Unified Collaborative and Content-Based Web Service Recommendation

Lina Yao, Quan Z. Sheng, *Member, IEEE*, Anne. H. H. Ngu, Jian Yu, and Aviv Segev, *Member, IEEE*

Abstract—The last decade has witnessed a tremendous growth of Web services as a major technology for sharing data, computing resources, and programs on the Web. With increasing adoption and presence of Web services, designing novel approaches for efficient and effective Web service recommendation has become of paramount importance. Most existing Web service discovery and recommendation approaches focus on either perishing UDDI registries, or keyword-dominant Web service search engines, which possess many limitations such as poor recommendation performance and heavy dependence on correct and complex queries from users. It would be desirable for a system to recommend Web services that align with users' interests without requiring the users to explicitly specify queries. Recent research efforts on Web service recommendation center on two prominent approaches: *collaborative filtering* and *content-based recommendation*. Unfortunately, both approaches have some drawbacks, which restrict their applicability in Web service recommendation. In this paper, we propose a novel approach that unifies collaborative filtering and content-based recommendations. In particular, our approach considers simultaneously both rating data (e.g., QoS) and semantic content data (e.g., functionalities) of Web services using a probabilistic generative model. In our model, unobservable user preferences are represented by introducing a set of latent variables, which can be statistically estimated. To verify the proposed approach, we conduct experiments using 3,693 real-world Web services. The experimental results show that our approach outperforms the state-of-the-art methods on recommendation performance.

Index Terms—Web service recommendation, service discovery, collaborative filtering, content-based recommendation, hybrid approach, three-way aspect model, data sparsity

1 Introduction

After a decade of research and development, Web services have become one of the standard technologies for sharing data and software and the number of Web services available on the Web is constantly increasing [1], [2], [3], [4], [5]. This increase has been further accelerated by the emerging cloud computing as a new computing paradigm for provisioning of diverse services on demand [6], [7]. According to a recent statistics from `seekda.com`, there are 28,606 Web services available on the Web, offered by 7,739 different providers. This increasing adoption and presence of Web services calls for novel approaches for efficient and effective service recommendation, which is a critical issue in many practical applications such as service discovery and composition [8], [9], [10], [11].

Web service recommendation is the process of automatically identifying the usefulness of services and proactively recommending services to end users. We can also view Web service recommendation as the process of service selection augmented with end user behavior analysis to achieve relevant and accurate service suggestions. Traditional Web service discovery centers around UDDI (Universal Description, Discovery and Integration) registries [12]. Unfortunately, UDDI is no longer the choice of publishing Web services, evidenced by the shutdown of the public UDDI registries by big players such as IBM, Microsoft, and SAP [13]. Over the last decade, a considerable number of Web service discovery approaches have been proposed [14], [15] and several Web services publication websites have emerged such as `WebServiceList`[1], `XMethods`[2], and `ProgrammableWeb`[3]. These approaches and websites largely exploit keyword-based search techniques and are insufficient to fully describe the functionalities of Web services. Furthermore, accommodation for non-functional characteristics such as quality of service (QoS) of Web services during the service selection and recommendation are very limited [16], [17]. In a recent work by Zheng et al. [8], [18], a Web service search engine is designed and developed that ranks Web services not only by functional similarities to a user's query, but also by non-functional QoS characteristics of Web services.

The main goal of our work is to advance the current state-of-the-art on Web services recommendation. More specifically, our work is inspired by the following observations. To find desirable Web services by using Web service search engines, a user normally has to supply the queries and often at a loss as to what queries are appropriate (e.g., which keywords should be used, what values should be set for a QoS attribute). Another problem

- *L. Yao and Q.Z. Sheng are with the School of Computer Science, the University of Adelaide, SA 5005, Australia. E-mail: {lina, qsheng}@cs.adelaide.edu.au*
- *A. H.H. Ngu is with Department of Computer Science, Texas State University, United States.*
- *J. Yu is with School of Computer and Mathematical Sciences, Auckland University of Technology, New Zealand.*
- *A. Segev is with Department of Knowledge Service Engineering, KAIST, Korea.*

1. http://www.webservicelist.com.
2. http://www.xmethods.net.
3. http://www.programmableweb.com.

is that Web services that do not satisfy the user's search query are completely excluded from the recommendation list. It is therefore desirable that a recommendation system selects *probably-preferred* Web services by estimating user preferences without requiring users to explicitly specify those preferences.

In the last few years, two main service recommendation techniques have been proposed: namely *collaborative filtering* and *content-based recommendation*. Collaborative filtering [19], [20], [21], [8], [22] is a technique that has been widely used for recommending items (Web services in our case) to a given user by considering other similar users' ratings on the items. For instance, suppose that a user likes Web services s_a and s_b. If there are many other users who like s_a and s_b also like service s_c, then service s_c should probably be recommended to that user. Although this technique is effective, one big problem is that Web services without having a considerable set of user interactions (e.g., newly deployed Web services) cannot be recommended, which is also known as the *cold start* problem. On the other hand, content-based methods [23], [24], [25] recommend Web services based on the similarity of user preferences and the descriptive information of Web services (e.g., service functionalities). Newly-deployed Web services can be recommended by this technique. Unfortunately, associating user preferences with Web service content is not a trivial task and very few solutions have been proposed. In current Web service search engines, queries that represent user preferences are typically prepared by users, which may not be an easy task for the users.

In this paper, we propose a novel approach for Web service recommendation by unifying collaborative filtering and content-based recommendation in a normative manner. Our approach exploits the advantages of both techniques that enables more accurate recommendations with a rich variety. More specifically, our approach is built on a three-way aspect model [26] that directly represents unobservable user preferences as a set of latent variables, which can be statistically estimated using algorithms such as *expectation maximization* (EM) [27]. In a nutshell, the main contributions of our work are as the following:

- We conduct an extensive analysis on the existing recommender systems as well as the tasks on Web service recommendation. We identify three main requirements, including *high recommendation accuracy*, *recommendation serendipity*, and *recommending newly-deployed services*. It is important to consider these requirements for designing and developing effective Web service recommender systems.
- We propose a novel hybrid approach that combines collaborative filtering and semantic content-based methods for service recommendation. Our approach exploits a *three-way aspect* model that simultaneously considers the similarities of users and semantic content of Web services. User preferences are represented using a set of latent variables that can be statistically estimated. We further develop two strategies (*data smoothing* and *implicit user-descriptor aspect model*) to specifically deal with the overfitting problem caused by data sparsity.
- We conduct extensive experiments using real-world Web services to verify the proposed approach. A dataset [8] consisting 5,825 Web services are carefully examined and 3,693 live Web services are selected and used in the experiments. The experimental results show that our approach achieves better recommendation performance than the conventional collaborative filtering and content-based methods applied separately.

The remainder of the paper is organized as follows. Section 2 discusses Web service recommendation requirements and overviews two complementary recommendation approaches. Section 3 introduces our unified Web service recommendation approach. Section 4 describes techniques dealing with data sparsity. Section 5 reports our experimental results. Finally, Section 6 overviews the related work and Section 7 offers some concluding remarks.

2 Web Service Recommendation

In this section, we discuss the main requirements on Web service recommendation that we target in this paper, formalize the service recommendation task, and briefly introduce two typical recommendation approaches: collaborative filtering and content-based recommendation.

2.1 Requirements in Service Recommendation

Accuracy is an important metric when assessing a recommender system. However, there is an increasing awareness that good accuracy alone does not necessarily give users an effective and satisfying experience. From the analysis of the existing recommender systems Herlocker et al. [28] and our consideration of the specific tasks of service recommendation, we identify the following three major requirements in order to conduct an effective service recommendation task:

- *High recommendation accuracy.* A good recommender system should recommend more relevant Web services and fewer irrelevant ones, particularly in the situations where required information are not available (e.g., missing QoS of some services) [8].
- *Recommendation serendipity.* Recommending services that are already known to a user can be found unsatisfactory or meaningless. If the recommended services are not familiar to the user, the chances of finding new Web services that match the user's requirements would increase [28].
- *Recommending newly deployed services.* Overcoming the cold-start problem not only enables users to find newly-deployed Web services, but also enhances the recommendation serendipity.

As discussed in the introduction, neither collaborative filtering nor content-based recommendation can satisfy all the three requirements. Our approach will unify both methods for effective service recommendation. In the rest of this section, we will first formalize the service recommendation task, and then briefly introduce collaborative filtering and content-based recommendation.

2.2 Service Recommendation Task

Let $\mathcal{U} = \{u|1, \ldots, \mathcal{N}_u\}$ and $\mathcal{S} = \{s|1, \ldots, \mathcal{N}_s\}$ be the set of users and Web services in a recommender system respectively. Here $\mathcal{N}_u$ and $\mathcal{N}_s$ are the number of the users and the services. The relationship between service users and Web services can be denoted by a user-item matrix $\mathcal{R}$:

$$\mathcal{R} = \{r_{u,s}|1 \leq u \leq \mathcal{N}_u, 1 \leq s \leq \mathcal{N}_s\} \tag{1}$$

An entry in $\mathcal{R}$, denoted by $r_{u,s}$, represents a vector of QoS values (e.g., response time) of service s that is observed by the service user u. When user u has not invoked a service s, $r_{u,s} = \emptyset$. It should be noted that most entries in $\mathcal{R}$ are empty in real-world applications. The reason is that the number of services invoked by each user is usually very small. Collaborative filtering based approaches use $\mathcal{R}$ for service recommendation.

