2011 International Conference on Advanced Information Networking and Applications

A Web Service Recommender System Using Vector Space Model and Latent Semantic Indexing

Nguyen Ngoc Chan*, Walid Gaaloul† and Samir Tata‡
Information Department
TELECOM & Management SudParis
UMR 5157 CNRS Samovar, France
*Email: *chan.nguyen_ngoc, †walid.gaaloul, ‡samir.tata@it-sudparis.eu*

***Abstract*—The tremendous growth in the amount of available web services (WS) impulses many researchers on proposing recommender systems to help users discover services. Most of the proposed solutions analyzed query strings and web service descriptions to generate recommendations. However, text based recommendations approaches depend mainly on user's perspective, languages and notations which easily decrease recommendation's efficiency. In this paper, we propose to take into account user's behaviors instead of text based analysis. We apply collaborative filtering technique on user's interactions. We propose and implement two algorithms based on Vector Space Model and Latent Semantic Indexing to validate our approach. We also provide evaluation methods with different datasets in order to compare and assert the efficiency of our two algorithms.**

***Keywords*-web service, recommender system, vector space model, latent semantic indexing**

I. Introduction

A. Context

The growing number of web services and the discontinuing of public UDDIs make web services more and more scattered. Some portals such as XMethods, BindingPoint, WebServiceX.NET, WebServiceList, StrikeIron, Woogle, RemoteMethods, or eSynaps assist users by providing interfaces for searching and invoking web services. However, the common users may get confused about the number of web services returned by a search engines. They also do not know about the advantage of a web service in comparison with other ones. On the other hand, there is no collaboration among web services providers and there is no system checking the availability, accessibility and usability of web services [1]. Intuitively, users need a system which can help them validate the service's functionality and suggest them appropriate services as they are looking for.

B. Problem statement

For web service discovery, recommender systems [2], [3] are well-known solutions to help users find their expected WS operations. Current researches have applied techniques used in Information Retrieval such as vector space model (VSM) and term frequency-inverse document frequency (TF-IDF) to build recommender systems or special search engines for web services [4], [5], [6]. They syntactically matched terms in *query strings* and *web service descriptions* or stored and tracked queries. These solutions, however, encountered the lexical analysis problems due mainly to the semantic ambiguity of the text-based approaches.

The performance of the recommender system can be enhanced using semantic annotation markup languages. Some solutions analyze the services in semantic descriptions with ontologies and relations [7], [8]. In these approaches, each user is able to employ a standard ontology, consisting of a set of basic classes and properties, for declaring and describing services. Nevertheless, it is time-consuming to create and publish ontology annotated content as it needs to be done by domain human experts and powerful editing tools.

Basically, text-based approaches cannot capture user's interests because they worked on the service *provider side* instead of the *user side*. They provide recommendations using query strings and services descriptions as inputs, or *explicit knowledge* in which users interests or service semantic description are explicitly described. These inputs used to build the recommender encounter the synonym and polysenym problems, and are not easy to be collected because they commonly need additional efforts from the users or the service providers to have the explicit knowledge to describe the related users' profiles or services' semantic descriptions. Obviously, recommender systems based on *query strings* or *text processing* could not completely satisfy users because of the lexical and semantical problems. There should be another solution which can overcomes the shortcomings of these approaches.

C. Our approach

In this paper, we propose an approach that aims at solving the problem from the *user side*. We focus on user's IDs and their used WS operations instead of query strings and services descriptions. We realize that user's interactions are very important because they reflect user's behaviors and interests. This *implicit knowledge* is not presented in any individual document. Indeed, the above described approaches could not perfectly capture user's interests because they worked on the service provider side instead of the user side.

1550-445X/11 $26.00 © 2011 IEEE
DOI 10.1109/AINA.2011.99

They provide recommendations using *explicit knowledge* (users interests or service (semantic) description, etc.). These inputs used to build the recommender encounter also the synonym and polysemy problems, and are not easy to collect because they commonly need additional efforts from the users or the service providers to have the explicit knowledge to describe the related users' profiles or services' semantic descriptions.

By tracking *user's interactions*, we can capture interesting and meaningful *implicit knowledge* and plan strategies to build a recommender system for web services discovery without asking additional efforts from the users (such as their profiles and comments) or the service providers (such as functional and nonfunctional descriptions). Concretely, we proposed a collaborative filtering technique on user's interactions to generate web service recommendations by implementing and comparing two algorithms (VSM-based and LSI-based).

The paper is processed in the next section with our algorithms to generate the recommendations. Section 3 presents our implementation and experimental results based on the proposed algorithms. The related work and discussion are illustrated in section 4. Finally, we conclude our approach and sketch the future work in section 5.

