technology with elements of the Semantic Web [4][5].

In SWS discovery architectures, advertised services are described using service annotation ontologies in addition to WSDL parameters and operation names. These ontologies define a semantic model for the description of a Web service from several perspectives, including functionality, execution flow and invocation details. They define a set of attributes for service capability description, the most common being the so-called IOPE (Inputs, Outputs, Preconditions and Effects). Service annotations, in accordance with a service annotation ontology, use concepts contained in domain ontologies instead of non-standardized words, which are more commonly used in conventional non-semantic approaches.

Domain ontologies describe the terminology and the relationships between terms of a specific domain using an ontology language such as OWL or RDFS [6][7]. Each ontology language has its own unique expressive power, but all can model, at the minimum, hierarchies of concepts and roles of concepts, such as properties, attributes and relationships. When performing a search, the characteristics of the desired service, such as inputs and outputs, are specified by terms that represent ontology concepts. Matchmaking algorithms based on logical inference can then seek matches for the request parameters, taking into account the parameters of the available services. For each match found, a value that characterizes the matching degree (similarity) is computed. Finally, the identified services are returned to the requester in descending order of matching degree.

Search algorithms for semantic Web services present good results when the user is able to adequately describe the desired service. However, this is not always the case, and a request for a service cannot correspond fully to the intentions of the requester. For example, there may be a published service that partially matches the request and accomplishes the intentions of the requester, or the opposite scenario could also conceivably occur [8]. As the number of available services on the Web increase, this problem worsens. Currently, as pointed out in [9], one of the most challenging issues in Web service provision is not the matchmaking process but the selection of good services for a target user.

In addition, as the number of available Web services grows, there may be a lot of interesting available services that users are not aware of, and that they therefore will not take the initiative to request. Additionally, in the context of mobile and ubiquitous computing, it is unreasonable to assume that a user is constantly searching for interesting services available at the user locale. In this context, it is desirable to have a recommender system capable of identifying and of proactively recommending potentially interesting services to the user in the right situation.

A web service recommender can also be very valuable to proactively deal with failures and to recover to service workflows that have partially failed and in dynamic composition scenarios, provided the services and the recommender can deal with semantic markup [10] [11].

The recommendation problem can be reduced to an issue of estimating ratings for items that have not been used before by a user; items with higher estimated ratings are as a consequence recommended to the user.

Recommenders are usually implemented using filtering algorithms classified into three main categories, depending on how the recommendations are performed: (1) Content-Based algorithms (CB) filter and recommend items that are similar to others the user has accessed in the past; (2) Collaborative Filtering (CF) algorithms filter and recommend items based on the preferences of other users with similar tastes and preferences; knowledge-based (KB) recommenders use knowledge about users and items to generate a recommendation. It is also frequent to find hybrid systems that combine methods taken from two or more of the previous categories of recommenders [12] [13].

Content-based recommenders have their roots in the information retrieval field and were successfully implemented in domains where the items to be recommended are described through textual information. These systems are, however, limited by the features that are explicitly associated with the items. They are also limited to recommended items that are similar to those already rated by the user (over specialization). A particularly difficult task for this type of algorithm is to deal with new users, because new users have to rate a sufficient number of items before the system can understand their preferences and start making useful recommendations.

CF algorithms do not have some of the abovementioned shortcomings of content based algorithms. Since they employ the user's ratings, they can deal with any kind of content and recommend any type of item, even items that are dissimilar to those accessed in the past.

However CF systems have their own challenges, including coping with sparse data and scaling with increasing numbers of users and items. Several structural difficulties related to sparse data may be encountered, including the cold start problem, the reduced coverage problem and the neighbor transitivity problem. The cold start problem occurs when new users or items are inserted into the system. New items cannot be recommended until they are rated by some users, and, in turn, new users are unlikely to receive good recommendations because they lack a rating history. The reduced coverage problem occurs when the number of ratings is very small compared with the number of items in the rating database. In this situation, the system may be unable to generate recommendations for such users. The neighbor transitivity problem occurs when users with similar tastes do not have rated items in common and thus cannot be identified as similar.

Knowledge-based recommender systems avoid some of the drawbacks of content and CF system since their recommendations do not depend on a base of user ratings. Their main drawback is the well known knowledge acquisition bottleneck.

Algorithms for CF, the primary focus of this paper, can be further classified into two main categories: memory-based and model-based. Memory-based algorithms construct a neighborhood of users who have similar ratings to the target user using directly the available data. In this circumstance, the ratings of neighbors are used to predict how a target user will rate an item he has not yet accessed. Model-based techniques employ available rating data to learn a model to make predictions, usually using data mining or a machine learning algorithm. Then the model is used to make predictions for target items, instead of using raw rating data, as is done with memory-based algorithms.

When comparing memory-based and model-based CF algorithms it is usually accepted that memory-based algorithms are easy to implement and have higher prediction accuracy, particularly for dense datasets. Model-based algorithms are, in turn, more scalable and less vulnerable to profile injection attacks [12].

