Incorporating User, Topic, and Service Related Latent Factors into Web Service Recommendation

Xumin Liu and Isankumar Fulia
Department of Computer Science
Rochester Institute of Technology, Rochester, NY, 14623
Email: {xmlics, if2571}@rit.edu

***Abstract*—Due to the large and increasing number of web services, it is very helpful to provide a proactive feed on what is available to users, i.e., recommending web services. As collaborative filtering (CF) is an effective recommendation method by capturing latent factors, it has been used for service recommendation as well. However, the majority of current CF-based service recommendation approaches predict users' interests through the historical usage data, but not the service description. This makes them suitable for making QoS-based recommendation, but not for functionality-based recommendation. In this paper, we propose to use machine learning approaches to recommend web services to users from both historical usage data and service descriptions. Considering the great popularity of RESTful services, our approach is applicable to both structured and unstructured service description, i.e., free text descriptions. We exploit the idea of collaborative topic regression, which combines both probabilistic matrix factorization and probabilistic topic modeling, to form user-related, service-related, and topic related latent factor models and use them to predict user interests. We extracted public web service data and developer invocation history from ProgrammableWeb and conducted a comprehensive experiment study. The result indicates that this approach is effective and outperforms other representative recommendation methods.**

I. Introduction

Web services provide an efficient, convenient, and flexible way of delivering businesses, interacting with customers, and exchanging or sharing data on the Web. They also allow complicated and instant services accessible to ubiquitous mobile devices, such as smartphones and tablets. As a result, the number of online services and the size of their users have been dramatically increased over the past decade. This makes **service recommendation**, which is to proactively suggest interesting web services to users, a promising solution to effectively advertising web services, facilitating service discovery, and improving user experiences. This is analogous to making push notifications or recommending related products to customer by e-commerce companies, such as Amazon, which have been demonstrated to be very effective in promoting products and improving customers' purchasing experiences. Moreover, web service recommendation can help address the unbalanced, long tail distribution of service usages, where only a small portion of web services are being known and intensively used and the large portion of other services are seldom noticed [14]. However, it is very challenging to make a good recommendation. The large number of web services makes it difficult to locate web services that are relevant to a user's interest. For example, there will be more than 800 services returned by ProgrammableWeb, the largest online repository of public web services, after a search is conducted using map as the keyword. It would be a very tedious task to go through the complete list and find the interesting services. Meanwhile, many returned services may not be relevant as the search is keyword-based. Therefore, there is a need for an efficient and effective method.

There are several approaches proposed to recommend web services based on their functionality [2], [9], [12], [3], [10], non-functional features (i.e., QoS) [16], [15], [5], [11], [13], and the social networks the services involved into [6], [9]. More specifically, [2], [9], [12], [3] used clustering algorithms to generate *service clusters*, where each cluster contains a group of services providing similar functionalities. The similarity is measured by analyzing a service's functional description, such as WSDL files. service clusters can be used to efficiently locate services that provide relevant functionality. [16], [15], [5], [11], [13], [10] are CF-based. They made service recommendation from analyzing historical usage data. Some of them use *neighborhood methods*, where the recommendation is made based on the experiences or preferences of Top-k similar users [16], [15], [5]. They overlook that there may be some unobserved variables, i.e., latent factors, which affect a user's experience of using web services. To address this limitation, some other works exploit *matrix factorization* [11], [13], [10]. They modeled users and services as a latent vectors in a low-dimensional space with dimension K. The user's experience of using a service is computed as their dot product. These CF-based methods only use historical usage data as the learning input for prediction and do not consider service functionality. This makes them more suitable for making QoS-based recommendation, but not functionality-based one. An exception is that [10] proposed a hybrid method, which combines collaborative filtering and content-based recommendation. It assumes that there are some user-related latent variables, that determine the services they use and the description of these services. Such latent variables are be learned through an Expectation Maximization (EM) algorithm and then be used to predict a user's interest of a service. The limitation of this approach lies in the simplicity of the graphical model, where only user-related latent variables are included. It overlooks other possible latent factors, such as service-related or topic-related.