In content-based recommendation, the content of each service is represented as a vector of several features extracted from e.g., the WSDL file and the short service description. Let $\mathcal{N}_f$ be the number of features and $c_{s,t}$ be the value of the t^{th} feature of service s. By collecting all the feature vectors, we can have the content matrix $\mathcal{C}$ as:

$$\mathcal{C} = \{c_{s,t}|1 \leq s \leq \mathcal{N}_s, 1 \leq t \leq \mathcal{N}_f\} \tag{2}$$

Given a target user u, content-based approaches use $\mathcal{C}$ for service recommendation. Unlike collaborative-based approaches, they do not consider the results from other users. We will discuss further the two different kinds of approaches in the following sections.

2.3 Collaborative Filtering

Collaborative filtering (CF) predicts rating scores of a user for Web services by considering other users' rating on the services. CF is generally decomposed into two techniques: *memory-based* and *model-based*. Memory-based CF consists of user-based approaches that predict the ratings of active users based on the ratings of similar users found, while item-based CF methods predict the ratings of active users based on the computed information of items similar to those chosen by the active user. Pearson Correlation Coefficient is widely used to calculate similarities between users and predicts QoS values based on similar users in memory-based CF [29], [30], [8]. Based on the predicted QoS values, the Web service with the best score or the top n Web services are selected for the recommendation. The probability of a service s being recommended to a user u can be calculated using this method as the following:

$$\hat{y}_{u,s} = \frac{\sum_{u' \in \mathcal{U}} sim_{u,u'} y_{u',s}}{\sum_{u' \in \mathcal{U}} sim_{u,u'}} \tag{3}$$

where $y_{u',s}$ is the estimated value, and $sim_{u,u'}$ measures the preference similarity of users u and u', using the following formula:

$$sim_{u,u'} = \frac{\sum_{s \in \mathcal{S}} (r_{u',s} - \overline{r'_u})(r_{u,s} - \overline{r_u})}{\sqrt{\sum_{s \in \mathcal{S}} (r_{u',s} - \overline{r'_u})^2} \sqrt{\sum_{s \in \mathcal{S}} (r_{u,s} - \overline{r}_u)^2}} \tag{4}$$

Where $r_{u',s}$ is the score given to service s by user u', $\overline{r_u}$ and $\overline{r_{u'}}$ represent the average rating values of user u and u' respectively ($u, u' \in \mathcal{U}$), and $s \in \mathcal{S}$ is the Web service rated by both users u and u'.

Matrix factorization is a main method in model-based recommendation systems, which performs well in handling large scale user-item interaction matrix. This technique aims at factorizing the user-item interaction matrix into two low-rank approximations in the user space U_s and the item space V_s respectively, and utilizing the low-rank matrices to make predictions [31], [32]. The general matrix factorization method is employed to estimate the user-item rating matrix $\mathcal{R}$, which is approximated by multiplicating a low-rank factors multiplication $\mathcal{R} = U_s^T V_s$, where $U_s \in \mathbb{R}^{m \times d}$ and $V_s \in \mathbb{R}^{n \times d}$ by minimizing the objective function with regularization:

$$\min_{U_s, V_s} \frac{1}{2} \sum_{u}^{m} \sum_{s}^{n} \mathbb{I}_{u,s} (r_{u,s} - \mathbf{u}_u \mathbf{v}_s)^2 + \frac{\lambda_{U_s}}{2} ||U_s||_F^2 + \frac{\lambda_{V_s}}{2} ||V_s||_F^2 \tag{5}$$

Here $\mathbb{I}$ is the indicator function which is the rating score of service s given by user u. λ_{U_s} and λ_{V_s} are the regularization parameters. Recently, several research projects have explored the combination of memory-based and matrix factorization together to enhance the prediction performance [33].

It should be noted that there are usually very few Web services (i.e., matrix $\mathcal{R}$ is sparse). Particularly when the space of Web services is large, the above formulas often fail. A possible solution is to replace the empty scores in QoS matrix with a default score. For example, if Web services are rated on a 1 to 5 scale, we could set the default value as 3. In Section 4, we will also introduce techniques to effectively increase the density of the data by exploiting secondary data of Web services.

2.4 Content-based Recommendation

Content-based Web service recommender systems recommend a target user with services that are similar to those previously preferred by the user. Such systems are based on the analysis of the similarities of the content (e.g., WSDLs and short descriptions) of the Web services. There have been two main approaches in content-based Web service recommendation: *syntactic* based approaches [24] and *semantic* based approaches [34]. We discuss only semantic based approaches in this paper since syntactic based approaches have obvious limitations in suggesting high quality recommendations.

The semantics of a Web service s can be represented by a set of semantic attributes: i) functional category $\mathcal{F}(s)$, ii) functional parameters (i.e., inputs $\mathcal{IP}(s)$ and outputs $\mathcal{OP}(s)$), and iii) requirements (i.e., preconditions $\mathcal{P}(s)$ and effects $\mathcal{E}(s)$). We assume that these attributes can be provided by a domain ontology through semantic annotations, which will ensure to provide users with recommendations that are semantically similar to Web services previously invoked. It is possible to construct a domain ontology

by analyzing Web service descriptions (WSDLs and free text descriptors). Interested readers can refer to [35] for an approach for bootstrapping ontologies based on Web service descriptions.

The semantic content similarity of Web services s_i and s_j can be calculated using:

$$q(s_i, s_j) = \sum_{l \in \{\mathcal{F}, \mathcal{IP}, \mathcal{OP}, \mathcal{P}, \mathcal{E}\}} w_l \times (q_{cd}(l(s_i), l(s_j)), q_m(l(s_i), l(s_j)) \quad (6)$$

where $w_l \in [0, 1]$ is the weight assigned to the l^{th} service description attribute and $\sum_{l \in \{\mathcal{F}, \mathcal{IP}, \mathcal{OP}, \mathcal{P}, \mathcal{E}\}} w_l = 1$. Preferences on some particular service attribute can be done by simply adjusting the value of w_l. The result returned by the formula is a pair of values in [0,1] $\times$ [0,1], representing the common description rate q_{cd} and matching quality q_m between s_i and s_j respectively. The matching quality between two semantic descriptions (i.e., $q_m(sd_i, sd_j)$) is a value in [0, 1] defined by a matchmaking function (i.e., 1 for exact match, and 0 for disjoint). The common description rate reflects the degree of similarity between the semantic descriptions of two Web services. Formally, the rate can be calculated using:

$$q_{cd}(sd_i, sd_j) = \frac{|lcs(sd_i, sd_j)|}{|sd_j \backslash sd_i| + |lcs(sd_i, sd_j)|} \quad (7)$$

where $lcs(sd_i, sd_j)$ is the least common subsumer of sd_i and sd_j, which refers to information shared by sd_i and sd_j. $sd_j \backslash sd_i$ represents all the information which is a part of sd_i but not a part of sd_j. The expression in between | refers to the size of ALE concept descriptions of DL (Description Logics) [36].

It should be noted that q_{cd} measures the proportion of descriptions that are in common for services s_i and s_j, while q_m does not measure the similarity between service descriptions but indicates a general overview of two services' semantic relationship (usually in discretized values) denoted in subsumption relationship. By combining these two measures, we can have a more accurate estimation on the similarity of two semantic Web services. The interested readers are referred to [37] for more details on calculating semantic similarities of Web services.

3 The Unified Service Recommendation Model

To meet the three requirements described in Section 2.1, we propose a unified approach that combines collaborative filtering technique and content-based approach. In this hybrid approach, it is necessary to reflect both rating and content data in modeling of user preferences. Unfortunately, user preferences are only indirectly represented and the observable data such as ratings or content (e.g., semantic descriptions) do not completely reflect the preferences.

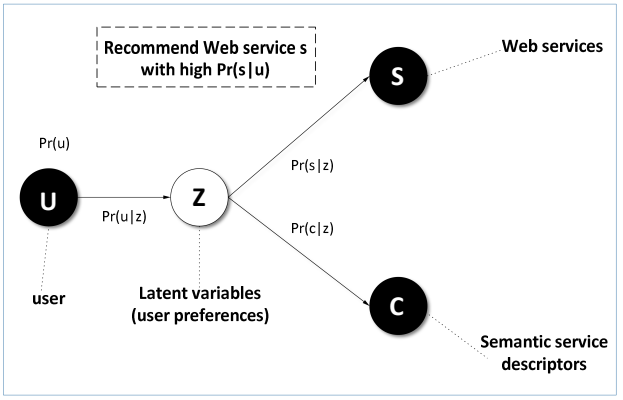


Fig. 1. Graphical representation of our approach

To solve the problem, we propose to use a probabilistic model that associates ratings and content data with newly-introduced variables that represent user preferences. In particular, we adapt the *three way aspect model* [26] in our unified Web services recommendation. This model assumes that users have some latent preference in terms of functional and non-functional attributes (e.g., QoS values), from which the desirable services and their content information can be deduced. The preferences are statistically estimated using *expectation maximization* (EM) [38] that thereafter contribute to better recommendation. In the rest of this section, we will describe how to adapt this model for Web service recommendation.