II. Recommendations based on user's historical data

VSM and LSI are two collaborative filtering techniques commonly used in Information Retrieval (IR) [9], [10] to find the similar documents to a given one. In common, documents are the text files and terms in each document are words. The data about documents and terms in a collection can be denoted by a $m \times n$ matrix where m is the number of terms and n is the number of documents. The relationships among them can be inferred by the two basic techniques VSM and LSI.

In our approach, we provide two techniques inspired respectively by VSM and LSI to completely overcome the drawbacks of the text-based recommender systems. These two techniques proceeds in three steps :

1) Processing historical data : In the first step (see section II-A), users' usage data are processed to map to the terms-documents structure.
2) Computing similarities vectors : In the second step (see section II-B), the weight matrix and the distribution matrix of WS operations which contain similarities vectors are computed based on the historical data.
3) Generating recommendations: In the third step (see section II-C), based on the similarities vectors, recommendations are provided.

A. Processing historical data

Inspired the way of computing similarities among documents in a corpus, in our approach, we apply VSM and LSI on the user's historical data to recommend the most suitable WS operations for each user. To deal with this issue, we define *each user as a document and the set of WS operations that he used as terms* in that document. By this definition, the number of users is the number of documents in the corpus and this number increases when the number of system's users increases. Then, given a historical data with m WS operations and n users, we construct a $A_{m \times n}$ matrix called the operation-user matrix. The value of each entry $A_{i,j}$ in this matrix is the number of used times that the WS operation O_i was used by the user U_j.

To illustrate our techniques, we use scenario with four users who used some of service operations as following: U_1 = {getWeather(2), getLocation(2), getNews(1), getHotels(3)}; U_2 = {getLocation(4), getBook(1), getNews(2), getHotelID(4), getPlaces(3), getWeatherByZipCode(2)}; U_3 = {getPlaces(2), getWeatherByDate(5), getWeatherByZipCode(3), getHotelID(5), getCityDescByName(2)} and U_4 = {getHotel(7), getCityDescByName(8)}. The record "getWeather(2)" in the set U_1 means that the user U_1 has used the WS operation getWeather two times and this meaning is the same to other records. These historical data is represented in a $m \times n$ matrix given in the Table.I where $m = 10$ and $n = 4$.

	U_1	U_2	U_3	U_4
getWeather	2	0	0	0
getLocation	2	4	0	0
getNews	1	2	0	0
getHotels	3	0	0	7
getBook	0	1	0	0
getHotelID	0	4	5	0
getPlaces	0	3	2	0
getWeatherByZipCode	0	2	3	0
getWeatherByDate	0	0	5	0
getCityDescByName	0	0	2	8

Table I
Original usage matrix

B. Computing similarities vectors

1) VSM.: VSM is firstly introduced by Gerard Salton et al. [11] and became a popular technique in research and industry. It is an algebraic model which represents objects in vectors by which we can infer the similarity between two objects based on the angle between their respective vectors. All objects are presented in the same k-dimensional space and each entry in a vector presents the weight of that vector in a particular dimension. The common weight which is used along with VSM in most of applications is Term Frequency-Inverse Document Frequency (TF-IDF) [9]. In principle, TF-IDF reflects the importance of a word in a collection of documents and in practical, it is used to assess the importance of an object in a particular dimension in a k-dimensional space.

In our approach, we apply the TF-IDF computation on the original matrix $A_{m\times n}$ to compute the weight of each item. Suppose that $|A_{ij}|$ is the number of times that the user U_j has used the WS operation O_i, $|U_j|$ is the number of times that the user U_j used all WS operations and $|O_i|$ is the number of users who used the WS operation O_i. Then, the weight of each item A_{ij} is given by (1).

$$w_{i,j} = TF_{i,j} * IDF_i = \frac{|A_{ij}|}{|U_j|} * log\frac{|n|}{|O_i|} \quad (1)$$

The result achieved after the TF-IDF computation is a weight matrix (see Table.II), which presents the weight of each WS operation in the collection.

	U_1	U_2	U_3	U_4
getWeather	0.35	0	0	0
getLocation	0.17	0.17	0	0
getNews	0.09	0.09	0	0
getHotels	0.26	0	0	0.32
getBook	0	0.09	0	0
getHotelID	0	0.17	0.20	0
getPlaces	0	0.13	0.08	0
getWeatherByZipCode	0	0.09	0.12	0
getWeatherByDate	0	0	0.41	0
getCityDescByName	0	0	0.08	0.37

Table II
WEIGHTS COMPUTED BY TF-IDF.

