Open access • Proceedings Article • DOI:10.1109/ICWS.2015.38

Service Recommendation for Mashup Composition with Implicit Correlation Regularization — Source link

Lina Yao, Xianzhi Wang, Quan Z. Sheng, Wenjie Ruan ...+1 more authors

Institutions: University of Adelaide

Published on: 27 Jun 2015 - International Conference on Web Services

Topics: Mashup, Web service and Latent variable model

Service Recommendation for Mashup Composition with Implicit Correlation Regularization

Lina Yao, Xianzhi Wang, Quan Z. Sheng, Wenjie Ruan, and Wei Zhang
School of Computer Science, The University of Adelaide, Australia; Email: {lina, xianzhi, qsheng, wenjie}@cs.adelaide.edu.au

***Abstract*—In this paper, we explore service recommendation and selection in the reusable composition context. The goal is to aid developers finding the most appropriate services in their composition tasks. We specifically focus on mashups, a domain that increasingly targets people without sophisticated programming knowledge. We propose a probabilistic matrix factorization approach with implicit correlation regularization to solve this problem. In particular, we advocate that the co-invocation of services in mashups is driven by both explicit textual similarity and implicit correlation of services, and therefore develop a latent variable model to uncover the latent connections between services by analyzing their co-invocation patterns. We crawled a real dataset from ProgrammableWeb, and extensively evaluated the effectiveness of our proposed approach.**

***Keywords*-Recommendation; matrix factorization; mashup; latent variable model**

I. Introduction

Service-oriented computing promises the exposure and consumption of computing resources through the Internet in the form of platform-independent Web services. This leads to a large number of applications developed heavily based on Web services [1][2]. Among all these applications, mashup services are a type of lightweight Web applications that composes existing Web services to shorten development period and enhance scalability [3]. Nowadays, mashup services have become a popular form of applications and have gained support from multiple platforms, such as Google Mashup Editor[1], IBM Mashup Center[2], and Yahoo pipes[3]. Until Feb. 2015, there has already been over 12,839 Web APIs and over 6,168 Mashups on ProgrammableWeb[4], and the number is still increasing.

As an example, Figure 1 shows the mashup scenario of a mobile application, *WunderWalk*[5], which enables users to search for places of interest in urban settings. The application combines the Google Maps API[6] and the Foursquare API[7]. In this application, the Google Maps API provides basic functionalities regarding the mapping, such as searching and marking a specific location, while the Foursquare API enables the basic functionalities regarding social activities, such as check-in services and sharing comments and pictures with friends. By invoking the two types of APIs, *WunderWalk* is able to provide new and powerful functionalities to users, such as viewing and reviewing marked locations and friends' check-in records or sharing pictures associated with the locations.

[1]https://developers.google.com/mashup-editor/
[2]http://www-10.lotus.com/ldd/mashupswiki.nsf
[3]https://pipes.yahoo.com/pipes/
[4]http://www.programmableweb.com/
[5]http://www.wunderwalk.com/
[6]https://maps.googleapis.com/maps/api/js
[7]https://api.foursquare.com/

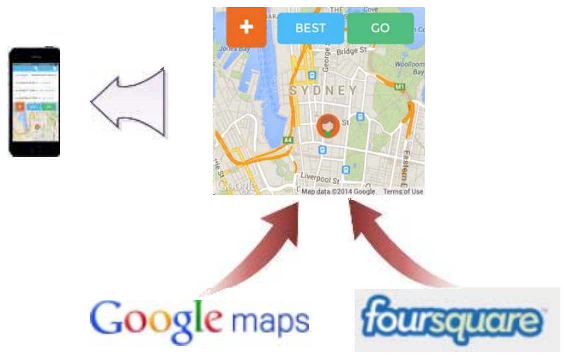


Fig. 1. An example mashup: the *WunderWalk* application integrates the Google Maps API and the Foursquare API to provide mobile social services

Given the large number and diversity (e.g., QoS) of the available services on the Web, it has become more difficult than ever to develop a mashup service due to the unprecedentedly large scope of choices on selecting the services. Thus, it becomes a significant challenge as how to effectively recommend mashup developers with high-quality services in order to accelerate the mashup development. A desirable recommendation result should not only fit the users' interest, but also be closely relevant to the other services in the mashup (some may have already been selected for the target mashup) to represent a high quality recommendation.

As a dominant approach to implementing the collaborative filtering methods, traditional matrix factorization based recommendation techniques use the historical interactions (e.g., API invocations in historical mashups) as input to make recommendations [4][5][6][7]. Traditional matrix factorization techniques focus on decomposing the interaction matrix (i.e., Mashup-API matrix in our paper) into two low rank matrix approximations, and then using these factorized matrices for recommendation. However, the matrix factorization based recommendation techniques rely on rich records of historical interactions to make accurate recommendations. For those newly released or rarely invoked APIs, the prediction accuracy of these methods is often quite limited.

Some recent works have demonstrated the effectiveness of integrating entity correlations into the matrix factorization process. By incorporating users' social relations [8] or location similarity dependency into service recommendation [9] as regularization terms, the recommendation quality can be greatly improved. The regularization terms introduced by social relations or location similarity can ensure that the distance of the latent feature vectors of two users or two locations is closer if they share some underlying similarity.