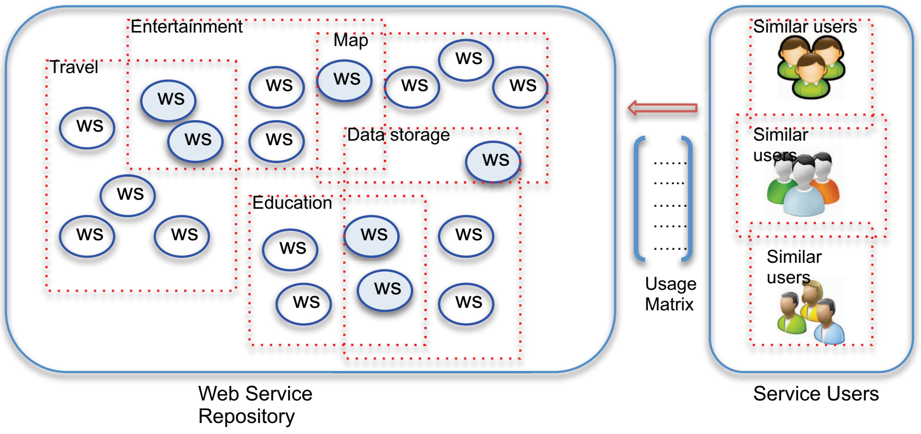


Fig. 1: Topic, Service, User Related Factors in Service Recommendation

We address the limitations of existing CF-based service recommendation approaches in this paper. We propose to make use of both service functional description and historical usage data when learning a recommendation model. Although WSDL is the standard web service description language, it is no longer practical to assume that most of the services have a WSDL description, due to the increasing popularity of RESTful services. Therefore, our approach is designed for unstructured, textual description of web services, such as the summary of service functionality and service tags. Obviously, it can be used for structured service description as well. Inspired by the collaborative topic regression idea presented in [8], we make service recommendation through probabilistic latent factor models, which include user-related, service-related, and topic-related latent variables, to improve the accuracy of recommendation. More specifically, the recommendation method combines probabilistic matrix factorization [7] and probabilistic topic modeling [1]. We use probabilistic matrix factorization to capture both user-related and service-related latent factors, and use probabilistic topic modeling to capture topic-related latent factors.

The remainder of this paper is organized as follows. In Section II, we give an overview of our recommendation method. In Section III, we present the latent factor models for web service recommendation and an optimization process for learning the model parameters. We also explain how the models are used to recommend interesting services. In Section IV, we present a comprehensive experimental study to demonstrate the effectiveness of the learning process. In Section V, we discuss some representative related works. In Section VI, we conclude our paper and discuss future work.

II. Overview of the Recommendation Method

In this Section, we briefly describe the method we use to recommend interesting web services to users.

As shown in Figure 1, a web service repository, such as a UDDI service registry or a service indexing website (e.g., ProgrammableWeb), includes various web services. Based on the functionality, each web service is associated to at least one specific theme or topic, such as travel, entertainment, map, education, and data storage. It is possible that a service may be associated to multiple topics as well. For example, a service that posts the movie times and driving directions to selected theaters provides both entertainment and map functionality.

Users select services from this repository for different purpose and out of different reasons. The factors that affect users to make selection can be classified into three categories: topic, service, and user related. The topics of a web service are related to the functionality of the service, which is always an essential factor that determines whether the service is desirable or not. The service related factors refer to those that are irrelevant to a service functionality but may be considered by users when choosing a service. These factors can be QoWS parameters, such as availability, reliability, and security level; or some other non-functional features, such as the outputting format (i.e., JSON or XML), and version compatibility. User related factors refer to those that are related to user preferences, which can be determined by users background, experience, consideration, and so on. This results in that, even with the services having similar features, different users may make difference selections. Users having the similar features tend to make the similar decision on selecting web services.

Our approach is to model the service recommendation as a prediction problem. That is, we can build a Service-by-User usage matrix, U, where each row represents a user and each column represents a service. Each entry of the matrix, $u_{i,j}$ represents whether the i-th user is interested in the j-th service, which can be either 1 or 0. Our approach is to predict the unknown entries in the matrix to determine what are the services that are interesting for a certain users. The prediction is made based on the information available to the public, including the service description and user usage history. User usage history records the list of services used by a user. This can be used to fill out entries in the usage matrix with the value of 1. The terms in a service description can be used to learn the latent topic models. Service related factors are not always observable. As they are also involved in the user selection process, it can be derived from the usage history.

Once the learning model which integrates the three types of latent factors is derived, it can be used to predict interesting services for a user.

III. Latent Factor Models for Web service Recommendation

Inspired by the idea presented in [8], we incorporate topic modeling methods into matrix factorization for service recommendation. That is, to follow the idea of matrix factorization based recommendation, we model both users and services as latent factors [7]. By using Latent Dirichlet Allocation (LDA), services are represented as a mixture of multiple topics based on their descriptions. Users' interests over multiple topics are learned and used in the recommendation process.

A. Topic Modeling for Web Services

Probabilistic topic modeling algorithms have been widely used as a text mining method to discover *latent* topics from a set of documents. Latent Dirichlet Allocation (LDA), a widely adopted topic modeling method, considers each topic as a probabilistic distribution over terms and each document as a mixture of multiple topics, i.e., multinomial distribution over topics. The posterior probabilistic distributions are derived through a statistical inference process, such as Gibbs sampling and variational inference. The learned distributions can be used for many text mining tasks, such as document clustering, classification, indexing, and querying.

By employing the idea of LDA, each service description s_i is represented by a mixture of multiple latent topics, such as advertisement, entertainment, and music. For example, Google Maps service's description could cover multiple topics, such as *mapping*, *traffic*, and *viewer*. This distribution is represented as θ, which can be sampled from a Dirichlet, i.e., $\theta \sim \text{Dir}(\alpha)$. α is the parameter of the Dirichlet distribution, i.e., $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_K\}$, assuming that there are K topics. Without any prior knowledge, we can assume that all the topics could have the same probabilistic proportion in a service description, i.e., $\alpha_1 = \alpha_2 = ... = \alpha_K$. α can also be tuned based on the prior knowledge. For example, we can set α_1 larger than α_2 if we know that a service's functionality is more related to the first topic, T_1, than the second topic, T_2. θ_j^k is the probability of choosing a term in s_j that belongs to the k-th topic.