In the recent past, recommender systems have been built for recommending different types of items in diverse domains, including CD, Web pages, books, news, movies and courses. However, research on Web service recommendation is in its preliminary stages and usually focuses on predicting service QoS (Quality of Service) parameters [14], which is a very limited way of capturing user interest [15].

In this paper, we present algorithms for constructing Web service recommender systems aimed at reducing the problems arising from sparse data. The proposed approach combines CF algorithms with logical inference to determine the semantic similarity between services, and between users. The rationale behind this approach is that if two users have not rated a common set of services but have rated similar services, these ratings can still be an indication of user similarity and therefore contribute to reduce the effects of data sparseness.

The remainder of the paper is organized as follows: section 2 presents memory and model-based CF algorithms for Semantic Web services recommendation; section 3 discusses the experimental set up used to evaluate the algorithms and the results that were obtained; section 4 presents related work; and, finally, section 5 concludes the paper by pointing out our main results and directions for future work.

2. CF ALGORITHMS FOR SEMANTIC WEB SERVICE RECOMMENDATION

In this section, variations of two recommender algorithms that exploit semantic similarities among web services are presented. Their performance will be compared in Section 3.

Instances of user feedback[1] are stored in a user-item matrix, represented as a set $T \subseteq U \times S \times F$, where U

[1] In this paper we use the terms 'feedback', 'score' and 'rating' as synonyms.

$= \{u_1, u_2, \ldots, u_m\}$ is the set off all users, $S = \{s_1, s_2, \ldots, s_n\}$ is the set of all rated services, $F = \{f_1, f_2, \ldots, f_m\}$ is the set of instances of feedback related to services in S and collected from the users in U. Each $f_u \in F$ is an n-dimensional vector over the space of all instances of user feedback, i.e, $f_u=(f_{u,s1}, f_{u,s2}, \ldots, f_{u,sn})$ where $f_{u,sj} \in [0..1]$ is the feedback given by user u to service s_j. If a service was not rated its feedback is represented as ϕ (null).

Although the collaborative filtering algorithms described in this section are independent of the notation used to describe service semantics, when they allow for the measurement of the level of semantic similarity among two services, a prototype for services described using OWL-S was implemented for the validation of the algorithms. OWL-S is an upper ontology that specifies that a service can be described by at most one service model, and a grounding must be associated with exactly one service [16]. OWL-S is a W3C recommendation based on the W3C standard OWL, an ontology language for the Semantic Web with formally defined meaning [6].

Computing Service Similarity

In our prototype implementation, the degree of similarity between OWL-S services is computed using a hybrid semantic service matching algorithm described in [17] that takes advantage of both logic-based reasoning and IR techniques.

If R represents a request for a service and S a service registered in the service database, the semantic matching algorithm computes the following matching degrees:

- Exact match (S exactly matches R) - The I/O (Input/Output) signature of S perfectly matches request R with respect to the logic-based equivalence of their formal semantics.
- Plug-in match (S plugs into R) - All input parameter concepts of S match more specific ones in R. In addition, S is expected to return more specific output data.
- Subsumed match (R subsumes S) - This matching degree is weaker than plug-in matching. The output of S is more specific than requested by R as before, but the constraint of immediate output concept subsumption is relaxed to arbitrary output concept subsumption.
- Subsumed-by match (R is subsumed by S) - The output of S is slightly more general than requested (direct parent output concepts).
- Nearest-neighbor match (S is the nearest neighbor of R) - It is checked if the degree of text similarity, *SynSim(S,R),* between the input and output concepts of S and R is greater than or equal to a defined syntactic similarity threshold α. This degree is computed as the averaged syntactic similarity of the serialized input and output concepts of S and R, *according* to a given similarity metric. A set of concepts is serialized by means of their expansion through the ontology implemented and by the conjunctive concatenation of the results into one unstructured text document, including only logical operators and primitive components of the basic vocabulary that is present in the ontological terminology. In the case of vector-space-based text similarity measurement, these documents are represented as weighted keyword vectors based on a term-weighting scheme.
- Fail (S does not match with R) - None of the above matching degrees was obtained.

Memory-based Feedback Prediction with K-NN

This recommendation algorithm is based on the construction of neighborhoods of similar users. The neighbors' ratings can then be used to make predictions for unrated items. A neighborhood is constructed comparing the similarity of each pair of existing users using the Pearson Correlation Coefficient (PCC).

Two variants of the algorithm were implemented. In the first, named PCC, the similarity between two users u and v, *sim(u,v),* is computed as shown in Eq. (1), where $S_{uv} = \{s \mid f_{u,s} \neq \phi \text{ and } f_{v,s} \neq \phi\}$ is the set of services that both users, u and v, have rated, $f_{u,s} \in [0..1]$ is the feedback given by user u to service s and $\overline{f_u}$ and $\overline{f_v}$ are the averages of the instances of feedback given by users u and v, respectively.

$$sim(u,v)= \frac{\sum_{s\in S_{uv}} (f_{u,s}-\overline{f}_u)(f_{v,s}-\overline{f}_v)}{\sqrt{\sum_{s\in S_{uv}} (f_{u,s}-\overline{f}_u)^2}\sqrt{\sum_{s\in S_{uv}} (f_{v,s}-\overline{f}_v)^2}} \quad (1)$$

In Eq. (1), if users u and v have not rated items in common it is not possible to compute their similarity. Also, if the number of common items that were rated is very low, the computed similarity may be unreliable.

