Web APIs Recommendation for Mashup Development Based on Hierarchical Dirichlet Process and Factorization Machines

Buqing Cao[1,2(✉)], Bing Li[2], Jianxun Liu[1], Mingdong Tang[1], and Yizhi Liu[1]

[1] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China
buqingcao@gmail.com, ljx529@gmail.com, tangmingdong@gmail.com, liuyizhi928@gmail.com

[2] State Key Laboratory of Software Engineering, International School of Software, Wuhan University, Wuhan, China
bingli@whu.edu.cn

Abstract. Mashup technology, which allows software developers to compose existing Web APIs to create new or value-added composite RESTful Web services, has emerged as a promising software development method in a service-oriented environment. More and more service providers have published tremendous Web APIs on the internet, which makes it becoming a significant challenge to discover the most suitable Web APIs to construct user-desired Mashup application from these tremendous Web APIs. In this paper, we combine hierarchical dirichlet process and factorization machines to recommend Web APIs for Mashup development. This method, firstly use the hierarchical dirichlet process to derive the latent topics from the description document of Mashups and Web APIs. Then, it apply factorization machines train the topics obtained by the HDP for predicting the probability of Web APIs invocated by Mashups and recommending the high-quality Web APIs for Mashup development. Finally, we conduct a comprehensive evaluation to measure performance of our method. Compared with other existing recommendation approaches, experimental results show that our approach achieves a significant improvement in terms of MAE and RMSE.

Keywords: Hierarchical dirichlet process · Factorization machines · Web APIs recommendation · Mashup development

1 Introduction

Currently, Mashup technology has emerged as a promising software development method in a service-oriented environment, which allows software developers to compose existing Web APIs to create new or value-added composite RESTful Web services [1]. More and more service providers have published tremendous Web APIs that enable software developers to easily integrate data and functions by the form of Mashup [2]. For example, until July 2016, there has already been more than 15,400

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 3–15, 2017.
DOI: 10.1007/978-3-319-59288-6_1

Web APIs on ProgrammableWeb, and the number of it is still increasing. Consequently, it becomes a significant challenge to discover most suitable Web APIs to construct user-desired Mashup application from tremendous Web APIs.

To attack the above challenge, some researchers exploit service recommendation to improve Web service discovery [3, 4]. Where, the topic model technique (e.g. Latent Dirichlet Allocation (LDA) [5]) has been exploited to derive latent topics of Mashup and Web APIs for improving the accuracy of recommendation [3, 4]. A limitation of LDA is that it needs to determine the optimal topics number in advance. For each different topic number in model training, there have a new LDA model training process, resulting in time-consuming problem. To solve this problem, Teh et al. [6] proposed a non-parametric Bayesian model—Hierarchical Dirichlet Process (HDP), which automatically obtain the optimal topics number and save the training time. Thus, it can be used to derive the topics of Mashups and Web APIs for achieving more accurate service recommendation.

In recent years, matrix factorization is used to decompose Web APIs invocations in historical Mashups for service recommendations [7, 8]. It decomposes the Mashup-Web API matrix into two lower dimension matrixes. However, matrix factorization based service recommendation relies on rich records of historical Mashup-Web API interactions [8]. Aiming to the problem, some recent research works incorporated additional information, such as users' social relations [9] or location similarity [10], into matrix factorization for more accurate recommendation. Even though matrix factorization relieves the sparsity between Mashup and Web APIs, it is not applicable for general prediction task but work only with special, single input data. When more additional information, such as the co-occurrence and popularity of Web APIs, is incorporated into matrix factorization model, its performance will decrease. FMs, a general predictor working with any real valued feature vector, was proposed by S. Rendle [11, 12], which can be applied for general prediction task and models all interactions between multiple input variables. So, FMs can be used to predict the probability of Web APIs invocated by Mashups.

In this paper, we propose a Web APIs recommendation approach based on HDP and FMs for Mashup development. The contributions of this paper are as follows:

- *We use the HDP to derive the latent topics from the description document of Mashups and Web APIs. Based on these topics, similar Mashups and similar Web APIs will be addressed to support the model training of FMs.*
- *We apply the FMs to train the topics obtained by the HDP for predicting the probability of Web APIs invocated by Mashups and recommending the high-quality Web APIs for Mashup development. In the FMs, multiple useful information is utilized to improve the prediction accuracy of Web APIs recommendation.*
- *We conduct a set of experiments based on a real-world dataset from ProgrammableWeb. Compared with other existing methods, the experimental results show that our method achieves a significant improvement in terms of MAE and RMSE.*

The rest of this paper is organized as follows: Sect. 2 describes the proposed method. Section 3 gives the experimental results. Section 4 presents related works. Finally, we draw conclusions and discuss our future work in Sect. 5.