Each topic is denoted by a multinomial distribution over the term vocabulary. We use β_k to denote the k-th topic and $\mathcal{O}$ to denote the term vocabulary. $\mathcal{O}$ can be generated by performing the union on all the terms that appear in service descriptions. Since β_k specifies the term mixture proportion of the k-th topic, β_k^i is the probability of the i-th term belonging to the k-th topic. β_k can also be sampled from a Dirichlet, i.e., $\beta_k \sim \text{Dir}(\eta)$. Similar to α, η is the parameter of the Dirichlet distribution, i.e., $\eta = \{\eta_1, \eta_2, ..., \eta_{|\mathcal{O}|}\}$. This modeling defines a generative learning process to generate a service description and the posterior distribution of the latent variables θ and β can be derived based on the observed appearance of terms in service descriptions, i.e., the model parameters are learned by maximizing the total likelihood of all the observed service descriptions. A detailed description of variational inference process can be found at [1].

We use free text description of web services, where there are usually many stop words, such as *this*, *is*, *a*, and *of*. As these stop words are irrelevant to a service's functional features, we remove them before learning topic models. Following this idea, when applying LDA to learn topic models from WSDL descriptions, WSDL tags should be first removed as they are irrelevant to a service's functionality.

B. A Generative Process for Terms and User Ratings

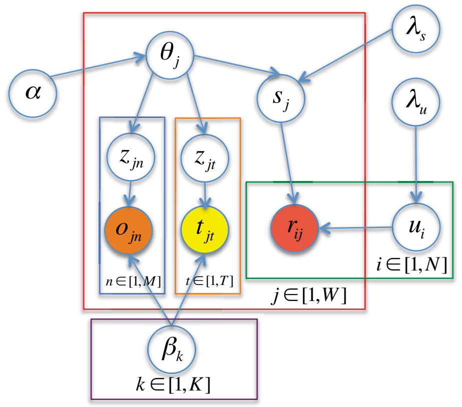
The graphical model is shown in Figure 2. There are two corpus-level hyper parameters, α and β, where α determines the topic proportion of a service description and β determines the term distribution in a topic. On the service-level, there are two service-level parameters, θ and s, where θ represents the topic proportion and s represents service-related latent vector. θ is determined by α. s is determined by θ and λ_s, which is a precision parameter for services used in probabilistic matrix factorization. On the term-level, there are three variables, z, t, and o, where o represents a term, t represents a tag, and z is a topic indicator, specifying the topic of a term. On the service-level, r is a variable that represents whether a user is interested in the service, which is determined by a user-level parameter, u. u is a user-related latent vector, which represents a user's interests over topics. It is also determined by λ_u, which is a precision parameter for users used in probabilistic matrix factorization. Here r and o are observable from historical usage data and service descriptions, respectively. Therefore, we can leverage the generative process of r and o to learn the posterior distributions of u and s. We then use the distributions to make predictions.

Suppose that there are K topics, M services, and N service users. Based on this graphical model, the generative process is described as follows:

A user is represented as a latent vector, $u_i \in \mathbb{R}^K$. For each user, we sample u_i using zero-mean spherical Gaussian distribution, i.e., $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$. Here λ_u^{-1} reflects the variance of the Gaussian. Therefore, λ_u is considered as precision parameter. The larger λ_u is, the smaller variation of u_i from its mean.

In probabilistic matrix factorization, a service is also represented as a latent vector in the shared latent space, $s_i \in \mathbb{R}^K$. It is also sampled using zero-mean spherical Gaussian distribution, i.e., $s_j \sim \mathcal{N}(0, \lambda_s^{-1} I_K)$. Differently, we leverage LDA and use the following steps to sample s_j:

1) Sample a topic mixture proportion for service j, $\theta_j \sim \text{Dir}(\alpha)$.
2) Sample a service latent offset vector ϵ_j using zero-mean spherical Gaussian distribution, $\epsilon_j \sim \mathcal{N}(0, \lambda_s^{-1} I_K)$. The reason that we include such a latent offset vector is that we assume that whether a user takes interest in a web service does not completely depend on the functionality of web services. There are also some other latent factors that affect user interest, which are not covered in a



α	Dirichlet parameter
θ	Topic proportion vector
β_k	Topic term distribution
λ_s	Precision of s
λ_u	Precision of u
z_j	Topic indicator
s	Service latent factor
u	User latent factor
o	Term in service
t	User interests in service

Fig. 2: Graphical Model for a Term and Rating Generative Process

service description. For example Google Maps service may cover the same set of topics as the ones covered by Microsoft Bing Maps service. However they attract different group of service users. We use the latent offset vector ϵ to capture the deviation between actual ratings and the ones determined by topic proportions. λ_s is the precision parameter. Therefore, the larger λ_s is, the smaller the variation of ϵ_j from its mean.