Inspired by above successful applications, we propose to incorporate the *API service correlations* as an extra regularization into the matrix factorization objective function. Instead of directly utilizing the explicit correlations (e.g., content-based similarity of API services), we design a latent variable model to infer implicit API service correlations, which reflect the latent similarities between APIs by studying API co-invocation patterns in historical mashup records. The motivation here is the generalized *homphily* in social science [10]: more similar two entities are, more interactions they may have (e.g., co-invoked by same mashups in this work). The co-invocations provide, for any pair of APIs, the number of mashups that comprise both APIs. The intuition behind this idea is that, similar API services are more possibly related to similar API services. To be more specific, API service i and j are more similar compared with API service i and k if the former pair are invoked by same mashups in a higher frequency than the latter pair. Our contributions are summarized as follows:

- We investigate the Web-based services (API) recommendation and selection in mashup under a regularized matrix factorization framework, where the API-mashup matrix is decomposed into two low-dimensional matrices, namely the *API latent subspace* and the *mashup latent subspace*.
- We specially explore the co-invocation patterns between APIs and propose a latent variable model to uncover the implicit similarity by capturing the underlying causal dependency of API invocation process. The experimental results demonstrate that the implicit similarity can boost recommendation performance.
- We crawl the Web to collect real mashup datasets and conduct extensive experiments to validate the proposed approach. The experimental results show the effectiveness of our approach. We also publicly release our mashup dataset for future study.

The rest of this paper is organized as follows. Section II introduces our proposed approach. Section III describes the derivation of implicit pairwise API correlations. Section IV reports the experimental settings and empirical study on proposed approach. Section V overviews related work and Section VI concludes the paper.

II. Method Overview

In this section, we first analyze data characteristics of the API invocation in mashup process. Then, we overview our proposed method.

A. Preliminary

To evaluate our proposed approach, we crawled 11,101 public Web-based API services and 5,658 mashups from ProgrammableWeb[8] (on April 2, 2014). Table I shows some statistics of the dataset.

TABLE I
Statistics of ProgrammableWeb dataset

Data Type	Statistic
Number of API	11,101
Number of Mashup	5,658
Size of Service Corpus	25,256
Mashup-API (MA) Composition Matrix Density	1.8811×10^{-4}
API-API (AA) Mutual Matrix Density	1.6397×10^{-4}

The API invocations have the following characteristics:

- **Sparsity.** Although numerous services are available on the Web, their occurrence in mashup applications is sparsity. An observed entry in the API-Mashup matrix (Figure 2 (a)) indicates the low average frequency of an API to participate in a mashup. Most APIs are never used or not even discovered yet, and the density of API-Mashup matrix is only approximately 1.8×10^{-4}. Meanwhile, a mashup service usually includes a very limited number of APIs (Figure 2 (b), 90% include less than 5 APIs);
- **Imbalance.** Among the API services that are involved in mashups, their usage frequency is imbalanced. A small portion is used very frequently, while the others are actually not used often. The API-Mashup Matrix (API invocation frequency) dataset is also imbalanced (Figure 2 (c)), with the "frequent" API used over 2,000 times and the "infrequent" APIs less than 10 times. The top 200 most frequent API invocations cover 99% of all API invocations by mashups (Figure 2 (d)).

In order to handle imbalance and sparsity, we propose a matrix factorization based recommendation approach, which have been proved successful [4][5][6] in addressing such two major challenges that also characterize the aforementioned API invocation in existing mashups.

B. The Proposed Method

We define the API recommendation problem as recommending a set of prospective APIs for a target mashup, given the invocation records of n APIs in k mashups. We denote invocation relation between APIs and mashups by a matrix $\mathbf{R} \in \mathbb{R}^{n \times k}$, where each element r_{ij} indicates whether or not an API $\mathbf{a}_i$ is invoked by a mashup m_j (true if $r_{ij} = 1$).

We explore the implicit functional correlations between services to make recommendations based on the matrix factorization approach. The basic idea is to take into account both API profiles and their co-invocation information in previous mashups to make recommendations. The involvement of co-invocation history can address the deficits of pure similarity based recommendation approaches, that sometimes some services simply supplement one another and would not be very similar in their profiles. In particular, we compute the explicit textural similarity of services to capture their domain

[8]http://www.programmableweb.com/

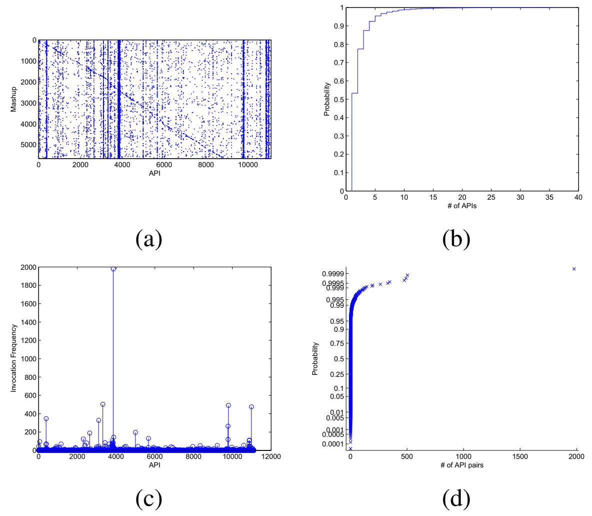


Fig. 2. (a) API-Mashup Matrix is highly sparse; (b) 90% mashup integrate less than 5 APIs; (c) The illustration of API invocation imbalance: infrequent APIS are only used less than 10 times, while frequent API over 2000 times; (d) The illustration of API invocation imbalance: top 200 most frequent APIs almost occupy 99% of all invocations between API and mashups

and investigate the co-invocation patterns of services to infer the implicit functional correlations between services. We then incorporate this correlation into the matrix factorization model as a regulation term to solve the recommendation problem.