2 Method Overview

2.1 The Topic Modeling of Mashup and Web APIs Using HDP

The Hierarchical Dirichlet Process (HDP) is a powerful non-parametric Bayesian method [13], and it is a multi-level form of the Dirichlet Process (DP) mixture model. Suppose (Θ, B) be a measurable space, with G_0 a probability measure on the space, and suppose a_0 be a positive real number. A *Dirichlet Process* [14] is defined as a distribution of a random probability measure G over (Θ, B) such that, for any finite measurable partition $(A_1, A_2, \ldots, A_r)$ of Θ, the random vector $(G(A_1), \ldots, G(A_r))$ is distributed as a finite-dimensional Dirichlet distribution with parameters $(a_0 G_0(A_1), \ldots, a_0 G_0(A_r))$:

$$(G(A_1), \ldots, G(A_r)) \sim Dir(a_0 G_0(A_1), \ldots, a_0 G_0(A_r)) \tag{1}$$

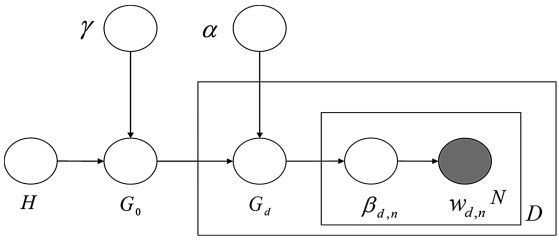

Fig. 1. The probabilistic graph of HDP

In this paper, we use the HDP to model the documents of Mashup and Web APIs. The probabilistic graph of the HDP is shown in Fig. 1, in which the documents of Mashup or Web APIs, their words and latent topics are presented clearly. Here, D represents the whole Mashup documents set which is needed to derive topics, and d represents each Mashup document in D. γ and a_0 are the concentration parameter. H is the base probability measure and G_0 is the global random probability measure. G_d represents a generated topic probability distribution of Mashup document d, $\beta_{d,n}$ represents a generated topic of the nth word in the d from G_d, and $w_{d,n}$ represents a generated word from $\beta_{d,n}$.

The generative process of our HDP model is as below:

(1) *For the D, generate the probability distribution* $G_0 \sim DP(\gamma, H)$ *by sampling, which is drawn from the Dirichlet Process* $DP(\gamma, H)$.
(2) *For each d in D, generate their topic distributions* $G_d \sim DP(a, G_0)$ *by sampling, which is drawn from the Dirichlet Process* $DP(a, G_0)$.
(3) *For each word* $n \in \{1, 2, \ldots, N\}$ *in d, the generative process of them is as below:*

- *Draw a topic of the nth word* $\beta_{d,n} \sim G_d$, *by sampling from* G_d;
- *Draw a word* $w_{d,n} \sim Multi(\beta_{d,n})$ *from the generated topic* $\beta_{d,n}$.

To achieve the sampling of HDP, it is necessary to design a construction method to infer the posterior distribution of parameters. Here, Chinese Restaurant Franchise (CRF) is a typical construction method, which has been widely applied in document topic mining. Suppose J restaurants share a common menu $\phi = (\phi)_{k=1}^{K}$, K is the amount foods. The jth restaurant contains m_j tables $\left(\psi_{jt}\right)_{t=1}^{m_j}$, each table sits N_j customers. Customers are free to choose tables, and each table only provides a kind of food. The first customer in the table is in charge of ordering foods, other customers share these foods. Here, restaurant, customer and food are respectively corresponding to the document, word and topic in our HDP model. Suppose δ is a probability measure, the topic distribution θ_{ji} of word x_{ji} can be regarded as a customer. The customer sits the table ψ_{jt} with a probability $\frac{n_{jt}}{i-1+a_0}$, and shares the food ϕ_k, or sits the new table $\psi_{jt_{new}}$ with a probability $\frac{a_0}{i-1+a_0}$. Where, n_{jt} represents the amount of customers which sit the tth table in the jth restaurant. If the customer selects a new table, he/she can assign the food ϕ_k for the new table with a probability $\frac{m_k}{\sum_k m_k+\gamma}$ according to popularity of selected foods, or new foods $\phi_{k_{new}}$ with a probability $\frac{\gamma}{\sum_k m_k+\gamma}$. Where, m_k represents the amount of tables which provides the food ϕ_k. We have the below conditional distributions:

$$\theta_{ji}|\theta_{ji},\theta_{ji},\ldots,\theta_{ji},a_0,G_0 \sim \sum\nolimits_{t=1}^{m_j}\frac{n_{jt}}{i-1+a_0}\delta_{\psi_{jt}}+\frac{a_0}{i-1+a_0}G_0 \tag{2}$$

$$\psi_{jt}|\psi_{jt},\psi_{jt},\ldots,\psi_{jt},\ldots,\psi_{jt},\gamma,H \sim \sum\nolimits_{k=1}^{K}\frac{m_k}{\sum_k m_k+\gamma}\delta_{\phi_k}+\frac{\gamma}{\sum_k m_k+\gamma}H \tag{3}$$

Thus, the construction of CRF justly is the process of assigning tables and foods for customers. Actually, the process of assigning tables and foods for customers is respectively corresponding to topic assignment of words and document topic clustering in Mashup documents set. After completing the construction of CRF, we use the Gibbs sampling method to infer the posterior distribution of parameters in the HDP model, and thus obtain topics distribution of whole Mashup documents set.

Similarly, the HDP model construction and topic generation process of Web APIs document set are same to those of Mashup documents set, which are not presented in details.

2.2 Web APIs Recommendation for Mashup Using FMs

2.2.1 Rating Prediction in Recommendation System and FMs

Traditional recommendation system is a user-item two-dimension model. Suppose user set $U = \{u_1, u_2, \ldots\}$, item set $I = \{i_1, i_2, \ldots\}$, the rating prediction function is defined as below:

$$y : U \times I \rightarrow R \tag{4}$$

Here, y represents the rating, i.e. $y(u, i)$ is the rating of user u to item i. The task of rating prediction is to predict the rating of any user-item pairs.

FMs is a general predictor, which can estimate reliable parameters under very high sparsity (like recommender systems) [11, 12]. The FMs combines the advantages of SVMs with factorization models. It not only works with any real valued feature vector like SVMs, but also models all interactions between feature variables using factorized parameters. Thus, it can be used to predict the rating of items for users. Suppose there are an input feature vector $x \in R^{n*p}$ and an output target vector $y = (y_1, y_2, \ldots, y_n)^T$. Where, n represents the amount of input-output pairs, p represents the amount of input features, i.e. the i^{th} row vector $x_i \in R^p$, p means x_i have p input feature values, and y_i is the predicted target value of x_i. Based on the input feature vector x and output target vector y, the 2-order FMs can be defined as below:

$$\hat{y}(x) := w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j=i+1}^{p} x_i x_j \sum_{f=1}^{k} v_{i,f} v_{j,f} \tag{5}$$

Here, k is the factorization dimensionality, w_i is the strength of the i^{th} feature vector x_i, and $x_i x_j$ represents all the pairwise variables of the training instances x_i and x_j. The model parameters $\{w_0, w_1, \ldots, w_{p,} v_{1,1}, \ldots, v_{p,k}\}$ that need to be estimated are:

$$w_0 \in R, w \in R^n, V \in R^{n*k} \tag{6}$$

2.2.2 The Prediction and Recommendation of Web APIs for Mashup Based on FMs

In this paper, the prediction target is a typical classification problem, i.e. $y = \{-1, 1\}$. The Web APIs prediction is defined as a task of ranking Web APIs and recommending adequate relevant Web APIs for the given Mashup. If $y = 1$, then the relevant API will be chosen as a member Web API of the given Mashup. But in practice, we can only obtain a predicted decimal value ranging from 0 to 1 derived from the formula (5) for each input feature vector. We rank these predicted decimal values and then classify them into positive value (+1, the Top-K results) and negative value (−1). Those who have positive values will be recommended to the target Mashup.