In the second variant of the algorithm, named PCC-SS (PCC with similar services), it is not required that users u and v rate the same services to compute their similarity as it takes into consideration the ratings of similar services. The similarity between services is computed using the semantic matching algorithm presented in the previous subsection.

PCC-SS computes the similarity between two users, u and v, using Eq. (2). In that equation, t is the service rated by v that is most similar to s (rated by u), respecting a minimum threshold of similarity δ. When both users have rated the same service, s and t represent the same service (the similarity between s and t is 1).

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

The similarity between two users, *sim(u,v)*, computed using Eq. (1) or Eq. (2), ranges from −1 to 1. A value of 1 implies a line that describes the relationship between feedback $f_{u,s}$ and $f_{v,s}$ given from users u and v, respectively, for service s (or a similar service), with all data points (instances of feedback) lying on the line where $f_{v,s}$ increases as $f_{u,s}$ increases. A value of −1 implies that all data points lie on the line where $f_{v,s}$ decreases as $f_{u,s}$ increases. A value of 0 implies that there is no linear correlation between the various instances of feedback. In our implementation only *sim(u,v)* values higher than 0 were considered relevant.

The feedback a user u would give to a service s that he has not yet rated can be estimated using the ratings that neighbor users assigned to that service. Having a neighborhood V, the feedback user u would give to service s, $f_{u,s}$, can be predicted using two variants of the weighted average of all neighbors' ratings, as shown in Eq. (3) and Eq. (4).

$$f_{u,s} = \overline{f}_u + \frac{\sum_{v \in V} sim(u,v)(f_{v,s} - \overline{f}_v)}{\sum_{v \in V} |sim(u,v)|} \quad (3)$$

For the result $f_{u,s}$ in Eq. (3), hereafter named WAAR (Weighted Average of All Ratings), the neighborhood V is formed by the k most similar users to u that rated service s.

$$f_{u,s} = \overline{f}_u + \frac{\sum_{v \in Vss} sim(u,v)(f_{v,t} - \overline{f}_v)}{\sum_{v \in Vss} |sim(u,v)|} \quad (4)$$

For the result $f_{u,s}$ in Eq. (4), hereafter named WAAR-SS (Weighted Average of All Ratings with Service Similarity), the neighborhood Vss is formed by the k most similar users to u that rated service s or a service t that is semantically similar to s. If V *or* Vss is empty, respectively in Eq. (3) or (4), $f_{u,s}$ is made equal to $\overline{f_u}$.

Model-based Feedback Prediction with K-means

Memory-based filtering algorithms tend to be more accurate than model-based algorithms, but the latter are more scalable and less vulnerable to profile injection attacks [18]. Considering that the number of available services in the Web is continuously increasing, and that in the context of Web-based open collaborative recommenders the likelihood of attacks is not negligible, model-based recommender algorithms can be good alternatives to memory-based algorithms, provided that their accuracy is acceptable

We describe in this section a model-based CF algorithm for semantic Web services that uses the k-means clustering method and the concept of semantic service similarity.

The k-means method is used to partition a set of points or observations into clusters. If we consider that $f_u \in F$ defines the profile of user u, where f_u is the vector of instances of feedback given by user u for the available services, the k-means algorithms can be used to cluster users with similar profiles. Once the clusters are defined, their centroids can be interpreted as aggregated profiles of the users in the clusters as done in [19].

The clustering algorithm works as follows. Initially k points (f vectors) are randomly chosen as the initial cluster centroids, after which an assignment step and an update step are repeated until the algorithm converges. In the assignment step, each point is assigned to the cluster with the closest centroid. In the update step, cluster centroids are updated to the mean of the points assigned to the cluster. The algorithm converges when the centroids no longer change.

In the assignment step, the distance between a point and a cluster centroid is computed using the PCC or the PCC-SS (Eq. 1 and Eq. 2, respectively). Following the assignment step, the update step computes a new centroid $f_c=(f_{c,s1}, f_{c,s2}, \ldots, f_{c,sn})$ for each cluster c. The new centroid vector is the mean of the user profiles assigned to cluster c. That is, $f_{c,si}$, for $i = 1$ to n, is computed by Eq. (5).

$$f_{c,si} = \frac{1}{|c|} \sum_{u \in C} f_{u,si} \quad (5)$$

When applying Eq. (5), if some $f_{u,si}$ is equal to ϕ (meaning that user u has not rated service s_i), the average score of the items rated by u is instead used.