The primary idea of matrix factorization is to map mashups (resp., APIs) into a shared lower dimensional space (the new dimensionality $d \ll \min\{n, k\}$). Given the factorization results of mashups $\mathbf{a}_i \in \mathbb{R}^d$ and of APIs $\mathbf{m}_j \in \mathbb{R}^d$, the probability that $\mathbf{a}_i$ would be invoked by $\mathbf{m}_j$ is estimated by:

$$\hat{r}_{ij} = \mathbf{a}_i^T \mathbf{m}_j \tag{1}$$

Thus, the latent factors of mashups and APIs can be denoted as matrices $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $\mathbf{M} \in \mathbb{R}^{k \times d}$, which can be learned by minimizing the ℓ_2 loss:

$$\min_{\mathbf{A},\mathbf{M}} \frac{1}{2} \sum_{i,j} I_{ij}(r_{ij} - \hat{r}_{ij})^2 + \frac{\lambda_{\mathbf{A}}}{2}||\mathbf{A}||_F^2 + \frac{\lambda_{\mathbf{M}}}{2}||\mathbf{M}||_F^2 \tag{2}$$

where I_{ij} equals 1 if API $\mathbf{a}_i$ is invoked by mashup $\mathbf{m}_j$, and 0 otherwise. $||\cdot||_F$ is the Frobenius norm of matrix, $\lambda_{\mathbf{A}}$ and $\lambda_{\mathbf{M}}$ are the regularization parameters. For the simplicity of parameter tuning, we simply set $\lambda_{\mathbf{A}} = \lambda_{\mathbf{M}}$. To incorporate the implicit functional correlations between APIs, we add a regularization term to Eq.2:

$$L = \min_{\mathbf{A},\mathbf{M}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{k} I_{ij}(r_{ij} - \mathbf{a}_i^T \mathbf{m}_j)^2 + \frac{\lambda_{\mathbf{A}}}{2}||\mathbf{A}||^2 + \frac{\lambda_{\mathbf{M}}}{2}||\mathbf{M}||^2 + \overbrace{\frac{\alpha}{2} \sum_{i=1}^{n} \sum_{b=1}^{n} \mathbf{Z}_{ib}||\mathbf{a}_i - \mathbf{a}_b||^2}^{1} \tag{3}$$

the last term (part 1) of Eq.3 integrates the link information of APIs, where $\mathbf{Z}$ indicates the pairwise latent relations of APIs (to be described in details in Section III). The intuition of adding the regularization term is to make as close latent representations as possible of the implicitly connected APIs. It can be easily solved by coordinating optimzation methods (i.e., alternately fixing one variable ($\mathbf{A}$ or $\mathbf{M}$) and optimizing the other by using gradient updating rules to progressively find a local minimum). The updating rules for the two variables are shown in Eq.4 and Eq.5, respectively.

$$\mathbf{a}_i \leftarrow \mathbf{a}_i + \eta_1(\delta_{ij}\mathbf{m}_j - \alpha \sum_{b \in \mathcal{N}_i} z_{ib}(\mathbf{a}_i - \mathbf{a}_b)) \tag{4}$$

$$\mathbf{m}_j \leftarrow \mathbf{m}_j + \eta_2(\delta_{ij}\mathbf{a}_i - \lambda_2 \mathbf{m}_j) \tag{5}$$

based on above definitions, we can quantify the possibility of API being invoked by each targeting mashup by calculating the predict score using Eq.1.

III. Implicit Correlation Z Derivation

In this section, we first formulate our proposed latent variable model for inferring the implicit correlations of API services from API co-invocation interactions, along with API service textual similarity. Then, we describe the model inference process.

A. *Model Specification*

We believe some implicit ingredients drive the co-use of APIs in mashup services. This latent relationship cannot be directly observed or obtained from API's descriptive profiles (e.g., category information and API description etc). We assume the latent correlation directly impacts the nature and frequency of API co-invocations in a mashup process. Intuitively, more co-invocations indicate stronger connections among pairwise APIs which may share stronger underlying similarities. In turn, the stronger the relationship, the higher likelihood that co-invocations will take place between the pair of APIs by the same mashup compositions. Thus, we model this latent correlations as a hidden factor of descriptive similarities indicated by API profiles.

Let $\mathbf{a}_i \in \mathcal{R}^m$ be the descriptive vectors of API service, y_{ij} be the counts of API services i and j being invoked by the same mashups, and z_{ij} be the pairwise implicit correlations of API services i and j. Specifically, we predict the implicit relations by adapting the statistical mixture model proposed in [11], [12] (as shown in Figure 3):

$$\begin{aligned} Pr(z_{ij}, y_{ij}|\mathbf{a}_i, \mathbf{a}_j) &= \underbrace{Pr(z_{ij}|\mathbf{a}_i, \mathbf{a}_j)}_{①}\underbrace{Pr(y_{ij}|z_{ij})}_{②} \\ &= Pr(z_{ij}|s_{ij})Pr(y_{ij}|z_{ij}) \end{aligned} \tag{6}$$

where s_{ij} indicates the explicit similarity of API services i and j calculated from their descriptive vectors. To obtain the latent relation z_{ij}, we need to solve Eq. 6 by specifying two types of dependencies: i) the dependency between the pairwise explicit similarity of API services s_{ij} and latent relations,