2) LSI.: Latent Semantic Analysis which was patterned by Scott Deerwester et al. [12] is a mathematical, statistical technique for extracting and inferring relations of documents and terms. It applies single value decomposition (SVD) which is one of the factorization algorithms used for collaborative filtering [13]. In a particular context of IR, LSA is applied as the Latent Semantic Indexing (LSI) which is able to correlate semantically related terms that are "latent" in a collection of contexts [14]. LSI is being used in variety of IR and text processing applications such as information discovery, relationship discovery, online customer support, filtering spams and so on [15], [16]. Like VSM, LSI also uses the term-document matrix and term-weight to identify the occurrences of terms within a set of documents. However, it applies another mathematical transform technique which is the SVD to reduce the number of dimensions in term space which is more usable and efficient [17]. Concretely, we reused the operation-user matrix and follow the LSI computation's principle [18], [17] to find the weight of each WS operation. By the SVD theorem [18], [15], the original matrix $A_{m\times n}$ is factorized in the form of matrices multiplication (2).

$$A_{(m\times n)} = U_{(m\times n)}S_{(n\times n)}V^T_{(n\times n)} \quad (2)$$

where $U_{(m\times n)}$ and $V_{(n\times n)}$ are orthogonal matrices, which present the left and right singular vectors of A as their columns. $S_{(n\times n)}$ is an m-by-n diagonal matrix holding the singular values, S = diag(σ_1, σ_2, ..., σ_n), $\sigma_i > 0$ for $1 \leq i \leq r \leq n$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$.

There exists dedicated research on calculating the SVD in the literature such as [19] and there exist also the implementations of LSI using C, C++ or Java[1]. Hence, we focus on how to apply SVD on our application. The matrix A has the rank r which is denoted by the number of non-zero values in the matrix S. In traditional LSI, the value of r infers the number of dimensions which used to present the documents. It means also the number of contexts that the documents belong to. The maximum value of r is n which infers that all of the documents are completely independent. In IR and other applications of LSI, r is decreased to k, $0 < k < r$ to reduce the number of dimensions, group documents in k contexts and avoid the large-scale matrices computation(3). Since WS operations are represented in k contexts and we are able to define each context as a dimension, the set of contexts can be considered as a space with k dimensions and each WS operation can be located at a particular position in that space. In our approach, we consider each user's usage as a document and apply the LSI, we group the usage data in to k contexts which are presented by the approximated matrix $A_{k(m\times n)}$ (see Fig.1).

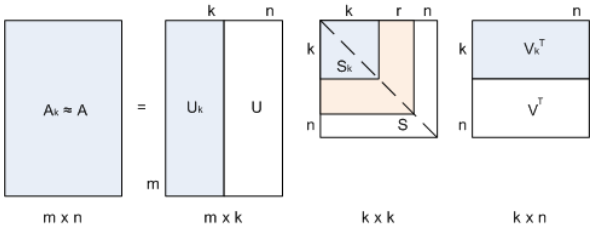

Figure 1. Decomposition in k dimensions.

$$A_{(m\times n)} \approx A_{k(m\times n)} = U_{(m\times k)}S_{(k\times k)}V^T_{(k\times n)} \quad (3)$$

The matrix $U_{(m\times k)}$ represents the distribution of m WS operations in the k-dimensional space. Each WS operation is represented by a k-dimensional vector. Rows of $U_{(m\times k)}$ hold eigenvector values which are used to find the coordinates of a query vector $\overrightarrow{q_{(m)}}$. These coordinates are determined by the multiplication of $\overrightarrow{q^T_{(1\times m)}}$ with the $U_{(m\times k)}$ and the inverse matrix of $S_{k\times k}$ (see (4)).

$$\overrightarrow{q_{(k)}} = \overrightarrow{q^T_{(m)}} \times U_{(m\times k)} \times S^{-1}_{(k\times k)} \quad (4)$$

In our application, whenever a user U_j uses an operation O_i, we map this to a query vector $\overrightarrow{q(m)}$ in which only the q_i element is equal to 1 and others are equal to 0 (5). Then, applying (4), we can easily find the coordinates of the user's input in the k-dimensional space.

$$\overrightarrow{q^T_{ij(m)}}(t) = \begin{cases} A_{ij} & \text{if } t = i,\ t \in [1..m] \\ 0 & \text{if } t \neq i,\ t \in [1..m] \end{cases} \quad (5)$$

In our given example, suppose that we target to find the distribution of WS operation in the 3-D space, we apply

[1]http://www.cs.utk.edu/~lsi/

the SVD on the original matrix and truncate the result with $k = 3$, we get matrix U_k, S_k and V_k shown in Table.III.