3.1 Probabilistic Recommendation Model

The graphical representation of the three-way aspect model for Web service recommendation can be found in Figure 1. The model includes four components: a user set $\mathcal{U} = \{u_1, u_2, ..., u_{\mathcal{N}_u}\}$, a Web service set $\mathcal{S} = \{s_1, s_2, ..., s_{\mathcal{N}_s}\}$, semantic content of Web services $\mathcal{C} = \{c_1, c_2, ..., c_{\mathcal{N}_f}\}$ where c_i is a semantic descriptive vectors of Web service s_i, and a set of latent variables $\mathcal{Z} = \{z_1, z_2, ..., z_{\mathcal{N}_z}\}$ that governs the recommendation process, e.g., users' latent preferences.

The model captures a three-way co-occurrence data among users, Web services, as well as the functionality attributes of services such as content of Web services in the form of semantic descriptions, and non-functionality such as rating score in term of QoS values. An observation is typically a triple (u, s, c) that corresponds to an event where a user u accesses a Web service s that contains a semantic service description c. In the three-way aspect model, observation data is associated with one of the latent variables ($z_i \in \mathcal{Z}$). The latent variables represent user latent preferences to Web services, e.g., preferences on functionalities or non-functionalities (QoS values). Each latent variable, z_i, actually corresponds to a "genre" of a service. A set of proportions of such "genres" (i.e., $\mathcal{Z}$) reflects a preference on services (similar to the taste in terms of music) of a user. It is assumed that users,

Web services, and semantic descriptors are independent in the model. It is also worth noting that the aspect model allows multiple semantic descriptions per user, unlike most clustering methods that assign each user with a single class.

In the context of Web service recommendation, an event of a user $u \in \mathcal{U}$ accessing a service $s \in \mathcal{S}$ containing semantic description $c \in \mathcal{C}$, is considered to be associated with one of the latent variables $z \in \mathcal{Z}$. Conceptually, users choose (latent) topics z, which in turn "generates" both Web services and their content descriptions. Therefore, a latent variable in this new model is not only associated with a distribution of services but also a distribution of semantic service content. The joint probability distribution $Pr(u, s, c, z)$ over user set $\mathcal{U}$, latent topic variables $\mathcal{Z}$, Web service set $\mathcal{S}$ and service content $\mathcal{C}$ is given by

$$Pr(u, s, c, z) = Pr(u)Pr(z|u)Pr(s, c|z) \tag{8}$$

Since we consider the distribution of s and c are independent in our model, we can have $Pr(s, c|z) = Pr(s|z)Pr(c|z)$. The above equation can be rewritten as:

$$Pr(u, s, c, z) = Pr(u)Pr(z|u)Pr(s|z)Pr(c|z) \tag{9}$$

An equivalent specification of the joint probability distribution that treats users and items symmetrically is:

$$Pr(u, s, c, z) = Pr(z)Pr(u|z)Pr(s|z)Pr(c|z) \tag{10}$$

Marginalizing out z, we obtain the joint probability distribution $Pr(u, s, c)$ over $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{C}$ as the following:

$$Pr(u, s, c) = \sum_{z} Pr(z)Pr(u|z)Pr(s|z)Pr(c|z) \tag{11}$$

This model has a set of parameters $Pr(z)$, $Pr(u|z)$, $Pr(s|z)$ and $Pr(c|z)$, which for simplicity is represented as θ. The model parameters are learned by mining the user-service history data $\mathcal{H} = \{< u, s, c >\}$. One way to learn θ is to maximize the log-likelihood of history data which is:

$$\mathcal{L}(\theta) = \sum_{<u,s,c> \in \mathcal{H}} n(u, s, c) \log(Pr(u, s, c|\theta)) \tag{12}$$

where $n(u, s, c)$ indicates how much a user u prefers the semantic descriptor c in Web service s. In general, $n(u, s, c) = r(u, s) \times n(s, c)$, where $r(u, s)$ is the rating score of user u for service s (e.g., via aggregating the QoS values such as response time and throughput), and $n(s, c)$ represents the weight of semantic descriptor c in Web service s.

The weight of a particular semantic descriptor in a Web service is calculated using widely-used TF/IDF. In particular, the term frequency of a semantic descriptor c is defined as:

$$tf(c) = \frac{freq(c, s)}{|s|} \tag{13}$$

where $freq(c, s)$ denotes the number of times semantic descriptor c occurs in service s description. The inverse document frequency idf is calculated as the ratio between the total number of services and the number of services that contain the semantic descriptor:

$$idf(c) = \begin{cases} \log \dfrac{|\mathcal{S}|}{|\{\mathcal{S}_i : c \in \mathcal{S}_i\}|} & \text{if } |\mathcal{S}_i| \neq 0 \\ \log \dfrac{|\mathcal{S}|}{1 + |\{\mathcal{S}_i : c \in \mathcal{S}_i\}|} & \text{if } |\mathcal{S}_i| = 0 \end{cases} \tag{14}$$

where $|\mathcal{S}|$ is the number of services, $\mathcal{S}_i$ denotes the number of services where semantic descriptor c appears. $n(s, c)$ is then calculated by using:

$$n(s, c) = tf(c) \times idf^2(c) \tag{15}$$

It should be noted that we choose to give higher weight to the idf value (i.e., idf^2). The reason behind this modification is to normalize the inherent bias of the tf measure in short documents [35].

3.2 Learning Process

As mentioned before, one way to learn θ (see Equation 12) is to maximize the log-likelihood of the history data. In our approach, the Expectation-Maximization (EM) algorithm is applied to learn the parameters. EM is an iterative optimization method of learning the probabilistic model parameters from incomplete data. The algorithm is implemented in two steps: i) the Expectation step, namely E-step, which calculates the expected value of latent variables z based on observations, and ii) the Maximum step, namely M-step, which is implemented to lift the lower bound of model parameters by utilizing the result obtained from the E-step. The detailed learning process can be described as the following:

$$\begin{aligned} &\sum_{<u,s,c> \in \mathcal{H}} \log Pr(u, s, c|\theta) \\ &= \sum_{<u,s,c> \in \mathcal{H}} \log \sum_{z} Pr(u, s, c, z|\theta) \\ &= \sum_{<u,s,c> \in \mathcal{H}} \log \left(\sum_{z} Pr(z|u, s, c, \theta^{(t)}) \frac{Pr(u, s, c, z|\theta)}{Pr(z|u, s, c, \theta^{(t)})} \right) \\ &\geq \sum_{<u,s,c> \in \mathcal{H}} \sum_{z} Pr(z|u, s, c, \theta^{(t)}) \log \left(\sum_{z} Pr(z|u, s, c, \theta^{(t)}) \right. \\ &\left. \frac{Pr(u, s, c, z|\theta)}{Pr(z|u, s, c, \theta^{(t)})} \right) \triangleq \mathcal{Q}(\theta|\theta^{(t)}) \end{aligned} \tag{16}$$

Therefore, instead of maximizing $\mathcal{L}(\theta)$ directly, the EM algorithm tries to find the model parameters $\theta^{(t+1)}$ to

maximize $\mathcal{Q}(\theta|\theta^{(t)})$. So:

$$\begin{aligned}\theta^{t+1} &= \arg\max\{\mathcal{Q}(\theta|\theta^{(t)})\} \\ &= \arg\max_{\theta}\left\{\sum_{<u,s,c>\in\mathcal{H}}\sum_{z} Pr(z|u,s,c,\theta^{(t)})\right. \\ &\quad \left.\log Pr(u,s,c,z|\theta)\right\} \\ &= \arg\max_{\theta}\left\{\sum_{<u,s,c>\in\mathcal{H}} \mathbb{E}_{z|u,s,c,\theta^{(t)}}\right. \\ &\quad \left.\{\log Pr(u,s,c,z|\theta)\}\right\}\end{aligned} \tag{17}$$

At this point, we can use the EM algorithm to solve Equation 17 with training dataset. In particular, the E step and the M step are iterated alternately until the log-likelihood $\mathcal{L}$ converges to a local maximum. It should be noted that both content and collaboration data can influence recommendations. The relative weight of each type of data depends on the nature of the given data for training.

The E-step is used to obtain the posterior probabilities in Equation 16 by calculating $Pr(z|u,s,w,\theta^{(t)})$, where the model parameters θ^t are known in this step:

$$Pr(z|u,s,c,\theta^t) = \frac{Pr(z)Pr(u|z)Pr(s|z)Pr(c|z)}{\sum_z Pr(z)Pr(u|z)Pr(s|z)Pr(c|z)} \tag{18}$$

In the M-step, we need to find new model parameters to maximize the expected log-likelihood found in the E-step, since

$$\begin{aligned}\mathcal{L}(\theta) &= \log Pr(u,s,c,z|\theta) \\ &= \log Pr(u|z) + \log Pr(s|z) + \\ &= \log Pr(c|z) + \log Pr(z)\end{aligned} \tag{19}$$

So, the maximization to the model parameters $\theta^{t+1} = \{Pr(u|z), Pr(s|z), Pr(c|z), Pr(z)\}$ can be obtained by maximizing the expectation with respect to θ:

$$\begin{aligned}Pr(u|z) &\propto \sum_{s,c} n(u,s,c)Pr(z|u,s,c) \\ Pr(s|z) &\propto \sum_{u,c} n(u,s,c)Pr(z|u,s,c) \\ Pr(w|z) &\propto \sum_{u,s} n(u,s,c)Pr(z|u,s,c) \\ Pr(z) &\propto \sum_{u,s,c} n(u,s,c)Pr(z|u,s,c)\end{aligned} \tag{20}$$

After the model is learned, the inference of Web services can be ranked for a given user according to $Pr(s|u) \propto \sum_c Pr(u,s,c)$, i.e., according to how likely it is that the user will invoke the corresponding Web service. Web services with high $Pr(s|u)$ that the user has not yet invoked are good candidates for recommendation. This addresses the requirement of *recommendation serendipity* discussed in Section 2.1. In addition, since the model considers the content of Web services, the cold-start problem can also be solved (i.e., newly-deployed Web services can be recommended).