3) Following the idea above, set the service latent vector s_j as the combination of topic proportion and latent offset, i.e., $s_j = \theta_j + \epsilon_j$.
Suppose there are M_j terms in a service description, s_j. For each of these M_j terms:
    a) Sample a topic $z_{jn} \sim$ Multinomial (θ_j). That is, probabilistically choose one of the topics that are obtained in step 1).
    b) Sample a term $o_{jn} \sim$ Multinomial $(\beta_{z_{jn}})$. That is, probabilistically choose one term from the topic $\beta_{z_{jn}}$.

Suppose there are T_j tags for the service, s_j. For each of these T_j tags:
    a) Sample a topic $z_{jt} \sim$ Multinomial (θ_j). That is, probabilistically choose one of the topics that are obtained in step 1).
    b) Sample a term $t_{jt} \sim$ Multinomial $(\beta_{z_{jt}})$. That is, probabilistically choose one term from the topic $\beta_{z_{jt}}$.

We can see from above that T_j and S_j share the same distribution on the same parameter, i.e., θ_j, we can then combine them for simplicity by merging T_j into S. Each user-service pair (i,j), sample the rating using Gaussian Distribution, i.e., $r_{i,j} \sim \mathcal{N}(u_i^T s_j, c_{i,j}^{-1})$. Here $c_{i,j}$ represents the confidence level of trusting the rating of user u_i on service s_j, i.e., $r_{i,j}$. $c_{i,j}$ ranges from 0 to 1. If $c_{i,j} = 0$, it means we don't trust the rating, $r_{i,j}$ at all so that the rating will be ignored. $c_{i,j} = 1$ means that the rating, $r_{i,j}$ is totally trustable and should be seriously considered during the learning process. In our work, we consider that $c_{i,j}$ should take a higher value when $r_{i,i} = 1$ (i.e., a user explicitly shows his or her interest in a service) than the one when $r_{i,j} = 0$. This is because if $r_{i,j} = 0$, it doesn't necessarily mean that the user dislikes the service. It is possible that the user does not know the service at all. Therefore, the rating value of 0 is not as reliable as the ratings whose values are 1.

C. Predict User interests in Web services

Following the idea of probabilistic matrix factorization, a user's interest in a web service can be predicted as the dot product between user-related and service-related vectors, i.e., $r_{i,j} = (u_i)^T s_j = (u_i)^T(\theta_j + \epsilon_j)$. Therefore, to make the prediction, the key step is to learn the posterior distributions of u_i, θ_j, and s_j, given the two precision parameters, λ_u, λ_v, and topic assignment, β. Following the generative process described above, we can learn the parameters of probabilistic distributions based on two observable variables: o and r, i.e., by maximizing the total log likelihood of all the observed ratings, tags, and web service descriptions, i.e.,

$$\begin{aligned}\mathcal{L} &= \log P(U,S,R,\theta)\\ &= \log P(R|U,S) + \log P(U) + \log P(S|\theta) + \log P(\theta)\end{aligned} \tag{1}$$

As $r_{i,j}$ is sampled from a Gaussian distribution with the mean as $u_i^T s_j$ and variance as $c_{i,j}^{-1}$, i.e., $r_{ij} \sim \mathcal{N}(u_i^T s_j, c_{i,j}^{-1})$, we have:

$$\begin{aligned}\log P(R|U,S) &= \log(\prod_{i,j})\frac{c_{i,j}^{1/2}}{(2\pi)^{1/2}} exp\{-(c_{i,j}/2)(x - u_i^T s_j)^2\}\\ &\approx -\sum_{i,j}(c_{i,j}/2)(r_{i,j} - u_i^T s_j)^2\end{aligned}$$

As u_j is sampled from a zero-mean Gaussian distribution with the variance as λ_u^{-1}, we have:

$$\begin{aligned}\log P(U) &= \log(\prod_i)\frac{\lambda_u^{1/2}}{(2\pi)^{1/2}} exp\{-(\lambda_u/2)(u_i)^2\}\\ &\approx -(\lambda_u/2)\sum_i u_i^T u_i\end{aligned}$$

As $s_j = \theta_j + \epsilon_j$, where $\epsilon \sim \mathcal{N}(0, \lambda_s^{-1} I_K)$, s_j follows a Gaussian distribution with the mean of θ_j and variance of

λ_s^{-1}. So we have:

$$\log P(S,T) = \log(\prod_j) \frac{\lambda_s^{1/2}}{(2\pi)^{1/2}} exp\{-(c_{i,j}/2)(s_j-\theta_j)^2\}$$
$$\approx -(\lambda_s/2)\sum_{i,j}(s_j-\theta_j)^T(s_j-\theta_j)$$

As θ represents the topic proportion of a web service description, it is sampled once for each web service. It determines the occurrence of terms in the service description. So we have:

$$\log P(\theta) = \log(\prod_j\prod_n(\sum_K(p(z_{jn}^{(k)}|\theta_j)p(s_{j,n}|z_{jn},\beta))))$$
$$= \sum_j\sum_n(\log\sum(\theta_{j,k}\beta_{k,w_{jn}}))$$

Here, s_{jn} is an indicator of whether the n-th term appears in the j-th service. That is, $p(s_{j,n}|z_{jn},\beta) = \beta_{k,wjn}$. $z_{jn}^{(k)}$ is an indicator of whether this term belongs to the k-th topic. That is, $p(z_{jn}^{(k)}|\theta_j) = \theta_{j,k}$.