$$U_k = \begin{bmatrix} 0.04 & 0.01 & 0.18 \\ 0.10 & -0.26 & 0.66 \\ 0.05 & -0.13 & 0.33 \\ 0.62 & 0.30 & 0.25 \\ 0.02 & -0.07 & 0.12 \\ 0.22 & -0.64 & -0.03 \\ 0.11 & -0.35 & 0.15 \\ 0.13 & -0.35 & -0.07 \\ 0.16 & -0.37 & -0.51 \\ 0.71 & 0.19 & -0.23 \end{bmatrix} \quad S_k = \begin{bmatrix} 11.13 & 0.00 & 0.00 \\ 0.00 & 9.28 & 0.00 \\ 0.00 & 0.00 & 5.50 \end{bmatrix}$$

$$V_k = \begin{bmatrix} 0.19 & 0.03 & 0.50 \\ 0.18 & -0.61 & 0.66 \\ 0.35 & -0.69 & -0.56 \\ 0.90 & 0.39 & -0.02 \end{bmatrix}$$

Table III
ORIGINAL MATRIX'S DECOMPOSITION.

Suppose that the user U_1 now uses the WS operation "getPlaces", the query vector of this usage is presented by $\overrightarrow{q^T_{(1\times m)}}$=(0,0,0,0,0,0,1,0,0,0). The coordinates of q in the 3-D space determined by (4) are given in (6).

$$\overrightarrow{q_k} = \overrightarrow{q^T_{(1\times m)}} \times U_{(m\times k)} \times S^{-1}_{(k\times k)} = (0.01, -0.04, 0.03) \quad (6)$$

C. Generating recommendations

1) VSM.: The weight matrix in IR is used to compute the similarities among items and the popular metric used for this computation is the cosine measure. We apply this principle on our strategy to generate the WS recommendations. Suppose that the user U_j currently selects the WS operation O_i, the recommendations are generated in two steps:

1) Finding the k-most similar users with U_j.
2) Finding the l-most similar WS operations with O_i from the set of WS operations used by the k selected users.

If we consider each row or each column in the weight matrix computed in the section II-B as a vector, this matrix contains either the distribution of m WS operations in the n dimensional space or the distribution of n users in the m dimensional space. The cosine value of the angle between two vectors presents their closeness or in other words their similarities. Hence, in our first step, the similar users with the current one U_j can be extracted based on the weight matrix and cosine value. In general, suppose that two items I and J are represented by two vectors $\overrightarrow{v_i}(\ i_1,\ i_2,\ \ldots,\ i_n)$ and $\overrightarrow{v_j}(\ j_1,\ j_2,\ \ldots,\ j_n)$, the similarity between them is inferred from the cosine value of the angle between them (7).

$$\begin{aligned} similarity(I,J) &= |cosine(\overrightarrow{v_i}, \overrightarrow{v_j})| \\ &= \frac{|\sum_{k=1}^{n} i_k * j_k|}{\sqrt{\sum_{k=1}^{n} i_k^2} \times \sqrt{\sum_{k=1}^{n} j_k^2}} \end{aligned} \quad (7)$$

Applying the equation (7) on users' similarities vectors, we easily compute the similarities between the user U_j and others. In our example, with the suppose that U_1 uses the WS operation "getPlaces", the similar users to U_1 are U_4(0.36) and U_2(0.25). Since the similarity value of (U_1,U_3) is 0, the user U_3's data is eliminated in the next step.

After collecting the k-most similar users to U_j, we represent the weight matrix by eliminating the unselected users. Each WS operation in the new weight matrix is represented by a k-dimensional vector. Then, in the second step, we re-aplly the cosine similarity (7) to find the l-most similar WS operations with the current one (O_i). In our example, the selected WS operations with $k = 3$ and $l = 3$ are {getBook(1.00), getHotelID(1.00), getWeatherByZipCode(1.00)} since they have the three highest similarity values equal to 1.00.

2) LSI.: In the LSI-based algorithm, we already have the distribution of the WS operations in k-dimensional space which is presented by U_k. In the other hand, the current used WS operation is also presented by the k-dimensional vector with the coordinates computed by (4). Hence, we also apply the cosine measure to compute the similarities between this current WS operation with others (7) and pick up the l-most similar ones for the recommendations. In our example, the LSI-based algorithm with $k = 3$ and $l = 3$ returns the WS operations {getBook(0.92), getNews(0.85), getLocation(0.85)} since they have the three highest similarity values respectively equal to 0.92, 0.85 and 0.85.