It is noted that the likelihood needs to iterate over all possible data connections of each data sample, which might be computationally expensive. For example, in the E-step of the algorithm, we need to calculate the expectation posterior distribution of $Pr(z|u,s,c)$, given the current estimated parameters $Pr(u|z)$, $Pr(s|z)$ and $Pr(c|z)$. This computation is heavy due to standard least-squares computation, which is estimation of the regression coefficients of the factors on the variables assuming that the current estimated $Pr(u|z)$, $Pr(s|z)$ and $Pr(c|z)$ are found from the M-step. The time complexity of implementing the EM algorithm is $\mathcal{O}(\mathcal{N}\cdot|c|\cdot\mathcal{N}_z)$, where $\mathcal{N}$ is the size of the training dataset, $|c|$ is the length of descriptive vectors of services and $\mathcal{N}_z$ is the number of latent variables. Another concern is that although the EM algorithm is guaranteed to be stable and to converge to a local maximum value of the estimated likelihood which depends on the initial data, there is no guarantee that this value is globally maximum.

In reality, the user-service interaction matrix could be very sparse, in other words, the users only invoke a few part of services. When data is extremely sparse, which is typical in many real-world applications, the EM algorithm will suffer from overfitting, i.e., poor generalization. We will discuss two strategies in the next section that can effectively increase the data density, which in turn improves the learning performance of the EM algorithm.

4 Dealing with Data Sparsity

In this section, we describe two strategies in overcoming the overfitting problem caused by sparse data. The first strategy is to preprocess data matrix using a data smoothing technique. The second strategy is to modify the three-way aspect model by eliminating Web services from direct participation in the model.

4.1 Data Smoothing

The idea of the data smoothing strategy to addressing the overfitting problem with sparse data is to smooth the data matrix based on the similarities between Web services. The intuition behind is as follows. Consider a user u who has invoked service s_i. Assume that another service s_j has not been invoked by u, but s_i and s_j are very similar in content (e.g., both services share many service attributes). Informally, if the content similarity function (see Equation 6), $q(s_i, s_j)$, yields 0.85, we could believe that there is a 85% chance that user u has actually invoked service s_j, even though the recommender system does not know it.

Based on this reasoning, we propose to *preprocess* the original rating matrix $\mathcal{R}$ in terms of functionality and non-functionality by filling in some of the empty entries with the average similarities above a certain threshold between a Web service and all other services invoked by user u. Clearly, when the threshold is bigger, the data matrix will

become sparser (i.e., less empty entries will be replaced). On the other hand, if the threshold is smaller, the data matrix will become less sparse (i.e., more empty entries will be replaced). By setting appropriate threshold, the density of the data matrix (i.e., the fraction of non-zero entries) can be effectively increased, as a result of the data smoothing.

It should be noted that the semantic content of Web services and the calculation of the similarities between Web services can be found in Section 2.4.

4.2 Implicit User-Descriptor Aspect Model

A Web service contains multiple semantic service descriptors and a service descriptor is contained in many Web services. Another method to overcome the overfitting problem due to sparsity is to propose a model where co-occurrence data points represent events that correspond to users looking for service descriptors in a Web service, i.e., (u, c). This is different from the model in Section 3 where the event corresponds to a user accessing a Web service with service descriptors, (u, s).

This modified aspect model produces estimates of conditional probabilities $Pr(u|z)$ and $Pr(c|z)$, as well as the latent variable priors $Pr(z)$. $Pr(u|c)$ can be calculated using:

$$Pr(u, c) = \sum_{z} Pr(z)Pr(u|z)Pr(c|z) \tag{21}$$

However, the task of a recommendation system is to recommend Web services that have the highest estimating probabilities $Pr(s|u)$ for a given user u. We can solve this problem by treating a Web service as a *bag of service descriptors*: the probability of a Web service is the product of the probabilities of the semantic service descriptors it contains adjusted for different service description lengths with geometric mean:

$$Pr(s, u) \propto (\prod_{i} Pr(c_i, u)^{1/|c|}) \tag{22}$$

where c_i are semantic service descriptors in a Web service s and $|c|$ is the length of the service description of s. Conditional probabilities $Pr(c_i, u)$ follow directly from the model:

$$Pr(c_i, u) = \frac{Pr(u, c_i)}{\sum_c Pr(u, c)} \tag{23}$$

5 Performance Evaluation

This section focuses on reporting the performance study of our proposed hybrid approach for Web service recommendation, including two experiments: i) comparing our hybrid approach with the state-of-the-art methods including three collaborative filtering (CF) approaches and content-based recommendation, and ii) studying the sensitivity of the hybrid approach under different markoff ratios and different number of latent variables. We first describe the dataset collected for the experiments and then report the experimental results.

5.1 Dataset Setup

To perform reliable experiments, it is ideal to use large-scale real world Web services. Unfortunately, collecting and preparing such data is extremely time-consuming. Fortunately, Zheng et al. [8] shared a large-scale real world Web services dataset collected in their WS-DREAM project. WS-DREAM is a Web crawling engine that crawled publicly available WSDL file addresses from the Internet. It also collected non-functional attributes (e.g., QoS) of these Web services, which are observed by 339 distributed computers located in 30 different countries, from PlanetLab[4]. The first half of Table 1 summarizes this dataset and Figure 2 depicts the distribution of services and users (i.e., 339 computers).

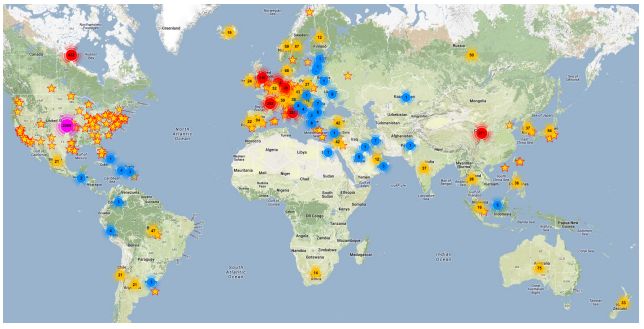


Fig. 2. Distribution of Web services and users (computers) of the WS-DREAM dataset: the circles denote Web services at different locations and the stars represent the users.

In our experiments, we used this dataset as our base dataset and performed some pre-processing activities. Firstly, we traversed all 5,825 WSDL addresses offered from the dataset and retrieved WSDL documents of 3,693 live Web services. The rest of the Web services from WS-DREAM dataset were considered dead due to unsucessful connections to these services[5]. We then generated an ontology by exploiting the approach developed in our previous work [35]. This ontology was bootstrapped by analyzing WSDL files and short textual descriptions of Web services using Term Frequency/Inverse Document Frequency (TF/IDF) and web context generation. The concepts from this ontology were used to annotate each Web service, which in turn generated the semantic description for the service. Consequently, each Web service is represented by a set of semantic service descriptors. Secondly, we collected the corresponding rating scores of Web services from websites such as `seekda` Web service search engine[6], `WebServiceLists`, and `ProgrammableWeb`. Due to different rating scales used in these websites, we normalized the rating scores between the range of 1 to 5.

For those Web services whose rating scores are not available, we determined their rating scores (between 1 and 5) based on their aggregated QoS values (e.g., response time,

4. http://www.planet-lab.org
5. This task was carried out in April 2012.
6. Its site webservices.seekda.com is unavailable as of 08/05/2014.

TABLE 1
Dataset Statistics

Original Dataset	
Number of Users	339
Number of Web Services	5825
User-Service (Response Time) Matrix Density	5.11×10^{-2}
User-Service (Throughput) Matrix Density	7.26×10^{-2}
Processed Dataset	
Number of Users	572
Number of Web Services	3693
User-Service (Rating) Matrix Density	7.67×10^{-2}
Average Number of Semantic Descriptors for Each Service	12.79
Average QoS Ratings	3.84

throughput) using a multi-attribute utility function [39], [40]. More specifically, we exploited two sets of QoS data from WS-DREAM dataset: the response time matrix and the throughput matrix (see Figure 3). For each of our 3,693 Web services, we extracted its response time and the throughput from the dataset. The aggregated QoS value of a service s can be calculated using:

$$\mathcal{U}(s) = \sum_{i \in \mathcal{A}} w_i \cdot Score_i(s) \tag{24}$$

where:

- $Score_i(s)$ is a QoS attribute scoring function, which, given a value of a QoS attribute i of the service s, returns a score (a positive integer value). $\mathcal{A}$ is the set of QoS attributes.
- w_i is the weight assigned to the QoS attribute i, and
- $w_i \in [0, 1]$ and $\sum_{i \in \mathcal{A}} w_i = 1$.