Therefore, the objective function is:

$$(u_i,s_j) = \arg\max_{u_i,s_j}\mathcal{L}$$
$$= \arg\max_{U,S}(-\sum_{i,j}(c_{i,j}/2)(r_{i,j}-u_i^T s_j)^2$$
$$-(\lambda_s/2)\sum_{i,j}(s_j-\theta_j)^T(s_j-\theta_j)$$
$$+\sum_j\sum_n(\log\sum(\theta_{j,k}\beta_{k,w_{jn}})))$$

The latent parameters, including u_i, s_j, and θ_j can be learned through an iterative optimization process. That is, a parameter can be learned through gradient decent while fixing the values of another two parameters. Following the idea of probabilistic matrix factorization, we can optimize U and S for a current parameter estimation of θ. As shown in Algorithm 1, θ is initialized by running a basic LDA algorithm (Line 1). Once a local optimal U and S are learned (Lines 5-7), we can use them to learn the optimal θ by updating the current estimation of θ (Line 8). Using this updated θ, we rediscover the optimal U and S again. This process iterates until all three of them converge. Once u_i and s_j are learned, user interest will be predicted as their dot product.

Algorithm 1 Learning User and service Latent Vectors

```
Require: $\hat{\mathbf{WS}}, \mathbf{T}, R, \lambda_u, \lambda_s, \beta$
Ensure: U, S, R*
 1: Run LDA on $\hat{\mathbf{WS}}$ and T, using $\beta$, to generate an initial estimation of $\theta_j$ for all $j$,
 2: Initialize $U$ and $S$ using $\lambda_u$ and $\lambda_s$.
 3: Iterate until converge
 4:     Use the current $\theta_j$,
 5:     Iterate until converge
 6:         Fixing U, update S by maximizing $\mathcal{L}$ via gradient decent in $S$
 7:         Fixing S, update U by maximizing $\mathcal{L}$ via gradient decent in $U$
 8:     Use the current $S$, update $\theta$ by maximizing $\mathcal{L}$ via gradient decent in $\theta$
 9: For the missing entries, make the predication as: $r_{i,j} = (u_i^*)^T s_j^*$
10: End.
```

IV. Experimental Study

In this Section, we performed a set of experiments to assess the effectiveness of using the combination of probabilistic matrix factorization (PMF) and topic modeling (TM) to make service recommendation. This approach is referred to as PMF-TM. We run our experiments on a real-word dataset, which we collected from ProgrammableWeb website. All experiments were carried out on a Machine with 2.6 GHz Quad-Core processor and 8GB DDR3 memory under Ubuntu operating system. Most of the existing web service recommendation systems predict QoS and use Mean Absolute Error (MAE) or Root Mean Square Error (RMSE) to evaluate their performance. However, both MAE and RMSE are not suitable for our evaluation as PMF-TM predicts a user's interest not the actual rating or QoS values. Therefore, we do not compare the performance of our work with theirs. As PMF and K-Nearest Neighborhood-based Collaborative Filtering (CF) have been widely used for recommendation, we compared PMF-TM with PMF and CF in our experiments.

A. Experimental Data Set

TABLE I: EXPERIMENT DATA SETS

Statistic Info.	DS1	DS2
# of Users	515	3744
# of services	779	6721
# of "1" Entries	4441	43405
Sparsity	98.9%	99.8%

ProgrammableWeb is the largest online service repository that includes the services in various categories and in different formats, e.g., SOAP-based and RESTful. With ProgrammableWeb, service providers can post the services and a description of their functionality. Service users can use services to build mashups and post the mashups to ProgrammableWeb. The link between service users and services can be tracked by the mashups posted by service users and the services used in the mashups. For example, a developer posts two mashups, m_1 and m_2, where m_1 uses s_1 and m_2 uses s_2, s_3, and s_4. This means that the user has used all the four services. This also implies that the developer likes these