As described in Sect. 2.2.1, traditional recommendation system is a two-dimension model of user-item. In our FMs modeling of Web APIs prediction, active Mashup can be regarded as user, and active Web APIs can be regarded as item. Besides the two-dimension features of active Mashup and Web APIs, other multiple dimension features, such as similar Mashups, similar Web APIs, co-occurrence and the popularity of Web APIs, can be exploited as input features vector in FMs modeling. Thus, the two-dimension of prediction model in formula (4) can be expanded to a six-dimension prediction model:

$$y : MA \times WA \times SMA \times SWA \times CO \times POP \rightarrow S \tag{7}$$

Here, *MA* and *WA* respectively represent the active Mashup and Web APIs, *SMA* and *SWA* respectively represent the similar Mashups and similar Web APIs, *CO* and *POP*

respectively represent the co-occurrence and popularity of Web APIs, and S represents the prediction ranking score. Especially, we exploit the latent topics probability of both the documents of similar Mashup and similar Web APIs, to support the model training of FMs, in which these latent topics are derived from our HDP model in the Sect. 2.1.

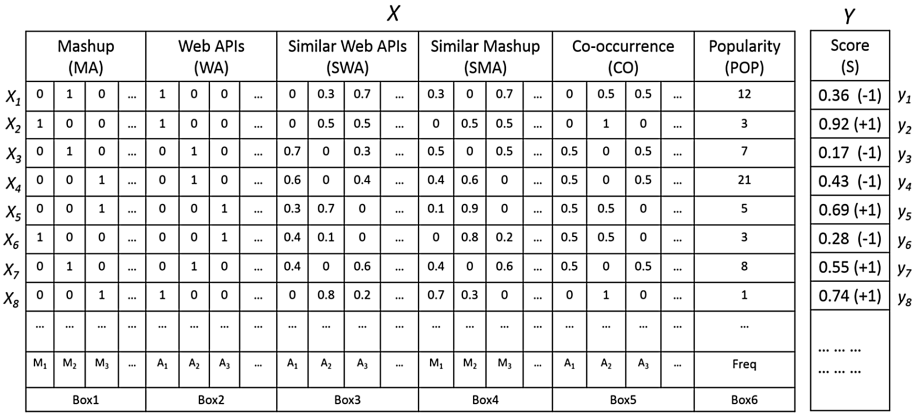

Fig. 2. The FMs model of recommending web APIs for mashup

The above Fig. 2 is a FMs model example of recommending Web APIs for Mashup, in which the data includes two parts (i.e. an input feature vector set X and an output target set Y). Each row represents an input feature vector x_i with its corresponding output target y_i. In the Fig. 2, the first binary indicator matrix (Box 1) represents the active Mashup *MA*. For one example, there is a link between M_2 and A_1 at the first row. The next binary indicator matrix (Box 2) represents the active Web API *WA*. For another example, the active Web API at the first row is A_1. The third indicator matrix (Box 3) indicates *Top-A* similar Web APIs *SWA* of the active Web API in Box 2 according to their latent topics distribution similarity derived from HDP described in Sect. 2.2. In Box 3, the similarity between A_1 and A_2 (A_3) is 0.3 (0.7). The forth indicator matrix (Box 4) indicates *Top-M* similar Mashups *SMA* of the active Mashup in Box 1 according to their latent topics distribution similarity derived from HDP described in Sect. 2.2. In Box 4, the similarity between M_2 and M_1 (M_3) is 0.3 (0.7). The fifth indicator matrix (Box 5) shows all co-occurrence Web APIs *CO* of the active Web API in Box 2 that are invoked or composed in common historical Mashup. The sixth indicator matrix (Box 6) shows the popularity *POP* (i.e. invocation frequency or times) of the active Web API in Box 2 in historical Mashup. Target Y is the output result, and the prediction ranking score S are classified into positive value (+1) and negative value (−1) according to a given threshold. Suppose $y_i > 0.5$, then $S = +1$, otherwise $S = -1$. These Web APIs who have positive values will be recommended to the target Mashup. For example, active Mashup M_1 have two active Web APIs member A_1 and A_3, A_1 will be preferred recommended to M_1 since it have the higher prediction value, i.e. $y_2 > 0.92$. Moreover, in the experiment section, we will investigate the effects of *top-A* and *top-M* on Web APIs recommendation performance.