A scoring function is provided for each QoS attribute (in our case, response time and throughput) that calculates the score of the attribute for a particular service and scales the score to the interval [1..5]. A higher score value indicates a better quality of the service. However, we draw attention that some of the QoS attributes are *negative* (e.g., response time), i.e., the higher the value is, the lower the quality is. While others are *positive* (e.g., throughput), i.e., the higher the value is, the higher the quality is. Therefore the attribute scores should be scaled differently.

The rating scores from the above aggregated QoS values were used for Web services for which we could not find a rating score from the Web sites. After the processing, we obtained the new dataset that was devoted to our experimental studies. The second half of Table 1 shows the statistical information of the new dataset. It should be noted that the users in this new dataset were eventually a combination of the real users (who gave rating scores on the Web sites) and the computers used in the WS-DREAM project (that collected QoS values).

5.2 Metrics

We used the micro-F1 and macro-F1 as the evaluation measures in our experiments. The F1 measure is the harmonic mean of $Precision(P)$ and $Recall(R)$, which can

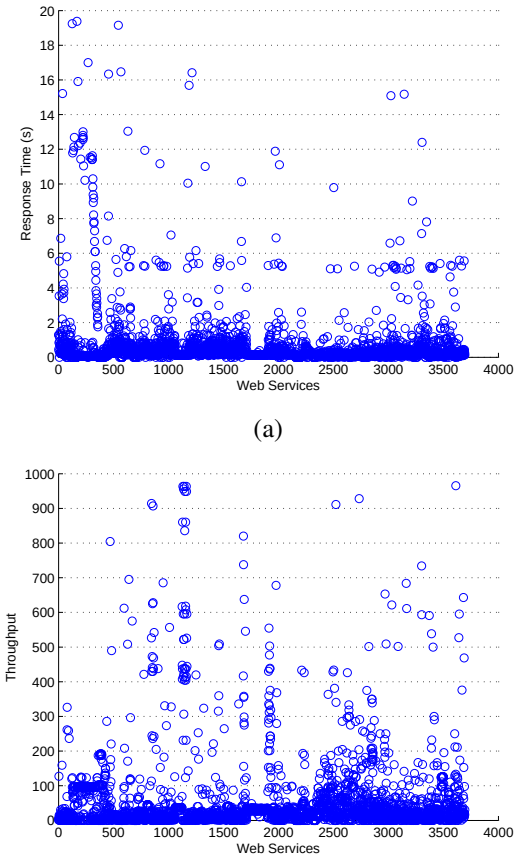


Fig. 3. QoS data collection in the WS-DREAM dataset (a) response time and (b) throughput

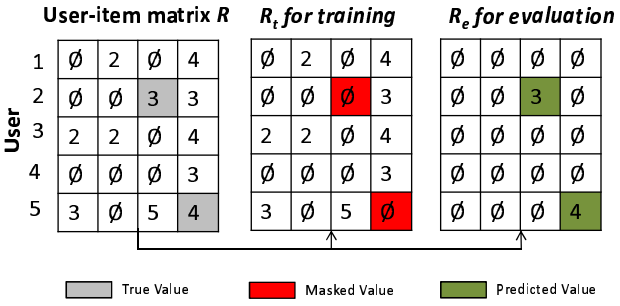


Fig. 4. Illustration of experiment implementation: we divided rating matrix $\mathcal{R}$ into the training matrix $\mathcal{R}_t$, in which partial rating scores are masked and the rest ratings are used to train our model, and evaluation matrix $\mathcal{R}_e$, in which the masked rating value can be predicted using our validated model.

be calculated as: $F_1 = 2\frac{P \times R}{P + R}$. The Micro-F1 is defined as:

$$Micro - F1 = \frac{2\sum_{j=1}^{c}\sum_{i=1}^{n} \hat{y}_i^c y_i^c}{\sum_{j=1}^{c}\sum_{i=1}^{n} \hat{y}_i^c + \sum_{j=1}^{c}\sum_{i=1}^{n} y_i^c} \tag{25}$$

TABLE 2
Performance Comparison with Other Approaches

	Training Ratio	10%			20%			30%			40%		
	Top N	N = 1	N = 5	N = 10	N = 1	N = 5	N = 10	N = 1	N = 5	N = 10	N = 1	N = 5	N = 10
Micro-F1	UCF	0.6133	0.5683	0.5542	0.6329	0.6269	0.6002	0.7288	0.6767	0.6271	0.7570	0.6925	0.6507
	ICF	0.5779	0.5602	0.5363	0.6216	0.6167	0.5838	0.7011	0.6352	0.6124	0.7214	0.6564	0.6367
	CBR	0.5845	0.5794	0.5559	0.6189	0.6117	0.5893	0.6329	0.6196	0.6072	0.6672	0.6426	0.6244
	LFM	0.6092	0.5615	0.5403	0.6654	0.6361	0.6189	0.7312	0.6895	0.6333	0.7369	0.7204	0.6963
	HR	**0.6784**	**0.6383**	**0.6225**	**0.7126**	**0.6773**	**0.6486**	**0.7511**	**0.7248**	**0.6917**	**0.7772**	**0.7511**	**0.7221**
Macro-F1	UCF	0.6023	0.5494	0.5353	0.6314	0.6027	0.5897	0.7153	0.6635	0.6213	0.7325	0.6892	0.6427
	ICF	0.5672	0.5512	0.5275	0.6268	0.5905	0.5764	0.6894	0.6501	0.6044	0.7124	0.7628	0.6303
	CBR	0.5822	0.5531	0.5462	0.6006	0.5943	0.5786	0.6205	0.6072	0.5977	0.6424	0.6278	0.6025
	LFM	0.5881	0.5552	0.5285	0.6532	0.6246	0.6024	0.7105	0.6739	0.6223	0.7273	0.7131	0.6793
	HR	**0.6627**	**0.6221**	**0.6167**	**0.7007**	**0.6712**	**0.6265**	**0.7436**	**0.7222**	**0.6879**	**0.7715**	**0.7463**	**0.7233**
nDCG	UCF	0.6279	0.5986	0.5312	0.6320	0.6337	0.6004	0.7323	0.6663	0.6207	0.7417	0.6791	0.6358
	ICF	0.5916	0.5578	0.5236	0.6172	0.6004	0.5866	0.7115	0.6520	0.6024	0.7166	0.6892	0.6152
	CBR	0.5673	0.5422	0.5103	0.5899	0.5947	0.5613	0.6325	0.6164	0.5993	0.6526	0.6318	0.6093
	LFM	0.6102	0.5786	0.5403	0.6584	0.6312	0.6057	0.7233	0.6944	0.6326	0.7574	0.7285	0.6972
	HR	**0.6762**	**0.6325**	**0.6263**	**0.7213**	**0.6758**	**0.6475**	**0.7625**	**0.7339**	**0.7081**	**0.7837**	**0.7549**	**0.7358**

where n is the number of test data, y_i is the true label vector of the i-th sample, $y_i^j = 1$ if the instance belongs to category j, -1 otherwise. $\hat{y}_i$ is the predicted label vector. The micro-F1 measure weights equally on all samples, thus favoring the performance on common category labels. Macro-F1 is calculated as mean arithmetical value for F1 on each label. It measures weights equally on all the category labels regardless of how many samples belong to it, thus favoring the performance on rare category labels. Macro-F1 can be calculated using:

$$Macro - F1 = \frac{2\sum_{i=1}^{n} \hat{y}_i^c y_i^c}{n^2|c|(\sum_{i=1}^{n} \hat{y}_i^c + \sum_{i=1}^{n} y_i^c)} \tag{26}$$

To achieve more accurate evaluation, we considered the ranking position in the recommendation results and adopted the normalized Discounted Cumulative Gain (nDCG) as a metric, which is the normalized position-discounted precision score. It gives more credit to top-ranked Web services and is defined as below:

$$DCG_x = \sum_{i=1}^{x} \frac{2^{rel_i} - 1}{\log_2(1 + p_i)} \tag{27}$$

where p_i is the ranking position of s_i in top@x Web services, and rel_i is the relatedness score of s_i at position p_i. The normalized DCG can be calculated using:

$$nDCG_x = DCG_x / IDCG \tag{28}$$

where $IDCG$ is the maximum possible DCG till ranking position p_i of the sorted result list. The value range of nDCG is $[0, 1]$.

5.3 Performance Comparison

This experiment studies the recommendation performance of our proposed hybrid recommendation approach (HR) with the pure collaborative filtering methods (CF) and pure content-based recommendation (CBR). The recommendation accuracy was evaluated by examining the Top-N rankings for all users. Top-N recommendation is to recommend the N top-ranked items that will be of interest to a given user. For collaborative filtering methods, Top-N recommendation techniques analyze the user-item matrix to discover relations between different users or items, which are used to compute the recommendations. In our experiments, we considered three collaborative filtering methods [41]:

- User-based Collaborative Filtering (UCF): This method first calculates the k most similar users for a given user based on the Pearson correlation. The corresponding rows of the k similar users in the user-item matrix are aggregated to identify a set of service items, which have been invoked by the group of users together with their ratings. With the identified service items, UCF then recommends the top-N highly-rated service items that the target user has not invoked.
- Item-based Collaborative Filtering (ICF): Unlike UCF, this method first discovers k most similar service items for each service item based on similarities. It then identifies the set as candidates of the recommended services by taking the union of the k most similar items and removing each of the items in the set which the user has already invoked. The resulting set of the service items will be sorted in decreasing order based on the similarities and ICF then recommends the top-N service items to a given user.
- Latent Factor Model (LFM): LFM [22] exploits matrix factorization, a main method in model-based CF recommender systems, based on singular value decomposition (SVD) with regularization (see Section 2.3 for some introduction on matrix factorization).