When the algorithm converges, each cluster centroid is seen as an aggregation of the user profiles in their respective cluster. User instances of feedback for unrated services are then estimated using Eq. (3) or Eq. (4), taking into consideration the neighborhood formed by the k clusters (represented by their centroids) most similar to the target user profile (represented by his feedback vector).

3. EXPERIMENTAL EVALUATION

The purpose of this section is to compare the performance of the algorithms presented on section 2.

The lack of public rating datasets is a major difficulty when validating recommender systems for Web services. To circumvent this difficulty, researchers usually adapt popular datasets constructed to recommend other types of items. For example, [20] use the Movielens[2] dataset and consider that a movie in the dataset represents a Web service. The evaluation of the algorithms we propose, adds an additional level of difficulty because we need a dataset of user ratings for semantic Web services.

In this context, an alternative is to synthesize a dataset that matches the properties of the target domain and task [21]. Following this approach we created a synthetic user-item matrix that can be used to provide some insights into the behavior of the implemented algorithms and serve as a proof of concept.

We used services from the OWL-S *Service Retrieval Test Collection* - OWLS-TC[3], version 2.2, a collection of 1004 Web services from several domains, specified according to the OWL-S ontology.

In the experiments, two groups with 50 users each were defined. Each user rated 56 services from the following four categories: cars, cameras, hotels and surf. Service ratings were set according to a base feedback defined for each pair (user_group, service_category). Each feedback was added to a value that varies from -1 to 1 according to the normal distribution.

The main objective of the experiments was to analyze the behavior of the proposed algorithms considering dense and sparse data scenarios. These scenarios were simulated by progressively hiding a number of service ratings from the algorithms: the 56 service ratings for each user were progressively reduced in steps of 10 until only 6 ratings were available for each user. After each removal step, the values of the removed scores were estimated using the algorithms previously discussed, with and without taking into consideration similar services, following which the average error of the predictions was computed. The experiments for each removal scenario were repeated 10 times and the results averaged. The time needed to compute the similarities between services was not taken into consideration because the computations were performed before running the experiments.

The prediction performance of the algorithms was measured using the Mean Absolute Error (MAE) and the Normalized Mean Absolute Error (NMAE), defined by Eq. (6) and Eq. (7), respectively.

[2] http://www.grouplens.org/node/73

[3] http://projects.semwebcentral.org/projects/owls-tc

$$MAE = \frac{\sum_{u,s} \left| p_{u,s} - f_{u,s} \right|}{N} \quad (6)$$

$$NMAE = \frac{MAE}{\sum_{u,s} f_{u,s} / N} \quad (7)$$

In Eq. (6), $p_{u,s}$ denotes the predicted feedback that user u will give to service s, $f_{u,s}$ denotes the actual (hidden) feedback that user u gave to service s, and N is the number of predicted instances of feedback. Lower values for MAE and NMAE indicate better prediction quality. A MAE or NMAE equal to zero corresponds to an ideal scenario, where all predictions are equal to the actual instances of feedback.

Evaluating the K-NN Memory-based Feedback Prediction Algorithm

In the experiments described in this section two services are considered similar if their matching degree is Exact, Plug-in, Subsumes, Subsumed-by or Nearest-neighbor with a threshold α of 0.8.

Two simple estimation schemes, the item-mean and the user-mean algorithms, were also implemented to be used as baselines. The item-mean (IMEAN) algorithm estimates the score for an item (a service) as the mean of the scores the target item received from all users that rated it. The user-mean (UMEAN) algorithm estimates the score for an item as the mean of the scores the target user gave to the items he rated.

When applying Eq. (3) or Eq. (4) (WAAR and WAAR-SS), the neighborhood used to estimate a score is defined by users with a degree of similarity to the target user that is greater than or equal to 0.8, as computed by Eq. (1) or Eq. (2) (PCC and PCC-SS). When setting this similarity threshold, we have to consider that if it is too low users with low similarity can be considered neighbors, negatively affecting the accuracy of the algorithm. On the other hand, if the threshold is very high it is possible that no neighbors will be found, making it impossible to predict feedback from the target user-service pair.

As can be observed in Figure 1, the prediction error when using the PCC and WAAR (without using service similarity) is significantly lower than when the IMEAN and UMEAN algorithms are used. In other words,

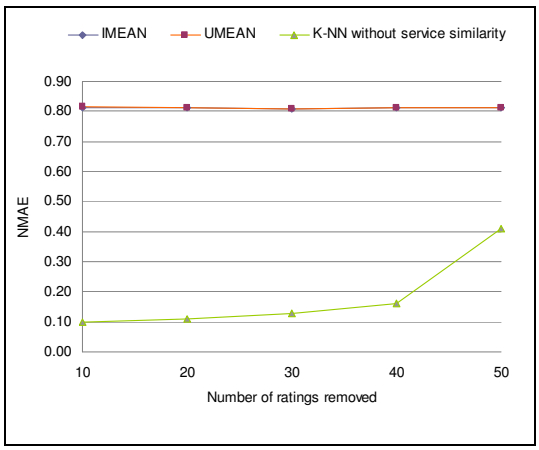

Figure 1. Prediction accuracy of IMEAN, UMEAN and k-NN without service similarity