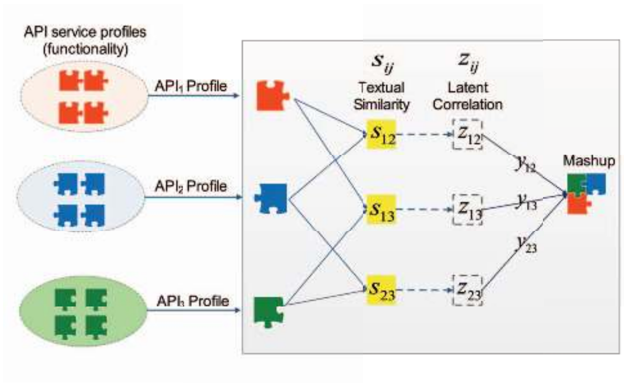
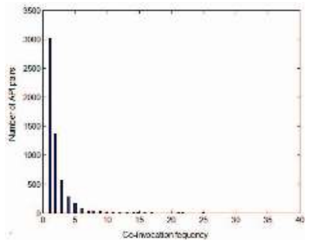
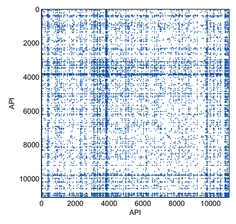


Fig. 3. The illustration of implicit correlation derivation model: s_{ij} denotes the explicit similarity between API services i and j, z_{ij} denotes their implicit relations between i and j, and the y_{ij} denotes the interaction between i and j, e.g., the frequency of service i and j engage in a same Mashup service.



(a)



(b)

Fig. 4. Illustrations of the co-invocation behaviors for API services

$Pr(z_{ij}|s_{ij})$, and ii) the dependency between the co-invocation patterns of each pair of API services y_{ij} and latent relations, $Pr(y_{ij}|z_{ij})$. We will describe the derivation process in details in the following subsections.

1) Specifying $Pr(z_{ij}|s_{ij})$*:* To measure the dependency between explicit similarity and implicit similarity between pairwise API services, we first use cosine similarity to quantify each pair of API services by evaluating their profile in the text level. We denote each API service file $\mathbf{a}_i \in \mathbb{R}^m$ using a TF/IDF weighting scheme. The process pipeline is briefly described as follows:

- Keywords corpus construction. This step aims to build a keyword repository of services, it is mainly divided into two stages. We first segment the service descriptions into terms and remove those terms with little meaning, such as 'a', 'the', and 'to', then eliminate the differences among the words with prefixes (e.g., -in, -un, -dis, -non, et al) or suffixes (e.g., -s, -es, -ed, -er, or,-ing, -ion, et al). As an example, the words "automate(s)", "automatic", "automation" should all map to "automate". In the end, we have a service corpus containing generic descriptive keywords with the size $k \approx 14,000$, denoted by $c_i \in \mathcal{C}$.
- Term frequency calculation. tf (term frequency) is the count of a term's occurrence in a given service description to measure the importance of the term. This count is usually normalized to prevent bias towards longer documents (which tends to have a higher term count regardless of the actual importance of that term in the document). The calculation is obtained by:

$$tf(c_{ij}) = \frac{freq(c_{ij}, x_i)}{|x_i|} \tag{7}$$

where $tf(c_{ij})$ means the frequency of the occurrence of j^{th} term c_j in the description of API service $x_i \in \mathcal{X}$.
- Inverse document frequency calculation. idf (inverse document frequency) is a measure of the general importance of a term in a set of API service descriptions, which can be calculated using:

$$idf(c_{ij}) = \begin{cases} \log \dfrac{|n|}{\sum_i \mathbb{I}(c_{ij} \in x_i)} & \text{if } \mathbb{I}(\cdot) \neq 0 \\ \log \dfrac{|n|}{1 + \sum_i \mathbb{I}(c_{ij} \in x_i)} & \text{if } \mathbb{I}(\cdot) = 0 \end{cases} \tag{8}$$

where $|n|$ is the number of services, $\mathbb{I}(\cdot)$ denotes the number of service descriptions where term c_j appears.

It should be noted that because most services have short description [13], we assign a higher weight to the idf value (i.e., $a_{ij} = tf \times idf^2$) to normalize the inherent bias of the tf measure in short documents.

Each API service can be represented as a descriptive vector $\mathbf{a}_i$. Thus, we can compute the explicit similarity s_{ij} between API services i and j using cosine similarity of their corresponding descriptive vectors as follows:

$$s_{ij} = \frac{\mathbf{a}_i \cdot \mathbf{a}_j}{||\mathbf{a}_i||||\mathbf{a}_j||} \tag{9}$$

up to this point, the first dependency in Eq. 6 can be specified:

$$\begin{aligned} Pr(z_{ij}|\mathbf{a}_i, \mathbf{a}_j) &= Pr(z_{ij}|s_{ij}, \mathbf{w}) \\ &= \mathcal{N}(w_{ij}s_{ij}, \sigma^2) \\ &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\dfrac{(z_{ij} - w_{ij}s_{ij})^2}{2\sigma^2}} \end{aligned} \tag{10}$$

We assume the relationship between z_{ij} and s_{ij} follows a zero-mean Gaussian distribution $z_{ij} = w_{ij}s_{ij} + \epsilon$, $\epsilon \sim N(0, \sigma^2)$. $w_{ij} \in \mathbf{w}$, which denotes a weight vector that is to be estimated and associated with the similarity of each pair of APIs. σ^2 is the variance in Gaussian model.

2) Specify $Pr(y_{ij}|z_{ij})$*:* To infer the co-invocation of APIs, we introduce an additional layer, *the latent relations of APIs*. This is based on the insight that higher frequency of co-invocation does not necessarily imply higher textual similarity of APIs in an explicit way. For example, two functionally complementary APIs may have low textual similarity but be frequently used together in mashups. Clearly, such similarity can not be captured well by the textual features.