3 Experiments

3.1 Experiment Dataset and Settings

To evaluate the performance of different recommendation methods, we crawled 6673 real Mashups, 9121 Web APIs and 13613 invocations between these Mashups and Web APIs from ProgrammableWeb. For each Mashup or Web APIs, we firstly obtained their descriptive text and then performed a preprocessing process to get their standard description information. To enhance the effectiveness of our experiment, a five-fold cross-validation is performed. All the Mashups in the dataset have been divided into 5 equal subsets, and each fold in the subsets is used as a testing set, the other 4 subsets are combined to a training dataset. The results of each fold are summed up and their averages are reported. For the testing dataset, we vary the number of score values provided by the active Mashups as 10, 20 and 30 by randomly removing some score values in Mashup-Web APIs matrix, and name them as Given 10, Given 20, and Given 30. The removed score values will be used as the expected values to study the prediction performance. For the training dataset, we randomly remove some score values in Mashup-Web APIs matrix to make the matrix sparser with density 10%, 20%, and 30% respectively.

3.2 Evaluation Metrics

Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are two frequently-used evaluation metrics [15]. We choose them to evaluate Web APIs recommendation performance. The smaller MAE and RMSE indicate the better recommendation quality.

$$MAE = \frac{1}{N}\sum_{ij}\left|r_{ij} - \hat{r}_{ij}\right| \tag{8}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{ij}\left(r_{ij} - \hat{r}_{ij}\right)^2} \tag{9}$$

Here, N is the amount of predicted score, r_{ij} represents the true score of Mashup M_i to Web API A_j, and $\hat{r}_{ij}$ represents the predicted score of M_i to A_j.

3.3 Baseline Methods

In this section, we investigate and compare our proposed method with baseline methods. The baseline methods are briefly described as below:

- ***WPCC.*** Like IPCC [15], Web APIs-based using Pearson Correlation Coefficient method (WPCC), uses PCC to calculate the similarities between Web APIs, and makes recommendation based on similar Web APIs.

- ***MPCC.*** Like UPCC [15], Mashups-based using Pearson Correlation Coefficient method (MPCC), uses PCC to calculate the similarities between Mashups, and predicts Web APIs invocations based on similar Mashups.
- ***PMF.*** Probabilistic Matrix Factorization (PMF) is one of the most famous matrix factorization models in collaborative filtering [8]. It supposes Gaussian distribution on the residual noise of observed data and places Gaussian priors on the latent matrices. The historical invocation records between Mashups and Web APIs can be represented by a matrix $R = \left[r_{ij}\right]_{n\times k}$, and $r_{ij} = 1$ indicates the Web API is invoked by a Mashup, otherwise $r_{ij} = 0$. Given the factorization results of Mashup M_j and Web API A_i, the probability A_i would be invoked by M_j can be predicted by the equation: $\hat{r}_{ij} = A_i^T M_j$.
- ***LDA-FMs.*** It firstly derives the topic distribution of document description for Mashup and Web APIs via LDA model, and then use the FMs to train these topic information to predict the probability distribution of Web APIs and recommend Web APIs for target Mashup. Besides, it considers the co-occurrence and popularity of Web APIs.
- ***HDP-FMs.*** The proposed method in this paper, which combines HDP and FMs to recommend Web APIs. It uses HDP to derive the latent topics probability of both the documents of similar Mashup and similar Web APIs, supporting the model training of FMs. It also considers the co-occurrence and popularity of Web APIs.

3.4 Experimental Results

(1) *Recommendation Performance Comparison*

Table 1 reports the MAE and RMSE comparison of multiple recommendation methods, which show our HDP-FMs greatly outperforms WPCC and MPCC, significantly surpasses to PMF and LDA-FMs consistently. The reason for this is that HDP-FMs firstly uses HDP to derive the topics of Mashups and Web APIs for identifying more similar Mashups and similar Web APIs, then exploits FMs to train more useful information for achieving more accurate Web APIs probability score prediction. Moreover, with the increasing of the given score values from 10 to 30 and training matrix density from 10% to 30%, the MAE and RMSE of our HDP-FMs definitely decrease. It means more score values and higher sparsity in the Mashup-Web APIs matrix achieve better prediction accuracy.