Our algorithms finally return the list of l WS operations for the recommendation. The historical data is computed as similarities vectors and represented in a reduced space with k dimensions. In experiments, we validate the algorithms with various values of k and analyze the results.

In our illustrating example, from the original usage matrix, we can see that all of the users are willing to get the information about weather, places and hotels that they want to stay during their vacations. Intuitively, we can guess that they have the same favorite as tourism. However, in reality, the VSM could not capture the similar favorite between U_1 and U_3, U_2 and U_4 because the value of similarity(U_1,U_3) and similarity(U_2,U_4) are 0. Hence, when we reduce the original space into k dimensions with VSM, we can lost some important usage data which can make the recommendation more accurate. Different from VSM, the LSI-based algorithm can infer the relationships among users based on the whole dataset. Therefore, we expect that it can capture the missed relations which VSM cannot detect and give us better recommendations. In experiments, our assessment showed that the LSI-based algorithm achieves better results than the VSM's algorithm.

III. VALIDATION

In this section, we present the implementation efforts we have done to validate our approach. First, we describe the application we have developed to implement our two (LSI, VSM) proposed algorithms (see section III-A). Then, we specify the evaluation's data (see section III-B) and analyze

the experimental results of original evaluation methods to assert the quality of the produced recommendations (see section III-C). Concretely, two datasets sources were used for our evaluation. Beside the usage data collected from our tool, we validated our algorithms on a big and equivalent dataset to a wider the data source (see sections III-B and III-C).

A. Implementation

We have developed a server-client application which allows users to register, set up their profiles and use the WS operations; and providers to upload their WSDL web services files. Besides a simple search engine, our tool provides to the user recommendation lists based on the current invoked WS operation. Our tool can be found at http://www-inf.it-sudparis.eu/SIMBAD/tools/WSRS/ws_recommender.html. The application was deployed in two parts: front-end written in Flex[2] and back-end in Java (see Figure 2). Mechanisms used to exchange data between server and client are the RemoteObject and BlazeDS framework[3]. Tomcat server 5.5 was set up for hosting our application.

Figure 2. WS Recommender System Application

Our tool's architecture is given in Fig.3. Concretely, a user can search or invoke a WS operation. His search results are recorded by an "Evaluation Module" for the evaluation steps. A "Data Preparation Module" manipulates the historical data for the recommendation and evaluation. The "DB Management Module" handles and monitors all of the interactions with the database. The "WS Collector & Checker" collects service descriptions from providers, crawlers, search engines, anonymous systems or even users. It not only extracts the WS operations but also checks the availability and usability of the collected services.

B. Evaluation's data collection

In this section, we present the different datasets used to evaluate our techniques. The first dataset was provided by

[2]http://www.adobe.com/devnet/flex/

[3]http://opensource.adobe.com/wiki/display/blazeds/BlazeDS

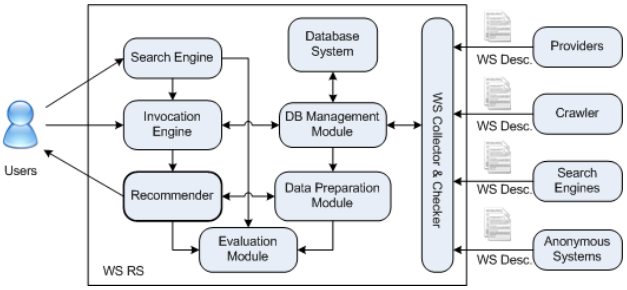

Figure 3. Architecture of the WS Recommender System

our tool. After the application was tested and published on our institute's intranet network, we invited PhD students and researches to register and use it. After two week, we collected the usage data, run the evaluation algorithms and analyzed the results. However, the usage data collected by our tool is not so big (24 registration and 271 usage records) because of the scope restriction, and the fact that it is hard to find a real dataset about web service usage of each user because is not commonly published by any provider, we decided to extend our experiments on a big and equivalent one which is the AudioScrobbler[4]. We extract the data from this dataset, which consists of 1003 users, 3435 items, 644.281 records and 12.332.214 times of usage, and divide it into two parts: a training part and a test part. We run our algorithm on the data in training part, got results, and evaluated them using the data in the test part. Details of the evaluation and results are presented in the next section.