In order to tackle this problem, we represent such *unobservable* similarity as latent variables, and specify them by studying the co-invocation behaviors of APIs in mashup activities. Figure 4 shows the co-invocation behaviors for pairwise API services. Figure 4 (a) shows the distribution of co-invocation counts for all pairs of API services, and

Figure 4 (b) shows the specific co-invocations of each pair of API services in mashups. As the natural stochastic model for the counting data is the Poisson distribution, we use *Poisson* distribution with a small λ to model how many times of co-invocations for each pair of API services given the influence of their latent correlations z_{ij}.

$$Pr(y_{ij}|z_{ij}) = \frac{\lambda^{y_{ij}} e^{-\lambda}}{y_{ij}!} = \frac{(\theta z_{ij})^{y_{ij}} \cdot e^{-\theta z_{ij}}}{y_{ij}!} \tag{11}$$

where $\lambda = \theta_i z_{ij}$, $\theta_i \in \theta$ is the weight vector, and y_{ij} is the count that $\mathbf{a}_i$ and $\mathbf{a}_j$ are engaged in same mashups.

B. Model Learning

Given three types of inputs, i.e., a training dataset $\mathcal{D}$ (denoted by n API pairs), the explicit similarity of services s_{ij} (Section III-A1), and the co-invocation of services y_{ij} (Section III-A2), the likelihood function of this model can be specified as below:

$$\begin{aligned} & Pr(\mathcal{D}|\mathbf{w},\theta)Pr(\mathbf{w})Pr(\theta) \\ & = \prod_{(i,j)\in\mathcal{D}} \Big(Pr(z_{ij}, y_{ij}|s_{ij}, \mathbf{w}, \theta)\Big) Pr(\mathbf{w})Pr(\theta) \\ & = \prod_{(i,j)\in\mathcal{D}} \Big(Pr(z_{ij}|s_{ij}, \mathbf{w})Pr(y_{ij}|z_{ij}, \theta)\Big) Pr(\mathbf{w})Pr(\theta) \\ & \propto \prod_{(i,j)\in\mathcal{D}} \Big(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z_{ij} - w_{ij}s_{ij})^2}{2\sigma^2}} \cdot (\theta z_{ij})^{y_{ij}} \cdot e^{-\theta z_{ij}}\Big) \\ & \cdot e^{-\frac{\lambda_w}{2}\mathbf{w}^T w} \cdot e^{-\frac{\lambda_\theta}{2}\theta^T\theta} \end{aligned} \tag{12}$$

To avoid overfitting, we put ℓ_2 regularization on the parameters $\mathbf{w}$ and θ as $Pr(w) \propto e^{-\frac{\lambda_w}{2}\mathbf{w}^T\mathbf{w}}$ and $Pr(\theta) \propto e^{-\frac{\lambda_\theta}{2}\theta^T\theta}$.

Then we take the logarithm of Equation 12 and adopt the stochastic gradient-based method to maximize the logarithm function by optimizing the model parameters $\mathbf{w}$ and θ and the latent variables z_{ij}.

$$\begin{aligned} \mathcal{L}(z_{ij\in\mathcal{D}}, \mathbf{w}, \theta) = & \sum_{(i,j)\in\mathcal{D}} \Big(-\frac{1}{2\sigma^2}(z_{ij} - w_{ij}s_{ij})^2 \\ & + y_{ij}\ln(\theta z_{ij}) - \theta z_{ij}\Big) - \frac{\lambda_w}{2}\mathbf{w}^T\mathbf{w} - \frac{\lambda_\theta}{2}\theta^T\theta \end{aligned} \tag{13}$$

The model parameters $\mathbf{w}$, θ and z_{ij} can be learned by following a generic coordinate ascent method, in which we update one parameter while fixing the other two parameters. The whole process runs iteratively until reaching a convergence threshold.

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial z_{ij}} & = \frac{1}{\sigma} \sum_{(ij)\in\mathcal{D}} \Big((w_{ij}s_{ij} - z_{ij}) + \frac{y_{ij}}{z_{ij}}\Big) \\ \frac{\partial\mathcal{L}}{\partial w_{ij}} & = \frac{1}{\sigma} \sum_{(i,j)\in\mathcal{D}} \Big((w_{ij}s_{ij} - z_{ij})s_{ij}\Big) - \lambda_w w_{ij} \\ \frac{\partial\mathcal{L}}{\partial\theta} & = \sum_{(i,j)\in\mathcal{D}} \Big(\frac{1}{\theta}y_{ij} - z_{ij}\Big) - \lambda_\theta\theta \end{aligned} \tag{14}$$

after obtaining the model parameters, for any given testing sample of an API service i, we first calculate its descriptive vector $\mathbf{a}_i$ and its textual similarity s_{ij} with other training APIs, then we can extract its co-invocation patterns with other training APIs y_{ij}.

$$z_{ij}^{new} = z_{ij}^{old} - \frac{\partial\mathcal{L}}{\partial z_{ij}} / \frac{\partial^2\mathcal{L}}{\partial(z_{ij})^2} \tag{15}$$

Up to this point, combined with learned model parameters, we can estimate the latent correlations between the testing API service i and the training API services z_{ij} using Eq. 15, where $\frac{\partial^2\mathcal{L}}{\partial(z_{ij})^2} = \frac{1}{\sigma}\sum_{(i,j)\in\mathcal{D}}\Big(1 + \frac{y_{ij}}{z_{ij}^2}\Big)$.