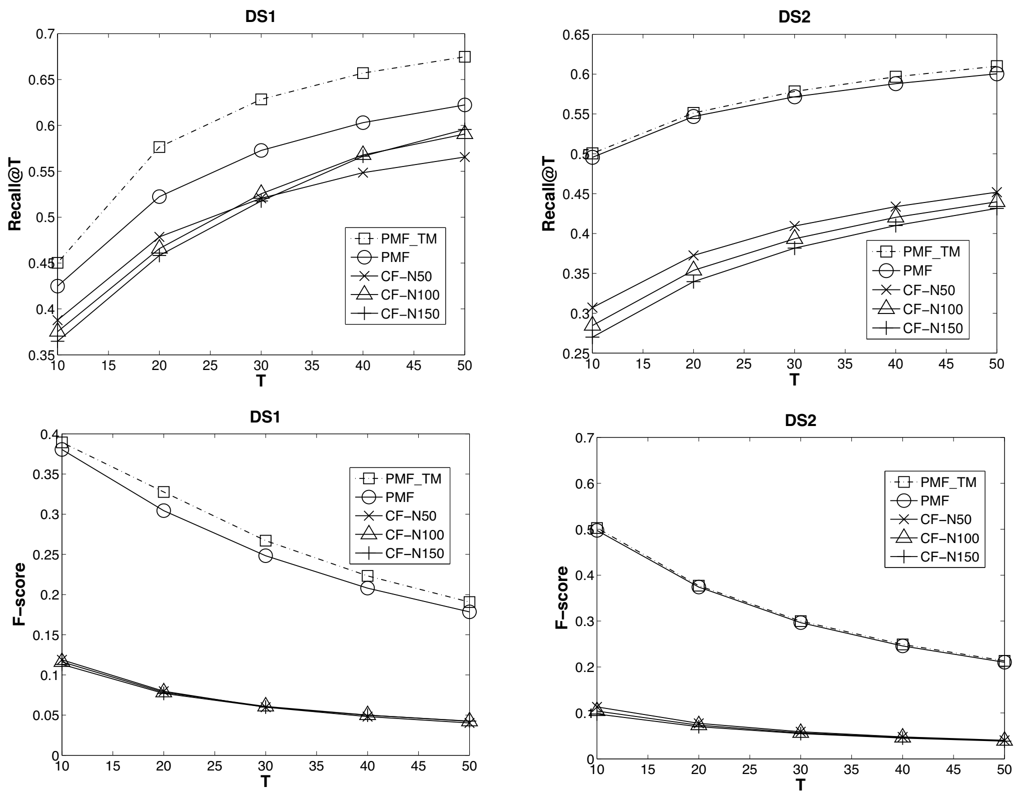


Fig. 3: Performance Comparison of PMF_TM, PMF, and K-Nearest Neighborhood Based CF

services. A service user can also follow a service no matter whether he or she has posted a mashup linking to the service.

As of July 2014, there were 11,199 registered services, 7,393 mashups that were developed using these services, and 78279 service users. We used the ProgrammableWeb developer services [1] to extract service summary, tags of services, and the link between services and service users. We also crawled the website to extract the followers of services since we believe that the follower relationship also reflects a developer's interests in a service.

From the data we extracted from ProgrammableWeb, we generated two data sets, as shown in Table 1. In DS1, we do not consider follower information. $r_{i,j} = 1$ only if u_i uses s_j. In DS2 we consider follower information. That is, $r_{i,j} = 1$ if u_i uses or follows s_j. For both data set, we only keep the service users that have more than four "1" entries (i.e., using or following more than four services) and remove those services that are not used by the selected service users. For each service, we use its brief introduction (usually containing few sentences), concatenated with its tags, as its description. We preprocess the description by removing stop words that are not relevant to the service's functionality.

B. Experiment Result

We performed experiments on the two data sets and compared PMF-TM with PMF and K-Nearest Neighborhood-based CF. For each data set, it is divided into a training set and testing set. We used the training set to learn the developer-related and service related latent vectors and train the prediction model. We then applied the prediction model to the testing set to compute the predication accuracy. Traditional metrics of evaluating recommendation systems includes *precision* and *recall*. Precision measuring the ratio of the number of properly recommended services to the total number of recommended services, is not suitable here. This is because a zero rating does not necessarily mean that a user dislikes the service. It could be the case that the user is not aware of the service at all. Therefore, in our experiments, we used **recall** as the evaluation metric. That is, given a threshold value T, a recommendation system suggests the top T services to a user. Recall is calculated as the ratio of the total number of properly recommended services in the suggested list (N_{Ti}) to the total number of services user likes (N_{u_i}). Therefore, the overall performance of the recommendation is the aggregation of all service users' recall. We also used **F-score**, a standard measure, for the evaluation.

We used five-fold cross validation to thoroughly evaluate PMF-TM. T's value ranges from 10 to 50. The experiment result using two data sets is shown in Figure 3. We compared the recommendation recall among PMF-TM, PMF, and K-Nearest Neighborhood Based CF. We choose three values for k when testing the CF, i.e., CF-N50, CF-N100, and CF-N150 in Figure 3. From the result, we can see that the recalls of these methods increase with the value of T in two data sets.