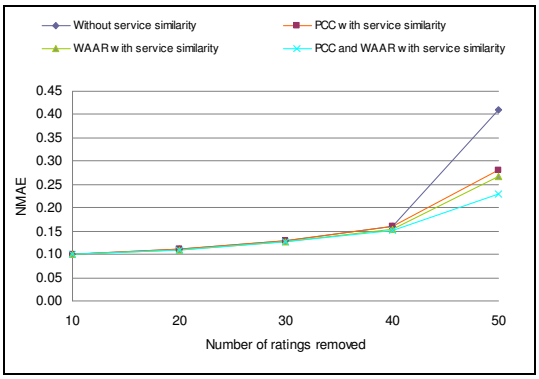

Figure 2. Impact of service similarity on the accuracy of the KNN-based prediction algorithm

considering a neighborhood of similar users to predict user feedback is better than using raw user or item averages.

Figure 2 shows that considering service similarity increases the prediction performance to an even greater extent. This happens when service similarity is used only to compute the PCC-SS (Eq. (2)) for the purpose of finding a neighborhood, or to estimate scores with WAAR-SS (Eq. (4)). Using service similarity both to compute the PCC-SS and the WAAR-SS produces even more accurate predictions. These results can be explained as follows. When the PCC is computed without taking into consideration service similarity, several similar users are not identified because the PCC equation correlates only users that rated a common set of services. When service similarity is taken into account, users who rated similar services are also taken into consideration, increasing the neighborhood and, as a consequence, the accuracy of the algorithm. In addition, using service similarity to predict a rating (WAAR-SS) contributes to increase the accuracy because it allows more scores to be considered when calculating the predictions. This happens because instead of only considering service scores that the target user and their similar users rated, scores for similar services are also included.

Figure 2 also shows that the effects of considering service similarity are not significant when a small amount of scores is removed, but are more dramatic when the amount of removed scores increases, that is, when the user-item matrix becomes sparser. As shown in figure 2, when 50 out of 56 scores are removed, the NMAE is equal to 0.23 if service similarity is considered in both the PCC and WAAR, while when it is not considered in any of the methods it rises to 0.41, an increase of 78%.

Evaluating The K-means Model-based Feedback Prediction Algorithm

Using the same scenarios from the previous section, experiments were conducted to evaluate the performance of the prediction approach based on k-means. One of the important parameters for this algorithm is the number of clusters, k. If k is too small user profiles with little similarity are clustered together, reducing the accuracy of the algorithm; on the other hand, if k is too high the scalability of the algorithm (one of its main expected advantages over the k-NN based algorithm) can be negatively affected. In the experiments presented in this section k was set to 8, a value chosen after some

preliminary tests demonstrated that it is a good choice for the data set used.

The neighborhood used to predict a feedback to a target user is formed by the cluster centroids that have a degree of similarity to that user (computed using the PCC and PCC-SS) greater than or equal to 0.8.

As can be observed in figure 3, the k-means prediction algorithm without service similarity has a prediction error significantly lower than that which is obtained when applying the IMEAN and UMEAN algorithms, except when the number of available scores is very low (when 50 out of 56 are removed). Under such circumstances, the small number of available user profiles prevents the construction of representative user groups, severely affecting the prediction accuracy of the algorithm. Under such sparse data conditions, the use of service similarity accounts for an appreciable increase in accuracy. As already verified for the k-NN algorithm, the best results are observed when service similarity information is used for computing both the PCC and the WAAR. These results can be explained in the same manner as done for the k-NN algorithm: when running the algorithm without service similarity information, several similar users are not identified as such and are not clustered together, because only users that rated the same set of services can be considered similar; when service similarity is taken into account, it is also possible to identify similar users among those users that rated similar services. In addition, when computing the WAAR, the use of service similarity information contributes to increase the accuracy because it allows for the consideration of more scores to calculate a prediction.

Figure 3 shows that when 50 out of 56 scores are removed, characterizing a situation of scarcity of evaluations, using service similarity for computing the PCC and WAAR accounts for a NMAE of 0.32, while when this information is not used the NMAE rises to 0.89, an increase of 178%.

Comparing the K-NN and the K-means Prediction Algorithms

The literature says that memory-based prediction algorithms, like those based on the k-NN, often have greater accuracy than model-based algorithms, such as those based on the k-means, but model-based algorithms are more scalable because they require less memory and are faster. Figure 4 confirms the first clause of the previous sentence. However, it is worth noting that the k-means algorithm with service similarity is more accurate than the k-NN one without service similarity.