IV. Experiments

In this section, we first introduce the experimental settings and then present the analysis of the experimental results. The experiments focus on the following aspects: i) comparing the performance of our proposed method and the representative baseline methods, ii) studying the impact of proposed implicit correlations, and iii) evaluating the impact of different parameter settings in terms of dimensionality and regularization.

A. Experimental Settings

1) Validation: The training set and the testing set are constructed as follows: firstly, we randomly select 20% API-Mashup pairs as the testing set; then we randomly split the rest data into 8 parts, each containing 10% API-Mashup pairs. These parts are added incrementally to the training set to represent different sparsity levels (as shown in Figure 5). Table II shows statistics of the testing set and the training sets with different sparsity levels, where 10% means the training set contains 10% API-mashup pairs, and 20% means it contains 20% API-mashup pairs, and so on.

TABLE II
Training Data with Different Sparsity Levels

Data	Density
10%	1.8604×10^{-4}
20%	1.8671×10^{-4}
30%	1.8712×10^{-4}
40%	1.8751×10^{-4}
50%	1.8774×10^{-4}
TestingData	1.8774×10^{-4}

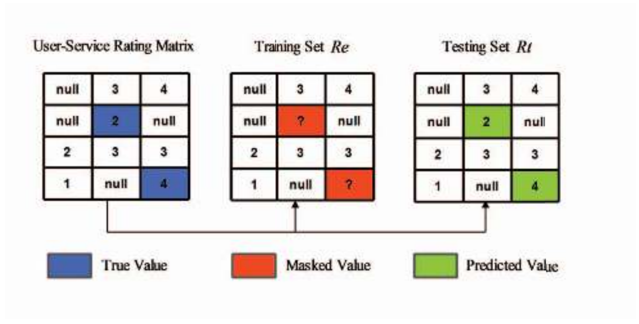


Fig. 5. The illustration of the experimental setting

2) Metrics: We adopt two commonly used metrics in collaborative filtering, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), to measure the recommendation performance. For the both metrics, smaller values indicate better performance.

$$MAE = \frac{1}{N}\sum_{ij}|r_{ij} - \hat{r}_{ij}| \tag{16}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{ij}(r_{ij} - \hat{r}_{ij})^2} \tag{17}$$

B. Comparison with Other Approaches

In this section, we investigate and compare our proposed approach with the following approaches over different sparsity levels (shown in Table II). The baseline methods are briefly depicted as follows:

- API-based Neighborhood (**AN**). This method uses Pearson Correlation to calculate similarities between APIs, and makes recommendation based on similar APIs [14].
- Mashup-based Neighborhood (MN). This method uses Pearson Correlation to compute the similarities between mashups and predicts invocations based on similar mashups.
- Non-negative Matrix Factorization (**NMF**). It applies non-negative matrix factorization on API-Mashup matrix for predicting the missing invocations. The invocation matrix between API and mashup $\mathbf{R}$ can be decomposed into two lower dimension matrices

$$\mathbf{R}_{ij} = \mathbf{A}_{ik}\mathbf{M}_{kj} \tag{18}$$

The model can be solved by the following optimization process

$$\min_{\mathbf{A},\mathbf{M}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}(r_{ij} - \mathbf{a}_i^T\mathbf{m}_j)^2 \tag{19}$$

- Regularized NMF (**RNMF**). It imposes two regularizations of $\mathbf{a}$ and $\mathbf{m}$ to avoid overfitting, and is formulated as:

$$\min_{\mathbf{A},\mathbf{M}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}(r_{ij} - \mathbf{a}_i^T\mathbf{m}_j)^2 + \frac{\lambda_{\mathbf{A}}}{2}||\mathbf{a}||_F^2 + \frac{\lambda_{\mathbf{M}}}{2}||\mathbf{m}||_F^2 \tag{20}$$

TABLE III
MAE AND RMSE PERFORMANCE COMPARISON

		10%	**20%**	**30%**	**40%**	**50%**
AN	**MAE**	0.2259	0.2135	0.2005	0.2002	0.1994
	RMSE	0.3646	0.3572	0.3469	0.3325	0.3253
MN	**MAE**	0.2284	0.2140	0.2067	0.2021	0.2003
	RMSE	0.3705	0.3525	0.3510	0.3497	0.3438
NMF	**MAE**	0.2413	0.2377	0.2215	0.2142	0.2123
	RMSE	0.3892	0.3811	0.3672	0.3511	0.3487
RNMF	**MAE**	0.1524	0.1447	0.1414	0.1392	0.1277
	RMSE	0.3002	0.2784	0.2646	0.2546	0.2452
PMF	**MAE**	0.1189	0.1142	0.1138	0.1119	0.1085
	RMSE	0.2867	0.2634	0.2554	0.2433	0.2325
Ours	**MAE**	**0.1164**	**0.1101**	**0.1062**	**0.1033**	**0.1014**
	RMSE	**0.2714**	**0.2552**	**0.2389**	**0.2205**	**0.2014**

- Probabilistic Matrix Factorization (**PMF**). It is one of the most famous MF models in collaborative filtering [6]. It assumes Gaussian distribution on the residual noise of observed data and places Gaussian priors on the latent matrices U and V. The objective function of PMF for the frequency data is defined as follows:

$$\min_{\mathbf{A},\mathbf{M}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}(g(r_{ij}) - g(\mathbf{a}_i^T\mathbf{m}_j)^2) + \frac{\lambda_{\mathbf{A}}}{2}||\mathbf{a}||_F^2 + \frac{\lambda_{\mathbf{M}}}{2}||\mathbf{m}||_F^2 \tag{21}$$

where $g(\cdot) = 1/(1+\exp(-x))$ is the logistic function.