(2) *HDP-FMs Performance vs. LDA-FMs Performance with different topics number*

As we know, HDP can automatically find the optimal topics number, instead of repeatedly model training like LDA. We compare the performance of HDP-FMs to those of LDA-FMs with different topics number. During the experiment, we set different topics number 3, 6, 12, and 24 for LDA-FMs, respectively denoted as LDA-FMs-3/6/12/24. Figures 3 and 4 respectively show the MAE and RMSE of them when training matrix density = 10%. The experimental results in the Figs. 3 and 4 indicate that the performance of HDP-FMs is the best, the MAE and RMSE of LDA-FMs-12 is close to those of HDP-FMs. When the topics number becomes smaller

Table 1. The MAE and RMSE performance comparison of multiple recommendation approaches

	Method	Matrix Density = 10%		Matrix Density = 20%		Matrix Density = 30%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE
Given10	WPCC	0.4258	0.5643	0.4005	0.5257	0.3932	0.5036
	MPCC	0.4316	0.5701	0.4108	0.5293	0.4035	0.5113
	PMF	0.2417	0.3835	0.2263	0.3774	0.2014	0.3718
	LDA-FMs	0.2091	0.3225	0.1969	0.3116	0.1832	0.3015
	HDP-FMs	0.1547	0.2874	0.1329	0.2669	0.1283	0.2498
Given20	WPCC	0.4135	0.5541	0.3918	0.5158	0.3890	0.5003
	MPCC	0.4413	0.5712	0.4221	0.5202	0.4151	0.5109
	PMF	0.2398	0.3559	0.2137	0.3427	0.1992	0.3348
	LDA-FMs	0.1989	0.3104	0.1907	0.3018	0.1801	0.2894
	HDP-FMs	0.1486	0.2713	0.1297	0.2513	0.1185	0.2291
Given30	WPCC	0.4016	0.5447	0.3907	0.5107	0.3739	0.5012
	MPCC	0.4518	0.5771	0.4317	0.5159	0.4239	0.5226
	PMF	0.2214	0.3319	0.2091	0.3117	0.1986	0.3052
	LDA-FMs	0.1970	0.3096	0.1865	0.2993	0.1794	0.2758
	HDP-FMs	0.1377	0.2556	0.1109	0.2461	0.1047	0.2057

(LDA-FMs-3, LDA-FMs-6) or larger (LDA-FMs-24), the performance of HDP-FMs constantly decreases. The observations verify that HDP-FMs is better than LDA-FMs due to automatic obtain the optimal topics number.

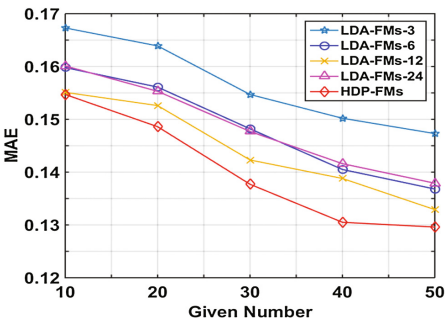

Fig. 3. The MAE of HDP-FMs and LDA-FMs

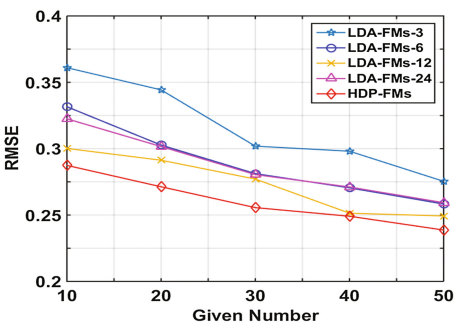

Fig. 4. The RMSE of HDP-FMs and LDA-FMs

(3) *Impacts of top-A and top-M in HDP-FMs*

As described in Sect. 2.2.2, we use *top-A* similar Web APIs and *top-M* similar Mashups derived from HDP as input variables, to train the FMs for predicting the probability of Web APIs invocated by Mashups. In this section, we investigate the

impacts of *top-A* and *top-M* to gain their optimal values. We select the best value of *top-M (top-A)* for all similar *top-A (top-M)* Web APIs (Mashups), *i.e. M = 10 for all top-A similar Web APIs, A = 5 for all top-M similar Mashups.* Figures 5 and 6 show the MAE of HDP-FMs when training matrix density = 10% and given number = 30. Here, the experimental result in the Fig. 5 indicates that the MAE of HDP-FMs is the optimal when *A = 5*. When *A* increases from 5 to 25, the MAE of HDP-FMs constantly increases. The experimental result in the Fig. 6 shows the MAE of HDP-FMs reaches its peak value when *M = 10*. With the decreasing (<=*10*) or increasing (>=*10*) of *M*, the MAE of HDP-FMs consistently raises. The observations show that it is important to choose an appropriate values of *A* and *M* in HDP-FMs method.