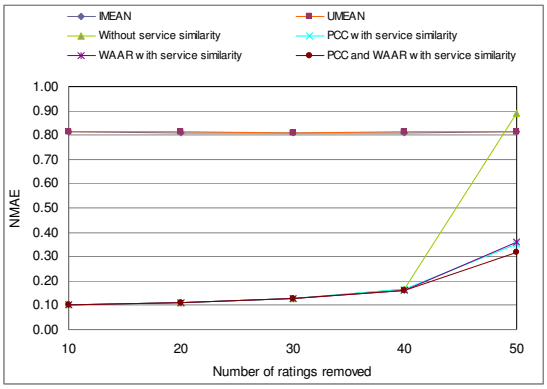

Figure 3. Impact of service similarity on the prediction performance of the k-means algorithm

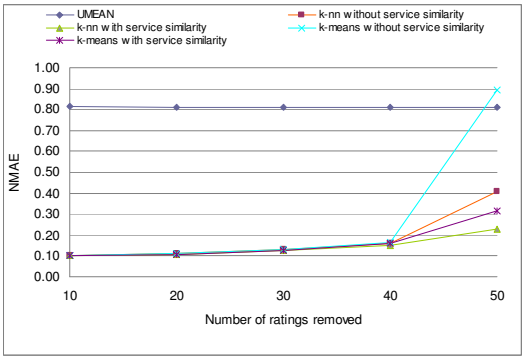

Figure 4. Comparing the prediction performance of k-NN and k-mean salgorithms

The lower accuracy of the k-means algorithm with respect to k-NN can be explained by the fact that the k-means method uses cluster centroids and not the profiles of similar users to predict the scores. Profiles are grouped into clusters based on the similarity of each profile to a cluster centroid; thus a poorly chosen centroid directly influences the quality of the cluster. In the implementation described, the initial eight centroids were chosen randomly among the available profiles. The particularly bad result for the k-means algorithm when many scores are removed and similar services are not considered can be explained by the difficulty in finding similar users to group together when data is sparse.

Figure 5 shows the time required by the algorithms to predict the removed scores when using a notebook with an Intel® Core™ Duo 1.66 GHz processor and 2 GB of RAM. Regarding the k-means algorithm, the required time for score predictions with already created clusters is shown. Under these conditions, the run time is lower for the k-means algorithm, particularly when the user-item matrix is dense. This result was expected because a high number of profiles are considered in the computation of the PCC and the WAAR when using the k-NN method, while only a small number of cluster centroids are used when applying the k-means method.

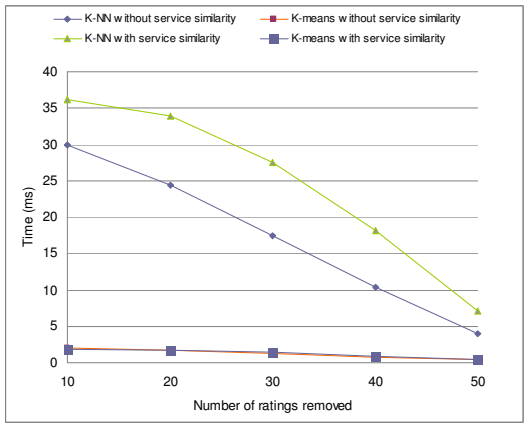

Figure 5. Run-time of the k-NN and k-means algorithms for score prediction

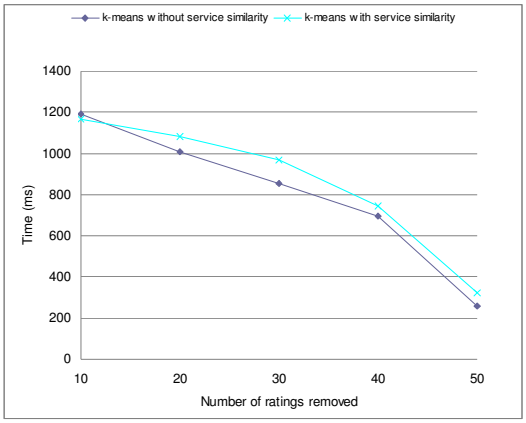

Figure 6. Computation time needed to construct the clusters in the k-means algorithm

The run-time for the k-NN algorithm tends to increase sharply as the amount of users and services increases. Under such conditions, the lower accuracy of the k-means algorithm can be acceptable provided that its run time is satisfactory. Figures 5 and 6 can be used to demonstrate this point. Figure 5 shows that the run time needed to predict the scores is lower for the k-means algorithm when the users' profiles are already grouped. However, as can be seen in Figure 6, the time needed by the k-means algorithm to cluster the users is very high when compared to the time needed by the k-NN algorithm to predict scores. That means that the k-means prediction algorithm would be suitable when the user-item matrix is not updated very often, in which case the clustering procedure can be run off-line or in background.

Besides accuracy and run-time, other criteria could also be considered when choosing between both algorithms. In particular, in open environments it is usually not difficult to perform profile injection attacks that insert fake users on the user-item matrix in order to manipulate the recommendations. When that is the case, model-based algorithms, such as the k-means one, may be the best choice, since it has been shown that they are more resistant to this type of attack [18].