[1]http://api-portal.anypoint.mulesoft.com/Programmable-web/api/Programmable-web-api

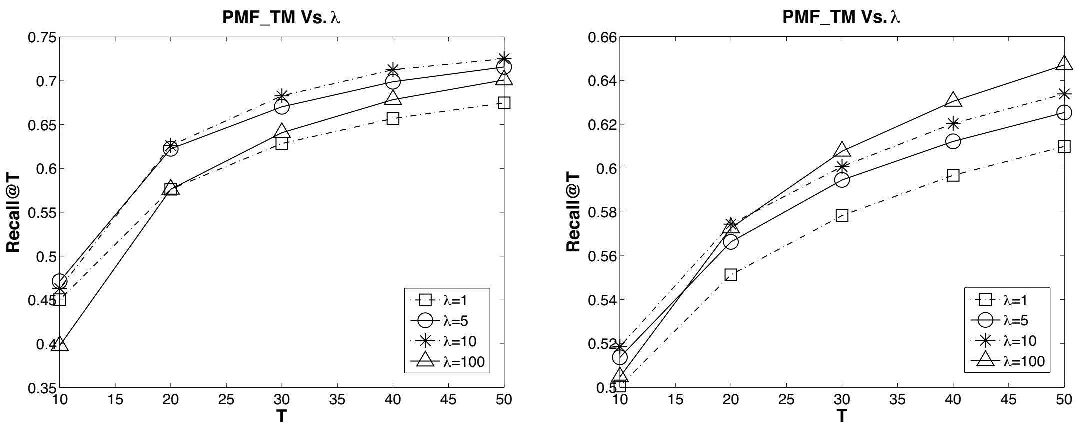


Fig. 4: Impact of λ

As the larger the T is, the more services a service user is recommended to. Therefore, the chances of recommending correct services is higher. In both datasets, both PMF-TM and PMF outperforms CF methods. This shows that there are some uncertain factors that affect a developer's interests in services. Using probabilistic models can better capture these factors and generate better recommendation results. PMF-TM outperforms the other two. The performance difference between PMF-TM and PMF is more obvious in DS1 (shown in the left chart) than in DS2 (shown in the right chart). This implies that the selection of web services to develop mashups is functionality-driven. Therefore, leveraging topic modeling can significantly contribute to recommendation as it incorporates functionality features into the recommendation process. However, it is not always the case that a service attracts its followers purely by its functionality. A service user may follow a service for different reasons, such as the service's advertisement or the advocation of its current users. Therefore, topic modeling is not as useful in DS2 for recommendation as it is in DS1. The F-scores of all the methods decrease with the number of T. As F score is affected by *precision*, which decreases with the growth of T. The F score tendency and comparison among these methods are similar to the recall.

We also examined the impact of λ_s on the predication performance, where the left and right charts give the result on DS1 and DS2, respectively. λ_s is a precision parameter of service-related latent vector. The larger it is, the smaller the service vector deviates from its topic proportion. From the experiment result shown in Figure 4, the recall increases with the value of λ_s. Therefore, topic modeling can effectively help predict user's interests of a service. This shows that, overall, a service attracts its users by its functionality. However, there are also exceptions. For example, when λ_s equals 100, the performance on DS1 (shown in the left chart) is not the best among all the numbers. This implies that the actual user interests do not completely follow service topic proportion. For example, two services are both related to mapping and viewer. But the first service focuses more on mapping and the second focuses more on viewer. Such difference cannot be captured by their topic proportions. As a result, recommendation recall will decrease if topic proportion takes a high weight for making prediction. Therefore, it is important to determine a proper λ_s. A too small value suggests a large service variance from topic proportion, which leads the prediction go too much towards PMF. A too large value suggests a small variance, which leads the prediction go too much towards TM.

Overall, our experiment result suggests that, by leveraging probabilistic matrix factorization and topic modeling, a good predication accuracy can be achieved even in a very sparse usage matrix (98.9% and 99.8% sparsity in our datasets). Therefore, it is effective for service recommendation.

V. Related Works

In this section, we describe several representative related works and differentiate them with our approach. Based on the methods they used, these works can be classified into three categories: QoS based, Functionality based, and hybrid.

[16], [15] proposed collaborative filtering based method to make QoS-aware service recommendation. It uses Pearson Correlation Coefficient (PCC) to compute the inter-user similarity and inter-service similarity. QoS value predication is performed by identifying top-k neighbors and aggregating their ratings. [5] proposed to select and recommend web services. In addition to integrating both user-based and item-based collaborative filtering algorithms, this work leveraged the influence of personalization. [13] proposed to predict user preference on services and tackle cold start problems in service recommendation. It integrates matrix factorization and decision tree learning to allow a fast and adaptive interview process with new users in order to solicit their information. [4] proposed to leverage time information into the recommendation process. The above approaches focus on recommending services based on non-functional features of web services. On the contrary, our approach focuses on recommending services based on all features of web services, including functional and nonfunctional, as well as user preferences.

[17] incorporated several factors including the user requirement, service network topology, service descriptions, mashup descriptions, and service evolution, to recommend web services to build mashups. This is different from our focus. [10] proposed to combine collaborative filtering and content-based recommendation. In the collaborative filtering part, it predicts a user's rating on a web service based on the ratings received from other similar users. In the content-based recommendation part, it computes the semantic similarity between services based on their WSDL descriptions. It uses a relatively simple graphical model and only considers user-related latent variables. In our work, we integrate both PMF and TM, and use a comprehensive graphical model, which includes user-related, service-related, and topic-related latent variables.