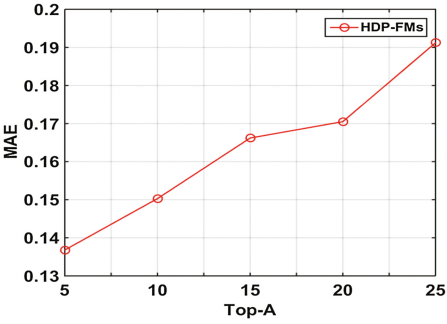

Fig. 5. Impact of top-A in HDP-FMs

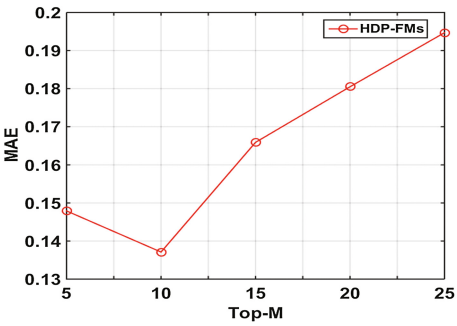

Fig. 6. Impact of top-M in HDP-FMs

4 Related Work

Service recommendation has become a hot topic in service-oriented computing. Traditional service recommendation addresses the quality of Mashup service to achieve high-quality service recommendation. Where, Picozzi [16] showed that the quality of single services can drive the production of recommendations. Cappiello [17] analyzed the quality properties of Mashup components (APIs), and discussed the information quality in Mashups [18]. Besides, collaborative filtering (CF) technology has been widely used in QoS-based service recommendation [15]. It calculates the similarity of users or services, predicts missing QoS values based on the QoS records of similar users or similar services, and recommends the high-quality service to users.

According to the existing results [19, 20], the data sparsity and long tail problem lead to inaccurate and incomplete search results. To solve this problem, some researchers exploit matrix factorization to decompose historical QoS invocation or Mashup-Web API interactions for service recommendations [21, 22]. Where, Zheng et al. [22] proposed a collaborative QoS prediction approach, in which a neighborhood-integrated matrix factorization model is designed for personalized web service QoS value prediction. Xu et al. [7] presented a novel social-aware service recommendation approach, in which multi-dimensional social relationships among potential users, topics, Mashups, and services are described by a coupled matrix model.

These methods address on converting QoS or Mashup-Web API rating matrix into lower dimension feature space matrixes and predicting the unknown QoS value or the probability of Web APIs invoked by Mashups.

Considering matrix factorization rely on rich records of historical interactions, recent research works incorporated additional information into matrix factorization for more accurate service recommendation [4, 8–10]. Where, Ma et al. [9] combined matrix factorization with geographical and social influence to recommend point of interest. Chen et al. [10] used location information and QoS of Web services to cluster users and services, and made personalized service recommendation. Yao et al. [8] investigated the historical invocation relations between Web APIs and Mashups to infer the implicit functional correlations among Web APIs, and incorporated the correlations into matrix factorization model to improve service recommendation. Liu et al. [4] proposed to use collaborative topic regression which combines both probabilistic matrix factorization and probabilistic topic modeling, for recommending Web APIs.

The above existing matrix factorization based methods definitely boost performance of service recommendation. However, few of them perceive the historical invocation between Mashup and Web APIs to derive the latent topics, and none of them use FMs to train these latent topics to predict the probability of Web APIs invoked by Mashups for more accurate service recommendation. Motivated by above approaches, we integrated HDP and FMs to recommend Web APIs for Mashup development. We use HDP model to derive the latent topics from the description document of Mashups and Web APIs for supporting the model training of FMs. We exploit the FMs to predict the probability of Web APIs invocated by Mashups and recommend high-quality Web APIs for Mashup development.