4. RELATED WORK

Recent research has focused on CF for Web service recommendation. For example, [14] developed a prediction algorithm of QoS values for Web services that combines user-based and item-based CF methods. The predicted QoS is used to recommend services to users. In [22] it is presented a hybrid CF algorithm that clusters users into regions based on similarities of their physical locations and historical QoS. The clusters are used to identify region-sensitive services, and a nearest neighbor approach predicts the QoS of a candidate Web service for an active user. The prediction occurs by exploiting historical QoS information gathered from users of highly correlated regions. The service with the best predicted QoS is then recommended to the active user. In [23] it is also addressed the problem of Qos prediction using a neighborhood-based collaborative filtering approach for QoS based service selection.

While the above cited works use QoS parameters as indicators of user interest, [15] points out that Web service QoS parameters, such as availability and response time, are too limited to capture the experience provided to end users. In our work, we assume that user interest in a service is represented by explicit or implicit rates provided by the user.

In [20] it is described a framework for Web service selection inspired by memory-based CF methods that considers the dependencies among Web services in composition processes. The invocation rate of a Web service carried out by a user in different Web service compositional processes is used as an indicator of the user preference for that service. The experimental evaluation of the framework was performed using the Movielens data set to simulate Web service compositions. The main focus of the authors is service selection during a composition process, when some or all of the services to be composed are already known.

Using general Web services naming tendencies coupled with enhanced syntactical methods, the work in [24] aggregates services by their messages and proactively suggests candidate services to users.

Service similarity to predict user feedback is examined in [25], although in a different context from ours. The authors propose a method for service discovery that combines multiple matching criteria with user feedback, based on the assumption that users rate how appropriate retrieved services are according to the results of their requests. Considering a given pair with one request R and one service S, when no ratings exist in the database, the method takes into account not only the ratings assigned to the current service requests R, but also ratings assigned to requests similar to R. Differently from our work, and from CF methods in general, all available user ratings are considered equally important, independently of user similarity. The predicted feedback value (score) is computed as the average of all user ratings for the corresponding service.

In contrast to our work, none of the above surveyed articles use semantic similarity of services as a strategy to increase accuracy under sparse data conditions.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented algorithms for the construction of semantic Web service recommenders using CF. The focus of our work was to use semantic markup for Web services to increase the accuracy of the recommendations based on CF algorithms when the user-item matrix is sparse. We implemented and evaluated two algorithms for recommendation: a memory-based algorithm using the k-NN method, and a model-based algorithm using the k-means method. In both algorithms, the similarity between users is computed by the Pearson Correlation Coefficient (PCC).

Usually, when the PCC method is employed in user-user CF algorithms, the similarity between two users is computed utilizing the ratings given by the users to items (services) rated in common. If the users have not rated items in common it is not possible to compute their similarity. In addition, when the number of common rated items is low, the reliability of the computed similarity degree may also be low.

In our algorithms, instead of only using the ratings of common services, the ratings of services that are semantically similar to those services rated by the users are also taken into consideration. Likewise, when predicting the rating a target user will give to a target item he has not yet accessed, the algorithms consider the ratings

given to the target item by neighbor users (or groups, in the case of the k-means algorithm), as is customary, while also considering the ratings given by neighbors to items that are semantically similar to the target item.

The experimental evaluation described shows that considering similar services when computing user similarity and predicting user ratings has a significant impact on the accuracy of the implemented algorithms, particularly when the user-item matrix is sparse. As expected, the memory-based algorithm using k-NN was more accurate than the model-based algorithm based on k-means, but the k-means algorithm is more scalable when the dynamics of the application domain permits the clustering process to be run in background. It is also interesting to point out that when the k-means algorithm considers similar services it has higher prediction accuracy than the k-NN based algorithm when the latter does not takes service similarity into account.

As a final remark it is worth noting that recommender systems usually present their recommendations in decreasing order of predicted user interest, and that users frequently consider only the top n rated items. In [26] it is observed that CF algorithms based on the k-NN method make some obscure or inaccurate recommendations at the top positions when implemented using the PCC to find neighborhoods. Usually that behavior is not evident because the algorithms are commonly rated using the MAE (as was done in our work). This metric favors algorithms that have a low average error rate over a set of predictions, but that do not necessarily place the n best recommendations at the top of the list. This performance limitation has two primary sources: (1) target users with few neighbors who have rated an item and (2) target items rated by neighbors with low correlation to the target user. The PCC addresses the second problem giving more influence to neighbors with higher similarity. But this strategy does not account for cases where all the neighbors have low correlation with the target user. Although we have not analyzed the quality of the top n recommendations, we can notice that the two mentioned sources of poor performance are related to data sparsity. And as such, our algorithms contribute to alleviate both sources of low performance: (1) by enlarging the neighborhood through considering not only users who have rated the same service, but also users who have rated similar services; and (2) by setting a threshold to the minimum similarity between two users that must be observed to permit the placement of users in the same neighborhood.