For our approach, we set the dimensionality $d = 15$. For simplicity, we set the same value for the two regularization parameters $\lambda_{\mathbf{A}} = \lambda_{\mathbf{M}} = 0.01$ in the experiments. It should be noted that the impact of different parameter settings are studied in details in the rest of the section.

Table III shows that matrix factorization based methods generally achieve better performance than both AN and MN. Specially, our proposed method obtains better accuracy than other matrix factorization methods consistently. The reason lies in that our method integrates the inherent relations among APIs with the historical relations between APIs and mashups. The performance of our approach verifies the effectiveness of bringing the implicit relation information in the highly sparse API-Mashup matrix for improving the recommendation accuracy.

C. Impact of Implicit Correlations

To evaluate the impact of implicit relations of APIs, we compare them with two explicit similarity measures:

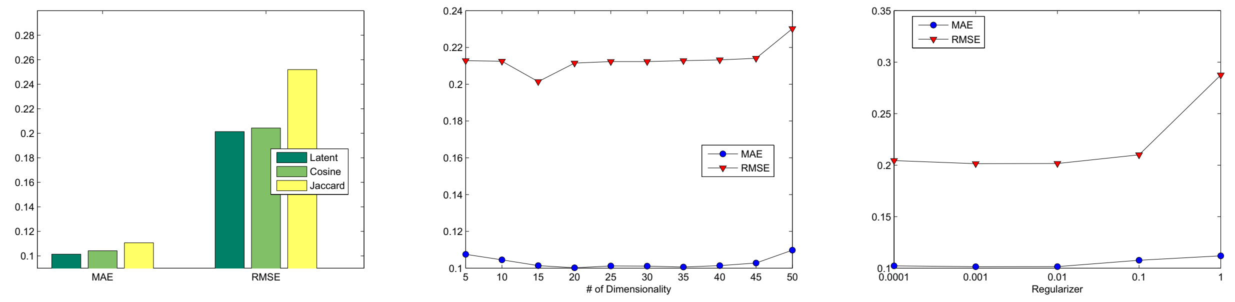
- Cosine similarity. We use cosine similarity to compute the relations between APIs $\mathbf{a}_i$ and $\mathbf{a}_j$, both denoted as TF/IDF vectors:

$$sim(\mathbf{a}_i, \mathbf{a}_j) = \frac{<\mathbf{a}_i, \mathbf{a}_j>}{||\mathbf{a}_i||||\mathbf{a}_j||} \tag{22}$$

- Jaccard similarity. The Jaccard similarity of each pair of APIs engaged in the mashup activities are calculated as:

$$sim(\mathbf{a}_i, \mathbf{a}_j) = \frac{|B_i \cap B_j|}{|B_i \cup B_j|} \tag{23}$$

where both B_i and B_j are binary vectors denoting the mashup sets that $\mathbf{a}_i$ and $\mathbf{a}_j$ participate in, respectively.



(a) (b) (c)

Fig. 6. Experimental results: (a) Impact of implicit correlations; (b) Impact of dimensionality; (c) Impact of regularizer

Figure 6 (a) shows that matrix factorization with implicit API correlations outperforms the other two similarity based methods. The reason lies in that implicit relations are derived from both the profiles and co-invocation records of APIs, which cannot be easily identified solely from content-based correlations. Jaccard similarity-based factorization achieves better performance than the pure content-based method, as it partially captures the intersections of different APIs in mashup activities.

D. *Impact of Dimensionality*

The parameter dimensionality determines the number of latent features used to characterize APIs and mashups. In this section, we study the impact of parameter dimensionality by varying the dimensionality value from 5 to 50 with a step value of 5. Figure 6 (b) shows that our approach achieves the best performance when the dimensionality value is 15, which may indicate the most appropriate latent factors for the API-Mashup histories. From the results, we can observe with the increase of latent factor number from 5 to 35, the MAE and RMSE keep decreasing. This observations are consistent with the intuitions that bigger number of latent factors can extract more informative structures. However, when the dimensionality exceeds 35, the performance begins to drop. The reason is that a larger dimensionality causes the overfitting problem. As a result, either too small (e.g., 5) or too large a dimensionality (e.g., 40, 50 etc) would degrade the recommendation performance.

E. *Impact of Regularization*

To decide the most appropriate regularization level, we study the sensitivity of regularization λ by tuning this parameter within the range of $\{10^{-4}, 10^{-3}, ..., 1\}$ with the step of 10^{-1}. Figure 6 (c) shows that both MAE and RMSE keep dropping as λ increases until $\lambda = 0.01$, and then begin to increase. That means the performance of our proposed approach achieves the best at $\lambda = 0.01$, which is also the reason that we take the value as the default setting in our comparative experiments.

V. Related Work

Service recommendation have been an active area of research for years. Traditional service recommendation approaches locate quality of mashup service to realize high-quality service recommendation [3][15]. Such methods require explicit specification of users' requirements to recommend the appropriate services. On the other hand, collaborative filtering (CF) models [16][17] can reflect to some extent users' implicit requirements. Thus, most recent service recommendation approaches are based on CF models. CF is a popular recommendation algorithm, which makes automatic prediction (filter) about the interests of a user by collecting preferences or taste information from many users (collaborating). These approaches compute similarity of users or services, predict missing QoS values based on the QoS records of similar users or services, and recommend the best services to users.