In the experiment, the user-service interaction matrix (see Section 5.1) was randomly divided into the evaluation data matrix, $\mathcal{R}_e$, and the training data matrix, $\mathcal{R}_t$, by masking $p\%$ (also called *markoff ratio*) actual values of historical service invocation for each user. Figure 4 shows an illustrative example in dividing a simple user-service matrix (5 users and 4 service items). In the example, some user-service historical interactions are masked randomly as the evaluation data (shaded boxes), the rest data is used for learning the model parameters. After each model is learned, we used the model parameters to find $\forall s$, $Pr(s|u)$ for all

users. The Web services in the testing dataset were ranked based on their $(Pr(s|u))$.

In the experiment, we set the size of latent users' preferences $\mathcal{N}_z$ is 50, based on our findings in Section 5.4. We compared the recommended services with four other methods and our model (HR) by employing 10, 20, 30, 40 percent sparsity of the training dataset respectively. Table 2 shows the results.

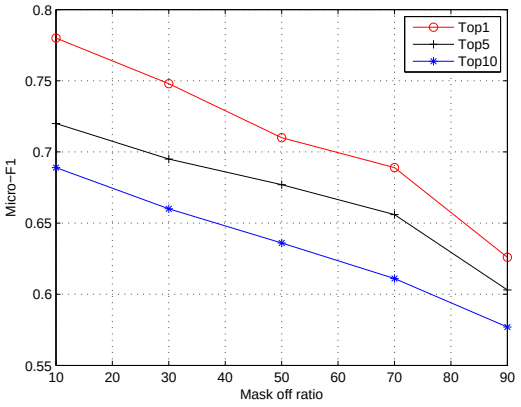
From Table 2 we can see that, the top N (N=1, 5, 10) performance of our approach (HR) are consistently higher than both UCF and ICF collaborative filtering methods, content-based recommendation approach (CBR), and regularized SVD (singular value decomposition) based latent factor model (LFM). It is clear that our approach outperforms the other approaches and more relevant Web services can be recommended. It also can be observed from the table that with the increase of the sparsity of our testing dataset[7], the recommendation performance of UCF, ICF, CBR and LFM decreases steadily, they can not handle the data sparsity issue very well. However, the performance of our proposed approach remains relatively stable and is not sensitive to the sparsity level changes. This is largely contributed by the two strategies dealing with data sparsity introduced in Section 4.

Another aim of this experiment is to evaluate our approach on handling the cold start problem, which refers to providing accurate prediction when some users only access few Web services or even have no access historical records at all. From the table, it is clear that our approach outperforms other methods on different markoff ratios, especially when the training ratios are small (e.g., 10%). We also notice that the performance of UCF, ICF and LFM degrades significantly when the training ratio is 10%. Although the CBR is not sensitive to the training ratio, its overall performance is poor.

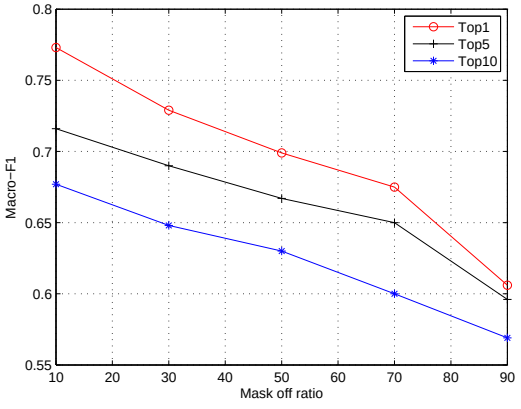
5.4 Impact of Markoff Ratio and Latent Variables

This section reports our experimental studies on the impact of markoff ratios and the number of latent variables to the performance of our proposed hybrid approach. As mentioned previously, we divided the whole matrix into training and evaluation matrices. In the first experiment, we studied the impact of the markoff ratios. We randomly masked $y\%$ (y = 10, 30, 50, 70) of actual scores and the rest of the matrix is used as training dataset to infer model parameters. Our algorithm was then used to recover the information that has been masked. We applied cross-validation method to find the average precisions and recalls for top N (N = 1, 5, 10) Web service recommendations. Figure 5 shows the result.

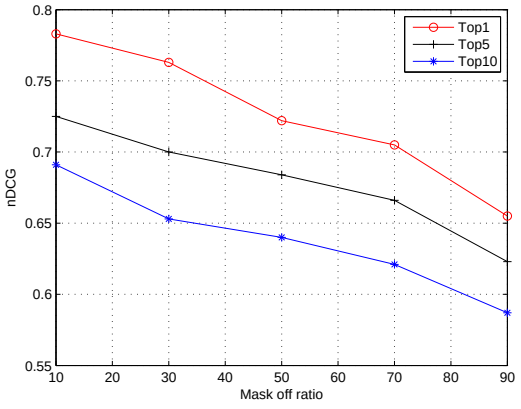
From the figure, we can see that with the increase of the markoff ratio, the overall recommendation performance decreases. This is attributed to the fact that a higher markoff ratio means less data available for training the approaches, therefore worse recommendation performance. Interestingly, we notice that the recommendation performance



(a)



(b)



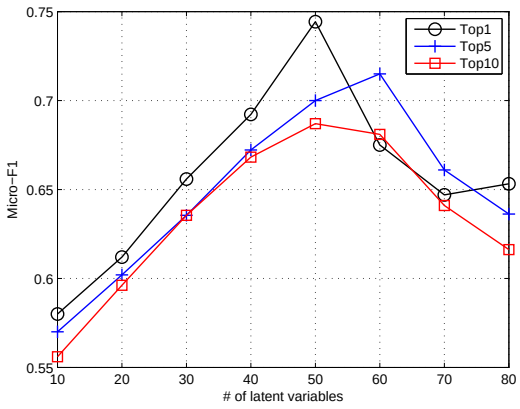
(c)

Fig. 5. Recommendation performance with different markoff ratios for top 1, 5, 10 Web services

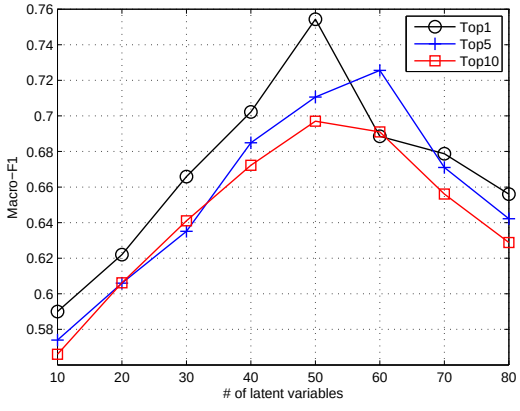
increases quickly when the markoff ratio decreases from 100% until it reaches 70% (i.e., training ratio of 30%). After that, the recommendation performance increases slowly when the markoff ratio decreases.

To study the impact of the latent variables of our model, we conducted similar experiments but masked 30% of actual scores of the matrix. We ran the algorithm using different number of latent variables $\mathcal{N}_z$ in the model, ranging

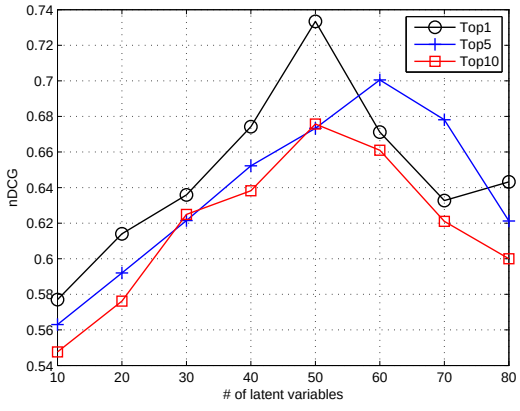
7. Note that the sparsity of $\mathcal{R}_e$ is 1 - markoff ratio.



(a)



(b)



(c)

Fig. 6. Recommendation performance with different number of latent variables $\mathcal{N}_z$ for top 1, 5, 10 Web services

from 10 to 80 with an interval of 10, and calculated Micro-F1, Macro-F1, and nDCG. Figure 6 shows the results. From the figure we can see that our system can mostly achieve the best recommendation performance around 50 latent variables. This is the reason that we chose $\mathcal{N}_z = 50$ in our other experiments.

We briefly discuss the threats to validity of our experimental results. Firstly, due to the lack of rating scores of some Web services, we used the aggregated QoS values as rating scores for these services by exploiting a multi-attribute utility function (see Section 5.1). To reduce this threat to construct validity, Web services can be manually supplied with rating scores in the future. We also plan to collect more Web services with rating scores. The threats to external validity primarily center on whether the Web services data set used in our experiments is representative in practice. In order to validate that our approach can be generalized, it is a necessity to conduct more experiments on randomly selected Web service datasets.