5 Conclusions and Future Work

This paper proposes a Web APIs recommendation for Mashup development based on HDP and FMs. The historical invocation between Mashup and Web APIs are modeled by HDP model to derive their latent topics. FMs is used to train the latent topics, model multiple input information and their interactions, and predict the probability of Web APIs invocated by Mashups. The comparative experiments performed on ProgrammableWeb dataset demonstrate the effectiveness of the proposed method and show that our method significantly improves accuracy of Web APIs recommendation. In the future work, we will investigate more useful, related latent factors and integrate them into our model for more accurate Web APIs recommendation.

Acknowledgements. This work is supported by the National Natural Science Foundation of China under grant No. 61572371, 61572186, 61572187, 61402167, 61402168, State Key Laboratory of Software Engineering of China (Wuhan University) under grant No. SKLSE2014-10-10, Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under grant No. SKLNST-2016-2-26, Hunan Provincial Natural Science Foundation of China under grant No. 2015JJ2056,2017JJ2098,Hunan Provincial University Innovation Platform Open Fund Project of China under grant No.14K037, Education Science Planning Project of Hunan Province

under grant No. XJK013CGD009, and Language Application Research Project of Hunan Province under grant No. XYJ2015GB09.

References

1. Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., Wu, C.: Category-aware API clustering and distributed recommendation for automatic mashup creation. IEEE Trans. Serv. Comput. **8** (5), 674–687 (2015)
2. https://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)
3. Chen, L., Wang, Y., Yu, Q., Zheng, Z., Wu, J.: WT-LDA: user tagging augmented LDA for web service clustering. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 162–176. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45005-1_12
4. Liu, X., Fulia, I.: Incorporating user, topic, and service related latent factors into web service recommendation. In: ICWS 2015, pp. 185–192 (2015)
5. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
6. The, Y., Jordan, M., Beal, M., Blei, D.: Hierarchical dirichlet process. J. Am. Stat. Assoc. **101**(476), 1566–1581 (2004)
7. Xu, W., Cao, J., Hu, L., Wang, J., Li, M.: A social-aware service recommendation approach for mashup creation. In: ICWS 2013, pp. 107–114 (2013)
8. Yao, L., Wang, X., Sheng, Q., Ruan, W., Zhang, W.: Service recommendation for mashup composition with implicit correlation regularization. In: ICWS 2015, pp. 217–224 (2015)
9. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 287–296. ACM (2011)
10. Chen, X., Zheng, Z., Yu, Q., Lyu, M.: Web service recommendation via exploiting location and QoS information. IEEE Trans. Parallel Distrib. Syst. **25**(7), 1913–1924 (2014)
11. Rendle, S.: Factorization machines. In: ICDM 2010, pp. 995–1000 (2010)
12. Rendle, S.: Factorization machines with libFM. ACM Trans. Intell. Syst. Technol. (TIST) **3** (3), 57–78 (2012)
13. Ma, T., Sato, I., Nakagawa, H.: The hybrid nested/hierarchical dirichlet process and its application to topic modeling with word differentiation. In: AAAI 2015 (2015)
14. Teh, Y., Jordan, M., Beal, M., Blei, D.: Sharing clusters among related groups: hierarchical dirichlet processes. Adv. Neural Inf. Process. Syst. **37**(2), 1385–1392 (2004)
15. Zheng, Z., Ma, H., Lyu, M., King, I.: WSRec: a collaborative filtering based web service recommender system. In: ICWS 2009, Los Angeles, CA, USA, 6–10 July, 2009, pp. 437–444 (2009)
16. Picozzi, M., Rodolfi, M., Cappiello, C., Matera, M.: Quality-based recommendations for mashup composition. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 360–371. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16985-4_32
17. Cappiello, C., Daniel, F., Matera, M.: A quality model for mashup components. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 236–250. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02818-2_19
18. Cappiello, C., Daniel, F., Matera, M., Pautasso, C.: Information quality in mashups. IEEE Internet Comput. **14**(4), 14–22 (2010)
19. Huang, K., Fan, Y., Tan, W.: An empirical study of programmable web: a network analysis on a service-mashup system. In: ICWS 2012, 24–29 June, Honolulu, Hawaii, USA (2012)

20. Gao, W., Chen, L., Wu, J., Gao. H.: Manifold-learning based API recommendation for mashup creation. In: ICWS 2015, June 27 - July 2, New York, USA (2015)
21. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
22. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Collaborative web service QoS prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. **6**(3), 289–299 (2013)