Among different CF methods, matrix factorization techniques [5][6] have gained popularity as a dominant class of methodology within collaborative filtering recommenders [4], due to the high accuracy and scalability. These methods focus on fitting the user-item rating matrix using low-rank approximations, and use it to make further predictions. Specially, Yu et al. [18] develop a trace norm regularized matrix factorization algorithm for recommending services with the best QoS to users. This work incorporates the low-rank structure and the clustered representation of real-world QoS data into a single unified objective function to estimate users' QoS experience.

In pursuit of higher accuracy, recent research commonly combines additional information into Matrix Factorization. Zheng et al. [7] propose a neighborhood-integrated Matrix Factorization approach for collaborative and personalized web service QoS value prediction. Chen et al. [19] take location in term of IP addresses of services into account to make more accurate recommendation. The approach combines *user interest value* based on the content similarity between user history records and Mashup services and *QoS predictive value* of Mashup services by collaborative filtering. Ma et al. [8] fuse MF with geographical and social influence for personalized point of interest (POI) recommendation in location-based social networks. Jamali et al. [20] incorporate trust propagation into the matrix factorization model for recommendation in social networks. More recently, Cao et al. [21] and Yao et al. [13] respectively propose a service recommendation approach based on both content similarity and collaborative filtering. Liu et al. [22] develop two extensions of the matrix factorization models, data weighting approach and time-aware modeling approach, for incorporating the social network structure in context aware recommendation. Xu et al. [23] propose a coupled matrix model to describe the multi-dimensional social relationships among users, mashups, and services, and design a factorization algorithm to predict unobserved relationships in the model to support more accurate service recommendations. Though tremendous efforts on modifying the Matrix Factorization model, few of them consider the impact of service invocation history to the probability of future invocations.

Inspired by above approaches and in sight of their shortcomings, we propose to extend a recommendation approach by integrating the API correlations regularization to the matrix factorization objective function. Lo et al. [24] also combine service similarity and Matrix Factorization in their missing value prediction. Besides serving different purposes, we infer implicit correlations among APIs rather than directly using the explicit API similarity for making recommendations. The implicit relations are more informative than simple explicit relations by taking the influence of historical invocation relations between mashups and APIs into account.

VI. Conclusion

This paper presents a Mashup service recommendation approach by integrating the implicit API correlations regularization into the matrix factorization model. The intuition is that both the content features of APIs and the historical invocation relations between APIs and mashups are essential in determining the future invocation of APIs in a target mashup. We define the model components and propose corresponding methods for inferring the model. The experimental results over a large real-world service dataset show that our approach outperforms the state-of-the-art collaborative filtering algorithms in term of accuracy. This work can be considered our preliminarily step of systematically exploring automatic service selection and recommendation in mashup composition by reusing online Web-based services.

As part of our future work, we will continue investigating more promising features of mashup applications to further improve the current approach. We will particularly focus on exploring structural information between service providers and service users by analyzing their *following* relations, e.g., some APIs have users as their followers. We will also perform verifications of the proposed methods in more practical mashup applications.

References

[1] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decades overview," *Information Sciences*, vol. 280, pp. 218–238, 2014.

[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] C. Cappiello, F. Daniel, M. Matera, and C. Pautasso, "Information quality in mashups," *Internet Computing, IEEE*, vol. 14, no. 4, pp. 14–22, 2010.

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

[5] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 713–719.

[6] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2007, pp. 1257–1264.

[7] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *Services Computing, IEEE Transactions on*, vol. 6, no. 3, pp. 289–299, 2013.

[8] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 287–296.

[9] X. Chen, Z. Zheng, Q. Yu, and M. Lyu, "Web service recommendation via exploiting location and qos information," 2013.

[10] M. Granovetter, "The strength of weak ties: A network theory revisited," *Sociological theory*, vol. 1, no. 1, pp. 201–233, 1983.

[11] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1999, pp. 50–57.

[12] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 981–990.

[13] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending web services via combining collaborative filtering with content-based features," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 42–49.

[14] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*. IEEE, 2009, pp. 437–444.

[15] X. Wang, Z. Wang, and X. Xu, "Semi-empirical service composition: A clustering based approach," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 219–226.

[16] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *Services Computing, IEEE Transactions on*, vol. 4, no. 2, pp. 140–152, 2011.

[17] L. Liu, N. Mehandjiev, and D.-L. Xu, "Multi-criteria service recommendation based on user criteria preferences," in *Proceedings of the fifth ACM conference on Recommender systems*, 2011, pp. 77–84.

[18] Q. Yu, Z. Zheng, and H. Wang, "Trace norm regularized matrix factorization for service recommendation," in *2013 IEEE 20th International Conference on Web Services (ICWS)*, 2013, pp. 34–41.

[19] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized qos-aware web service recommendation and visualization," *Services Computing, IEEE Transactions on*, vol. 6, no. 1, pp. 35–47, 2013.

[20] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 135–142.

[21] C. Buqing, M. Tang, and X. Huang, "Cscf: A mashup service recommendation approach based on content similarity and collaborative filtering." *International Journal of Grid & Distributed Computing*, vol. 7, no. 2, 2014.

[22] N. N. Liu, B. Cao, M. Zhao, and Q. Yang, "Adapting neighborhood and matrix factorization models for context aware recommendation," in *Proceedings of the Workshop on Context-Aware Movie Recommendation*, 2010, pp. 7–13.

[23] W. Xu, J. Cao, L. Hu, J. Wang, and M. Li, "A social-aware service recommendation approach for mashup creation," in *IEEE 20th International Conference on Web Services (ICWS)*, 2013, pp. 107–114.

[24] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "An extended matrix factorization approach for qos prediction in service selection," in *IEEE Ninth International Conference on Services Computing (SCC)*, 2012, pp. 162–169.