6 Related Work

Web service recommendation and selection has been a fundamental research issue since the dawn of Web service technologies. Traditional Web service discovery centers on UDDI registries such as the work presented in [42], [12], [43]. Unfortunately, UDDI is no longer the choice of publishing Web services. The available Web service search engines such as `XMethods` largely exploit keyword-based search techniques and are inadequate to match the functionalities of Web services. These search engines do not consider non-functional characteristics (QoS) of Web services. Furthermore, users normally have to know how to craft correct queries. The performance of Web service recommendation of these search engines are therefore limited. Over the past few years, service recommendation has been an active research area and many techniques have been proposed. These techniques can be classified into three categories: *collaborative filtering* (CF), *content-based*, and *hybrid* approaches. In the following discussions, we will focus on reviewing these techniques.

6.1 CF Methods for Service Recommendation

The collaborative filtering methods are widely used in recommender systems that recommend items (Web services in our context) based on the similarity of different users. A representative research effort in this area has been done by Zheng et al. [8], [18]. In their work, a QoS-based Web service prediction approach is proposed to predict missing QoS values based on the historical QoS information from similar Web services and users. QoS-based Web services selection supports optimized Web services selection by considering QoS attributes of Web services with similar functionalities, as well as the preferences from service users [39], [44], [45]. The quality of the recommendation from these approaches depends on the quality of available QoS information for Web services. Most QoS-based service selection approaches assume that the QoS information (e.g., availability of Web service) is pre-existing and readily accessible with guaranteed quality, which unfortunately is not always the case, as indicated by Zheng et al. in their work [8]. Service providers may not be able to deliver the QoS they promised and some QoS properties (e.g., network latency, invocation failure-rate, etc.) are highly related to the locations and network conditions of the service users. This makes such kind of approaches impractical to be used

in many applications. By combining the traditional user-based and item-based collaborative filtering methods, the approach proposed by Zheng et al. [8], [18] can overcome the problem and mostly importantly, does not require service invocations in order to obtain the values of user-dependent QoS properties.

The work by Chen et al. [19] presents RegionKNN, a collaborative filtering algorithm that is designed for large-scale Web service recommendation. This approach considers service users' physical locations and proposes a region model by considering the QoS characteristics of Web services. A refined nearest-neighbor algorithm is then developed for QoS-based service recommendation. In [46], Shao et al. propose a collaborative filtering based approach for mining users' similarities and predicting QoS values for Web services. In a very recent effort, Yu et al. [47] specifically tackle the data sparsity issue by proposing an algorithm based on the regularized Matrix Factorization.

Unfortunately, as discussed at the beginning of this paper, CF-based approaches have several inherent limitations. Since such approaches rely on interactions of Web services performed by other users, newly-deployed Web services cannot be recommended (i.e., the cold-start problem). In addition, the recommended services may be completely different (in terms of functionality) from the ones interacted by a given user in the past. In our work, we extend the collaborative filtering methods by considering semantic content similarities of services used by similar users. Our approach can effectively address the limitations of these CF-based service recommender systems.

6.2 Content-based Methods for Service Recommendation

The content-based approaches recommend items similar to those that a user appreciates based on the item's characteristics (e.g., functionalities). The cold-start issue can be successfully solved by content-based approaches. However, such approaches typically require end users to know what keywords to use for a specific kind of services, which can be difficult for end users.

The techniques on content-based recommendation can be classified into two categories: *syntactic* and *semantic* based approaches. Syntactic-based methods focus on string manipulation and thesauri approaches to correlate services discovery. Woogle [15] is one of the very earliest work in this direction. In Woogle, Dong et al. propose a Web service discovery approach based on matching users' requirements and the functionalities of Web services. In particular, with the help of the co-occurrence of the terms appearing in service inputs and outputs, names of operations and descriptions in services, the authors develop a set of similarity search primitives that algorithms can use to match Web services. In a recent effort, WS-Finder [5], Ma et al. improve existing Web service discovery techniques by employing the Earth Mover's Distance (EMD) for many-to-many partial matching between contents of user queries and service attributes. A k-NN algorithm is then used to produce top-k services for users. Blake and Nowlan [24] develop a Web service recommender system by exploiting an enhanced syntactic approach to compare the content of Web services. The approach aggregates and analyzes Web service messages and recommends services to end users. An interesting part of the work is the concept of *naming tendency* that is used to link strings from end users (e.g., queries) to the strings used in the definition of Web services (e.g., operation name, input and output). In general, syntactic-based approaches have limitations to suggest high quality recommendations.

In contrast, semantic-based methods recommend Web services by exploiting the semantic description of their functionalities using ontological descriptions [35]. OWL-S[8] is the first major ontology definition language for describing the semantics of services. There have been also efforts in developing languages for RESTful services such as the Web Application Description Language (WADL)[9] and SA-REST [48]. In [34], Lécué and Delteil exploit a semantic similarity measure in Web services selection and composition. However, most existing semantic-based approaches focus only on standard semantic reasoning (i.e., subsumption) when inferring semantic similarities. Similarities between other parts (e.g., preconditions and effects) are seldom considered. In [37], Lécué continues his work in this direction and develops a complete specification of semantic service description by considering different levels of service recommendations.

6.3 Hybrid Methods for Service Recommendation

By combining both collaborative filtering and content-based methods, hybrid approaches for service recommendation can incorporate the advantages of the both methods while eliminating the weaknesses found in each approach [49], [41]. Hybrid approaches have improved prediction performance and overcome the cold start and sparsity problems of CF methods. Although hybrid recommendation approaches have been actively proposed in other areas such as e-Commerce [41], there is very limited work in the literature exploiting hybrid methods for Web service recommendation. The work presented in [37] is the only effort we are aware in this direction. Unfortunately, the paper largely focuses on proposing an approach to recommend services based on semantic content similarities of Web services. It remains unclear on how the collaborative filtering and content-based methods are integrated.

Our work presents a hybrid approach for better Web service recommendation by systematically combining both methods together. In particular, we propose a three-way aspect model that considers both QoS ratings and the semantic content of Web services. User preferences are modeled as a set of latent variables in the aspect model [26], which can be statistically estimated using the expectation maximization (EM) method. To avoid overfitting problems caused by data sparsity, we further propose two strategies.

8. http://www.w3.org/Submission/OWL-S.
9. http://www.w3.org/Submission/wadl.

The first one is to pre-process data matrix using a data smoothing technique and the second one is a modified aspect model that captures relationship between users and semantic content descriptors of Web services. To the best of our knowledge, our work is one of the first that combines collaborative filtering and content-based approach for Web service recommendation.

7 Conclusion and Future Work

Web service recommendation and selection is a fundamental issue in service oriented computing. Existing Web service discovery and recommendation approaches focus on either the perishing UDDI registries, or keyword-dominant, QoS-based Web service search engines. Such approaches possess many limitations such as poor recommendation performance and heavily relying on the input from users (e.g., preparing correct queries). In this paper, we have proposed a novel hybrid approach for effective Web service recommendation. Our approach exploits a three-way aspect model that systematically combines classic collaborative filtering and content-based recommendation. The proposed hybrid approach simultaneously considers the similarities of user ratings and semantic content of Web services. Our approach is validated by conducting extensive experimental studies using 3,693 real-world Web services publicly available from the Internet. The experimental results show that our approach outperforms the conventional collaborative and content-based methods in terms of recommendation performance.

Our future work includes exploring more refined/personalized Web service recommendation by considering the specific contexts (e.g., goals that an end user would like to achieve, physical situations, etc.). We also plan to apply our approach to other areas such as service clustering.

Acknowledgments

Quan Z. Sheng's work has been partially supported by Australian Research Council (ARC) Discovery Grant DP0878917 and DP140100104. The authors would like to thank the anonymous reviewers for their valuable feedback on this work.

References

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *IEEE Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[2] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and Managing Web Services: Issues, Solutions, and Directions," *The VLDB Journal*, vol. 17, no. 3, pp. 537–572, 2008.

[3] A. Bouguettaya, Q. Z. Sheng, and F. Daniel, Eds., *Web Services Foundations*. Springer, 2014.

[4] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web Services Composition: A Decade's Overview," *Information Sciences*, vol. 280, pp. 218–238, 2014.

[5] J. Ma, Q. Z. Sheng, K. Liao, and A. H. Ngu, "WS-Finder: A Framework for Similarity Search of Web Services," in *Proceedings of the 10th International Conference on Service Oriented Computing (ICSOC 2012)*, Shanghai, China, November 2012.

[6] T. H. Noor, Q. Z. Sheng, A. H. Ngu, and S. Dustdar, "Analysis of Web-Scale Cloud Services," *IEEE Internet Computing*, vol. 18, no. 4, pp. 55–61, 2014.

[7] Z. Zheng, X. Wu, Y. Zhang, M. R. Lyu, and J. Wang, "QoS Ranking Prediction for Cloud Services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1213–1222, 2013.

[8] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-Aware Web Service Recommendation by Collaborative Filtering," *IEEE Transactions on Services Computing*, vol. 4, no. 2, pp. 140–152, 2011.