There are several approaches that cluster web services based on the structured description of their functionality, i.e., WSDL files and then use the clustering result for service selection and recommendation [12], [2]. More specifically, services providing similar functionality, which is derived from their WSDL descriptions, are clustered together. Once a user requests for a web service, the service cluster that is relevant to the request will be first located and the services in that clustered will be suggested. [12] uses a co-clustering algorithm that co-clusters services and operations of these services together in order to improve the accuracy of clustering result. [2] uses a LDA-based approach to integrate both user tags and service descriptions. These approaches only used the description of service functional features to determine relevant services for a user query. They did not consider latent factors that affect a user's interests in a service. Meanwhile, they relied on structured service description, WSDL files, which are not always provided, especially for RESTful services. Our work uses comprehensive latent factor models to recommend services based on their free text descriptions and historical usage data.

There are several approaches that leverage social media or social connections of users to make service recommendation to users. [6] studied the possibility of using various social networks to solve a service recommendation problem. [9] leveraged social relationships among service users, topics, mashups, and services to recommend suitable services to build mashups. It proposed a coupled matrix model that includes multi-dimensional relationships among service users, mashups, and services. This work is similar to our approach in terms of using graphical models and leveraging historical user usage information for making recommendation. However, it focuses on recommending services for mashups, not for service users.

VI. Conclusion and Future Work

In this paper, we exploited the idea of collaborative topic regression to make recommendation of web services through latent factor models. The recommendation is based on the free text description of a service's functionality. The approach combines probabilistic matrix factorization, which generates user-related and service-related latent factor models, and topic modeling, which generates topic-related latent factor models. Following the idea of PMF-TM, service users and services are modeled as latent vectors in a shared low-dimensional space. User ratings are computed as the dot product of the corresponding vectors. Following the idea of topic modeling, a service vector is augmented by the probabilistic topic proportion of its functionality description. The model parameters are learned through an iterative optimization process. The experiments performed on ProgrammableWeb data sets demonstrate the effectiveness of the proposed approach.

Our future work will focus on a more comprehensive recommendation method. We will incorporate other useful information into the recommendation process, such as the categories of services, user and provider profiles, and social connections between services through mashups, and examine how these factors impact the recommendation accuracy.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[2] L. Chen, Y. Wang, Z. Zheng Q. Yu, and J. Wu. WT-LDA: User Tagging Augmented LDA for Web Service Clustering. In *International Conference on Service Oriented Computing*, 2013.

[3] K. Elgazzar, A. E. Hassan, and P. Martin. Clustering wsdl documents to bootstrap the discovery of web services. In *IEEE International Conference on Web Services*, 2013.

[4] Y. Hu, Q. Peng, and X. Hu. A time-aware and data sparsity tolerant approach for web service recommendation. In *2014 IEEE International Conference on Web Services, ICWS, 2014, Anchorage, AK, USA, June 27 - July 2, 2014*, pages 33–40, 2014.

[5] Y. Jiang, J. Liu, M. Tang, and X. Liu. An effective web service recommendation method based on personalized collaborative filtering. In *IEEE International Conference on Web Services*, 2011.

[6] J. McDowall and L. Kerschberg. Leveraging social networks to improve service selection in workflow composition. In *International Conference on Advances in Social Networks Analysis and Mining*, 2012.

[7] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.

[8] C. Wang and D. Blei. Collaborative topic modeling for recommending scientific articles. In *ACM SIGKDD Conference on Knoweldge Discovery and Data Mining*, 2011.

[9] W. Xu, J. Cao, L. Hu, J. Wang, and M. Li. A social-aware service recommendation approach for mashup creation. In *IEEE International Conference on Web Services*, 2013.

[10] L. Yao, S. Q. Zheng, A. Segev, and J. Yu. Recommending web services via combining collaborative filtering with content-based features. In *IEEE International Conference on Web Services*, 2013.

[11] Q. Yu. Decision tree learning from incomplete qos to bootstrap service recommendation. In *IEEE International Conference on Web Services*, 2012.

[12] Q. Yu and M. Rege. On service community learning: A co-clustering approach. In *IEEE International Conference on Web Services*, 2010.

[13] Q. Yu, Z. Zheng, and H. Wang. Trace norm regularized matrix factorization for service recommendation. In *IEEE International Conference on Web Services*, 2013.

[14] Shuli Yu and C. Jason Woodard. Innovation in the programmable web: Characterizing the mashup ecosystem. In *ICSOC Workshops*, pages 136–147, 2008.

[15] Z. Zheng, H. Mao, M. Lyu, and I. King. Qos-aware web service recommendation by collaborative filtering. *IEEE Transactions on Services Computing* , 4(2):140–152, 2003.

[16] Z.B. Zheng, H. Ma, M.R. Lyu, and I. King. Wsrec: a collaborative filtering based web service recommendation system. In *IEEE International Conference on Web Services*, 2009.

[17] Y. Zhong, Y. Fan, K. Huang, W. Tan, and Jia Zhang. Time-aware service recommendation for mashup creation in an evolving service ecosystem. In *2014 IEEE International Conference on Web Services, ICWS, 2014, Anchorage, AK, USA, June 27 - July 2, 2014*, pages 25–32, 2014.