References

[1] Papazoglou, M. P., and Georgakopoulos, D. (2003). Service-Oriented Computing. *Communications of the ACM*, *46*(10), 25–28.

[2] Pan, Y., Tang, Y., & Li, S. (2011). Web Services Discovery in a Pay-As-You-Go Fashion. Journal of Universal Computer Science, 17(14), 2029–2047.

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

[4] McIlraith, S. A., Son, T. C., and Zeng, H. (2005). Semantic web services. *Intelligent Systems, IEEE*, *16*(2), 46–53.

[5] Pedrinaci, C., and Domingue, J. (2010). Toward the Next Wave of Services: Linked Services for the Web of Data. Journal of Universal Computer Science, 16(13), 1694–1719.

[6] W3C OWL Working Group. (2012). OWL 2 Web Ontology Language Document Overview (Second Edition). http://www.w3.org/TR/owl2-overview/

[7] Brickley, D., and Guha, R. V. (2006). RDF Vocabulary Description Language 1.0: RDF Schema, 2004. Retrieved from http://www. w3. org/TR/rdf-schema

[8] Tsetsos, V., Anagnostopoulos, C., and Hadjiefthymiades, S. (2006). On the Evaluation of Semantic Web Service Matchmaking Systems. *Web Services, 2006. ECOWS'06. 4th European Conference on*, 255–264.

[9] Sreenath, R. M., and Singh, M. P. (2004). Agent-based service selection. *Web Semantics: Science, Services and Agents on the World Wide Web*, *1*(3), 261–279.

[10] Stein, S., Payne, T. R., & Jennings, N. R. (2009). Flexible provisioning of web service workflows. ACM Transactions on Internet Technology (TOIT), 9(1), 2

[11] Tizzo, N. P., Adán-Coello, J. M., & Cardozo, E. (2011). Automatic composition of semantic web services using A-Teams with genetic agents. In Evolutionary Computation (CEC), 2011 IEEE Congress on (pp. 370–377).

[12] Su, X., and Khoshgoftaar, T. M. (2009). A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, *2009*, 1–19.

[13] Burke, R. (2000). Knowledge-based recommender systems. Encyclopedia of Library and Information Systems, 69, 175–186.

[14] Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2011). QoS-aware Web service recommendation by collaborative filtering. Services Computing, IEEE Transactions On, 4(2), 140–152.

[15] Van Moorsel, A. (2001). Metrics for the Internet Age: Quality of Experience and Quality of Business. *Fifth International Workshop on Performability Modeling of Computer and Communication Systems, Arbeitsberichte des Instituts für Informatik, Universität Erlangen-Nürnberg, Germany* (Vol. 34, pp. 26–31).

[16] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., et al. (2004). OWL-S: Semantic Markup for Web Services. Retrieved from http://www.w3.org/Submission/OWL-S

[17] Klusch, M., Fries, B., and Sycara, K. (2009). OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services. *Web Semantics: Science, Services and Agents on the World Wide Web*, *7*(2), 121–133. doi:10.1016/j.websem.2008.10.001

[18] Mobasher, B., Burke, R., and Sandvig, J. J. (2006). Model-Based Collaborative Filtering as a Defense Against Profile Injection Attacks. *Proceedings of the National Conference on Artificial Intelligence* (Vol. 21, p. 1388).

[19] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. (2002). Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. *Data Mining and Knowledge Discovery*, *6*(1), 61–82.

[20] Rong, W., Liu, K., and Liang, L. (2009). Personalized Web Service Ranking via User Group Combining Association Rule. *Proceedings of the 2009 IEEE International Conference on Web Services-Volume 00* (pp. 445–452).

[21] Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5–53.

[22] Chen, X., Liu, X., Huang, Z., and Sun, H. (2010). RegionKNN: A Scalable Hybrid Collaborative Filtering Algorithm for Personalized Web Service Recommendation. *2010 IEEE International Conference on Web Services* (pp. 9–16). Presented at the 2010 IEEE International Conference on Web Services (ICWS), Miami, FL, USA. doi:10.1109/ICWS.2010.27

[23] Wu, J., Chen, L., Feng, Y., Zheng, Z., Zhou, M. C., & Wu, Z. (2013). Predicting quality of service for selection by neighborhood-based collaborative filtering. Systems, Man, and Cybernetics: Systems, IEEE Transactions On, 43(2), 428–439.

[24] Blake, M. B., & Nowlan, M. F. (2007). A web service recommender system using enhanced syntactical matching. In Web Services, 2007. ICWS 2007. IEEE International Conference on (pp. 575–582).

[25] Averbakh, A., Krause, D., & Skoutas, D. (2009). Exploiting User Feedback to Improve Semantic Web Service Discovery. Presented at the 8th International Semantic Web Conference (ISWC 2009).

[26] McLaughlin, M. R., and Herlocker, J. L. (2004). A collaborative filtering algorithm and evaluation metric that accurately model the user experience. *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 329–336).

Received: March 2014. Accepted: May 2014.