C. Evaluation

Root mean square error(RMSE) [20] is very popular used to evaluate the efficiency of a prediction by comparing the prediction data with the ground-truth data. Especially, it is the significant and unique metric used in the Netflix Prize[5] Contest [21]. In our approach, we used this metric to evaluate the efficiency of our proposed algorithms. Small RMSE values indicate good prediction. To compute RMSE, suppose that the prediction matrix is $P \in R_{(m \times n)}$ and the ground-truth answer matrix $A \in R_{(m \times n)}$. Let $I \in \{0,1\}_{(m \times n)}$ be the indicator of A. The RMSE between the prediction P and the answer A is defined as:

$$RMSE(P,A) = \sqrt{\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(A_{ij}-P_{ij})^2}{\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}}} \quad (8)$$

In general, our experiments on the two datasets showed quite similar results. However, due the lack of space and as the AudioScrobbler is more meaningful, we choose[6] to explain and show in the following the experiments obtained

[4]http://www.audioscrobbler.net/development/

[5]http://www.netflixprize.com/

[6]Experimentsobtained from our other application's dataset can be found at http://www-inf.it-sudparis.eu/SIMBAD/tools/WSRS/ws_evaluation.html

using the AudioScrobbler. Concretely, we divided the refined AudioScrobbler dataset into two parts: the training part consists of 4/5 the number of records and the rest (1/5) was used as the test set. We run our algorithms on the data in the training part with the k parameter and compared the collected results with the test part. In the VSM algorithm, k is the number of the most similar users with the current one, meanwhile in the LSI, k is the number of dimensions reduced from the SVD.

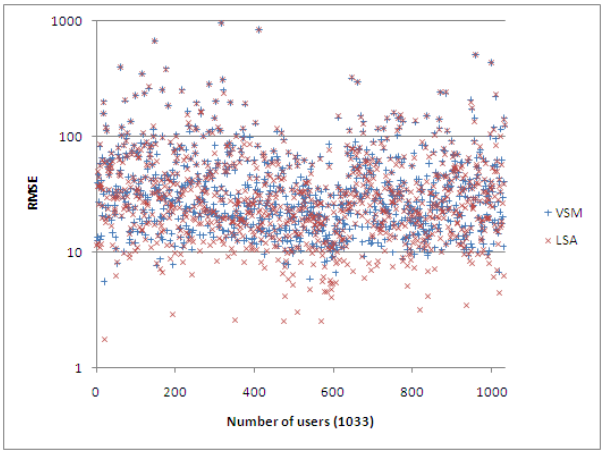

Figure 4. LSI and VSM distribution with k = 100

After running the algorithms with k ={1, 2, 5, 10, 20, 50, 100, 200} for all users, we analyzed the results and we concluded that *the LSI-based algorithm achieves better predictions than the VSM-based* with the lower values of RMSE. Typical result with $k = 100$ is shown in the Figure 4.

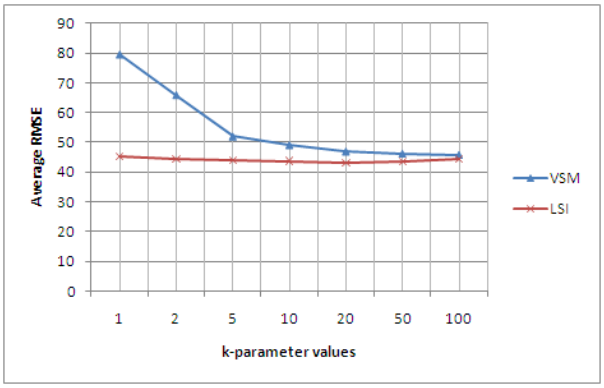

Figure 5. Average RMSE of LSI and VSM

Another result (see Figure 5) showed that showed that with small k (k<100), the average RMSE values of the LSI-based algorithm are smaller than the VSM-based and when k increases, the accurate of the VSM's prediction increases. And among these values, the best prediction using LSI-based algorithm is always better than the best prediction using VSM-based algorithm (see Figure 6). The minimum RMSE values returned by the LSI-based algorithm are lower than the VSM-based with the tested values of the k-parameter. This means that we can find a better prediction based on the LSI rather than the VSM.

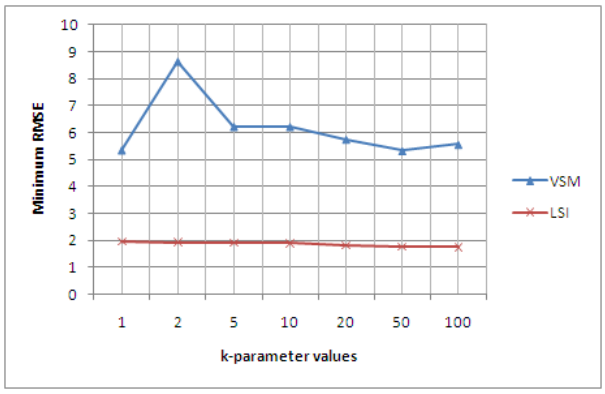

Figure 6. Minimum RMSE of LSI and VSM

Our two algorithms achieved good results with small values of RMSE. It means that these algorithms can be applied for generating recommendation for users, especially web services recommendations based on user's behaviors.