[9] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending Web Services via Combining Collaborative Filtering with Content-Based Features," in *Proc. of the 20th IEEE International Conference on Web Services (ICWS 2013)*, Santa Clara Marriott, CA, USA, 2013.

[10] M. Huhns and M. Singh, "Service-oriented computing: Key concepts and principles," *IEEE Internet Computing*, vol. 9, no. 1, pp. 75–81, 2005.

[11] E. Al-Masri and Q. Mahmoud, "Investigating Web Services on the World Wide Web," in *Proceedings of the 17th International World Wide Web Conference (WWW'08)*, Beijing, China, April 2008.

[12] D. Kourtesis and I. Paraskakis, "Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery," in *Proceedings of the 5th European Semantic Web Conference (ESWC'08)*, Tenerife, Canary Islands, Spain, June 2008.

[13] P. Krill, "Microsoft, IBM, SAP Discontinue UDDI Registry Effort," http://www.infoworld.com/d/architecture/microsoft-ibm-sap-discontinue-uddi-registry-effort-777, visited on 07 May 2013.

[14] P. Bonatti and P. Festa, "On Optimal Service Selection," in *Proceedings of 14th International World Wide Web Conference (WWW 2005)*, Chiba, Japan, 2005.

[15] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity Search for Web Services," in *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004)*, Toronto, Canada, 2004.

[16] L. Yao and Q. Z. Sheng, "Particle Filtering based Availabiity Prediction for Web Services," in *Proceedings of the 9th International Conference on Service Oriented Computing (ICSOC 2011)*, Paphos, Cyprus, December 2011.

[17] L. Chen, L. Hu, Z. Zheng, J. Wu, J. Yin, Y. Li, and S. Deng, "WT-Cluster: Utilizing Tags for Web Services Clustering," in *Proceedings of 9th International Conference on Service-Oriented Computing (ICSOC 2011)*, Paphos, Cyprus, 2011.

[18] Y. Zhang, Z. Zheng, and M. Lyu, "WSExpress: A QoS-Aware Search Engine for Web Services," in *Proceedings of 8th IEEE International Conference on Web Services (ICWS 2010)*, Miami, Florida, USA, 2010.

[19] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *Proceedings of IEEE International Conference on Web Services (ICWS 2010)*, Miami, Florida, USA, July 2010.

[20] J. Cao, Z. Wu, Y. Wang, and Y. Zhuang, "Hybrid Collaborative Filtering Algorithm for Bidirectional Web Service Recommendation," *Knowledge and Information System*, vol. 36, no. 3, pp. 607–627, 2013.

[21] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'99)*, Stockholm, Sweden, 1999.

[22] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[23] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering WSDL Documents to Bootstrap the Discovery of Web Services," in *Proceedings of the International Conference on Web Services (ICWS 2009)*, Los Angeles, CA, USA, July 2009.

[24] M. B. Blake and M. F. Nowlan, "A Web Service Recommender System Using Enhancing Syntactical Matching," in *Proceedings of IEEE International Conference on Web Services (ICWS'07)*, Salt Lake City, Utah, USA, 2007.

[25] W. Liu and W. Wong, "Web Service Clustering Using Text Mining Techniques," *International Journal of Agent-Oriented Software Engineering*, vol. 3, no. 1, pp. 6–26, 2009.

[26] A. Popescul, L. Ungar, D. Pennock, and S. Lawrence, "Probabilistic Models for Unified Collaborative and Content-based Recommendation in Sparse-Data Environments," in *Proceedings of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI'01)*, Seattle, Washington, USA, 2001.

[27] N. Ueda and R. Nakano, "Deterministic Annealing EM Algorithm," *Neural Networks*, vol. 11, no. 2, pp. 271–282, 1998.
[28] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Computing Surveys*, vol. 22, no. 1, pp. 5–53, 2004.
[29] J. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.
[30] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International World Wide Web Conference (WWW'01)*, Hong Kong, China, May 2001.
[31] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS'07)*, Vancouver, Canada, December 2007.
[32] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*, Bonn, Germany, August 2005.
[33] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring Latent Features for Memory-based QoS Prediction in Cloud Computing," in *Proceedings of the 30th IEEE Symposium on Reliable Distributed Systems (SRDS 2011)*, Madrid, Spain, October 2011.
[34] F. Lécué and A. Delteil, "Making the Difference in Semantic Web Service Composition," in *Proceedings of the 22nd International Conference on Artificial Intelligence (AAAI'07)*, Vancouver, Canada, July 2007.
[35] A. Segev and Q. Z. Sheng, "Bootstrapping Ontologies for Web Services," *IEEE Transactions on Services Computing*, vol. 5, no. 1, pp. 33–44, 2012.
[36] W. Cohen, A. Borgida, and H. Hirsh, "Computing Least Common Subsumers in Description Logics," in *Proceedings of International Conference on Artificial Intelligence (AAAI'92)*, San Jose, CA, USA, July 1992.
[37] F. Lécué, "Combining Collaborative Filtering and Semantic Content-based Approaches to Recommend Web Services," in *Proceedings of the 4th IEEE International Conference on Semantic Computing (ICSC 2010)*, Pittsburgh, PA, USA, 2010.
[38] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.
[39] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality Driven Web Services Composition," in *Proceedings of the 12th International World Wide Web Conference (WWW 2003)*, Budapest, Hungary, May 2003.
[40] B. Benatallah, Q. Z. Sheng, and M. Dumas, "The Self-Serv Environment for Web Services Composition," *IEEE Internet Computing*, vol. 7, no. 1, pp. 40–48, January/February 2003.
[41] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, 2009.
[42] T. Qiu, L. Li, and P. Lin, "Web Service Discovery with UDDI Based on Semantic Similarity of Service Properties," in *Proceedings of the Third International Conference on Semantics, Knowledge and Grid (SKG'07)*, 2007.
[43] J. Yu and G. Zhou, "Web Service Discovery and Dynamic Invocation based on UDDI/OWL-S," in *Proceedings of the Third International Conference on Business Process Management (BPM'05)*, Nancy, France, 2005.
[44] N. Thio and S. Karunasekera, "Automatic measurement of a qos metric for web service recommendation," in *Proceedings of 2005 Australian Software Engineering Conference*, 2005, pp. 202–211.
[45] H. Zheng, J. Yang, W. Zhao, and A. Bouguettaya, "QoS Analysis for Web Service Compositions Based on Probabilistic QoS," in *Proceedings of the 9th International Conference on Service Oriented Computing (ICSOC 2011)*, Paphos, Cyprus, December 2011.
[46] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS Prediction for Web Services via Collaborative Filtering," in *Proceedings of IEEE International Conference on Web Services (ICWS'07)*, 2007.
[47] Q. Yu, Z. Zheng, and H. Wang, "Trace Norm Regularized Matrix Factorization for Service Recommendation," in *Proceedings of IEEE International Conference on Web Services (ICWS 2013)*, 2013.
[48] J. Lathem, K. Gomadam, and A. P. Sheth, "SA-REST and (S)mashups: Adding Semantics to RESTful Services," in *Proceedings of International Conference on Semantic Computing (ICSC 2007)*, 2007.
[49] M. Balabanović and Y. Shoham, "Content-Based, Collaborative Recommenation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, March 1997.

Lina Yao is currently a PostDoc researcher at School of Computer Science, the University of Adelaide. She received her PhD and M.Sc, both in Computer Science, from the University of Adelaide and B.E from Shandong University. Her research interests include Web mining, Internet of Things, ubiquitous computing and Service-Oriented Computing. She is the author of more than 20 publications.

Quan Z. Sheng is an associate professor and Head of Advanced Web Science Research Group at School of Computer Science, the University of Adelaide. He received the PhD degree in computer science from the University of New South Wales in 2006. His research interests include service-oriented architectures, distributed computing, Internet computing, and Web of Things. He is the recipient of ARC Future Fellowship in 2014, Chris Wallace Award in 2012, and Microsoft Research Fellowship in 2003. He is the author of more than 180 publications. He is a member of ACM and IEEE.

Anne H.H. Ngu is currently a full Professor with the Department of Computer Science at Texas State University-San Marcos. From 1992-2000, she worked as a Senior Lecturer in the School of Computer Science and Engineering, University of New South Wales (UNSW), Australia. She has held research scientist positions with Telecordia Technologies and Microelectonics and Computer Technology (MCC). She was a summer faculty scholar at Lawrence Livermore National Laboratory from 2003-2006. Her main research interests are in large-scale discovery and integration of information services, scientific and business process automation, agent systems and Internet of Things.

Jian Yu is currently a lecturer in the School of Computer and Mathematical Sciences at Auckland University of Technology, New Zealand. He is an Adjunct Professor at North China University of Technology, China. He received the PhD degree in Computer Software and Theory from Peking University, Beijing, China. His research interests include service-oriented computing, pervasive computing, business process management, and software engineering. He has published two books and more than 50 technical papers.

Aviv Segev is an Associate Professor at the Knowledge Service Engineering Department at KAIST - Korea Advanced Institute of Science and Technology. His research interests include classifying knowledge using the Web, context recognition and ontologies, knowledge mapping, and implementations of these areas in the fields of Web services, medicine, and crisis management. He is the author of over 50 publications. In 2004, he received his Ph.D. from Tel-Aviv University in Technology and Information Systems. He is a member of the IEEE and ACM.