In the other hand, we have also done experiments using precision and recall measures which showed also good results (details of the evaluation can be found at http://www-inf.it-sudparis.eu/SIMBAD/tools/WSRS/ws_evaluation.html). Precision and recall are two of the most popular measures used to evaluate the accuracy of a recommender system. They are computed by comparing the list of elements retrieved by our algorithm with a "relevant data" set (or correct list). Depending on the context of applications, the "relevant data" can be either collected from historical data, or predefined by the experts, and so on.

However, due to the space's limitation, we prefer to present the results measured by the RMSE which is more suitable to our evaluation. Indeed, the RMSE is a useful measure which shows off the efficiency of the proposed algorithms regardless the "relevant data". With precision and recall, we defined the "relevant data" as the data returned from the application's search engine and the last-used operations of each users. However, this "relevant data" cannot perfectly reflect the correctness and effectiveness of the proposed algorithms. Therefore, we used the RMSE as a main measure to evaluate our algorithms.

IV. Related Work and Discussion

The main goal of building a recommender system for web service discovery is providing the closest services with

user's interests. The basic measure which is used to find the expected services is the similarity between request from users (query, profile, historic data, etc) and data stored in the service repository. Previous approaches, which were based on the traditional service descriptions(WSDL documents), could be classified in the categories: clustering [4], rating based [22], words analysis [23] and vector space model [5], [6], [24]. A recent work built a WS recommender system based on the Pearson Correlation Coefficient (PCC) and collaborative prediction [25]. However they provided the recommendations based on the QoS values which do not reflect the user's interest and expectation.

In semantic web services discovery, the similarity can be computed using the service's elements [7] (such as input, output, precondition and effect), service requests [26] or ontologies matches [27], [22]. However, the semantic web service descriptions are somewhat the extensions of traditional WSDL files with new elements and annotations which can link them to semantic concepts and defined domains or ontologies. The recommendation processes based on semantic descriptions are also the text based analysis. The advantage of our approach is that we can capture the "semantic" model without using the semantic service descriptions or the web ontology language(OWL).

LSI is applied in various fields of study, including IR, synonym testing, semantic priming, information filtering, novel applications and other statistics aspects [18], [15]. In the area of deploying good recommenders for web services discovery, this method is also applied in different ways. A remarkable approach which applied LSI on a set of WSDL documents and terms was proposed by Chen Wu et. al. [16]. They followed the guide of [17] to find the most similar service descriptions with the user's query. However, their prototype supported only single-word queries and the evaluation was done with 10 queries at all. Another approach also applying the SVD on services descriptions to generate services recommendations was introduced by Jiangang Ma et. all. [28]. They proposed a "divide and conquer" method, using the Bisecting k-means algorithm and Euclidean distance, to handle the poor scalability in the Web environment and the issue of lacking semantics.

In practical applications, SVD, which is the heart of LSI, generates a reduced latent space, which is the best approximation of original vector space and contains the main latent information among the co-occurrence activities[29]. Hence, many approaches tried to analyze and improve the SVD to capture more exact semantic relations[30].

In our work, we employed two collaborative filtering algorithms based on VSM and LSI using implicit historical data. Our algorithms generate recommendations without asking the users any explicit knowledge such as interesting, ratings, comments, profiles, etc. In practice, our experiments showed that by tracking user's historical data, which reflects the user's behavior, we can infer good recommendations.

V. Conclusion

Collaborative filtering is one of the most powerful techniques in making recommendations. Based on this technique, we proposed two algorithms which can generate good recommendations and avoid the problems of string-based approaches. Our contributions in this work can be summarized in three main points: (1) we *adapted* the original VSM and LSI in a *new context* which is the recommendation based on user's behaviors; (2) we proposed *innovative recommendation methods* in which we represented the usage data as vectors in a k-dimensional space, especially in the LSI-based algorithm, we also computed the coordinates of the current usage in this space; and (3) we brought not only an efficient measure which is RMSE but also *original evaluation methods* which are based on precision and recall[7] to prove the efficiency of our techniques.

Our proposed algorithms generate recommendations based only on the user's usage data within the discovery process. Thus, our approach is *self-contained* within the web service discovery process, *independent* from any explicit or human centric or error prone knowledge. Whereas other information such as user's rating or service reputation can be hard to be captured, our approach is a good solution as it does not use such kind of explicit knowledge as inputs to propose recommendations.

In our future works, we will build a specific recommender system for web service discovery by taking into account other WS parameters such as services descriptions, relations, compositions, etc. We are also looking on measuring the satisfactory of users. To that end, other parameters such as the implicit relations among users (or WS operations) and the number of times that users access to the system will be considered. We are also working on how to compare our experiments with other approaches as we (our approach and the others) do not use the same input data. To do so, we have already published on the web our results and we are looking for the other approaches' published implementations and experiments.

References

[1] B. Daniel, S. Katharina, L. Holger, and D. Fensel, "Web service discovery ? a reality check," *3rd European Semantic Web Conference, Budva, Montenegro*, June 2006.

[2] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, March 1997.

[3] C.-T. Wu and H.-F. Wang, "Recent development of recommender systems," *Industrial Engineering and Engineering Management, 2007 IEEE International Conference on*, pp. 228–232, Dec. 2007.

[7] http://www-inf.it-sudparis.eu/SIMBAD/tools/WSRS/ws_evaluation.html

[4] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*. VLDB Endowment, 2004, pp. 372–383.

[5] C. Platzer and S. Dustdar, "A vector space search engine for web services," *Web Services, 2005. ECOWS 2005. Third IEEE European Conference on*, pp. 9 pp.–, Nov. 2005.

[6] A. Birukou, E. Blanzieri, V. D'Andrea, P. Giorgini, and N. Kokash, "Improving web service discovery with usage data," *Software, IEEE*, vol. 24, no. 6, pp. 47–54, Nov.-Dec. 2007.

[7] Z. Wang, K. Liu, G. Lv, and X. Hao, "Study of an algorithm of web service matching based on semantic web service," in *ALPIT '07: Proceedings of the Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 429–433.

[8] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara, "Semantic matching of web services capabilities," in *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*. London, UK: Springer-Verlag, 2002, pp. 333–347.

[9] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*. New York, NY, USA: ACM, 2000, pp. 158–167.

[11] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[12] S. C. Deerwester, S. T. Dumais, G. W. Furnas, R. A. Harshman, T. K. Landauer, K. E. Lochbaum, and L. A. Streeter, "Computer information retrieval using latent semantic structure," *US Patent No. 4839853*, June 1989.

[13] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman, "Using singular value decomposition approximation for collaborative filtering," in *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 257–264.

[14] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Informatioin Science*, vol. 41, no. 6, pp. 391–407, 1990.

[15] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Rev.*, vol. 37, no. 4, pp. 573–595, 1995.

[16] C. Wu, V. Potdar, and E. Chang, *Advances in Web Semantics I: Ontologies, Web Services and Applied Semantic Web*. Berlin, Heidelberg: Springer-Verlag, 2009, ch. Latent Semantic Analysis — The Dynamics of Semantics Web Services Discovery, pp. 346–373.

[17] A. Kontostathis and W. M. Pottenger, "A framework for understanding latent semantic indexing (lsi) performance," *Inf. Process. Manage.*, vol. 42, no. 1, pp. 56–73, 2006.

[18] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse Processes*, no. 25, pp. 259–284, 1998.

[19] M. W. Berry, "Large scale singular value computations," *International Journal of Supercomputer Applications*, pp. 13–49, 1992.

[20] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[21] B. James and L. Stan, "The netflix prize," *KDDCup.07*, vol. 0, 2007.

[22] U. S. Manikrao and T. V. Prabhakar, "Dynamic selection of web services with recommendation system," in *NWESP '05: Proceedings of the International Conference on Next Generation Web Services Practices*. Washington, DC, USA: IEEE Computer Society, 2005, p. 117.

[23] M. B. Blake and M. F. Nowlan, "A web service recommender system using enhanced syntactical matching," in *ICWS*, 2007, pp. 575–582.

[24] N. Kokash and R. Birukou, "Web service discovery based on past user experience," in *Proc. Intl Conf. Business Information Systems (BIS 07), LNCS 4439*. Springer, 2007, pp. 95–107.

[25] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in *ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 437–444.

[26] A. V. Paliwal, N. R. Adam, and C. Bornhovd, "Web service discovery: Adding semantics through service request expansion and latent semantic indexing," *Services Computing, IEEE International Conference on*, vol. 0, pp. 106–113, 2007.

[27] S. Wu, "A new web services matching algorithm," in *IUCE '09: Proceedings of the 2009 International Symposium on Intelligent Ubiquitous Computing and Education*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 414–416.

[28] J. Ma, Y. Zhang, and J. He, "Web services discovery based on latent semantic approach," in *ICWS '08: Proceedings of the 2008 IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 740–747.

[29] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, 1st ed. Addison Wesley, May 1999.

[30] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," *KDDCup.07*, vol. 0, 2007.