Journal of Web Engineering, Vol. 16, No. 7&8 (2017) 595-618

SERVICE RECOMMENDATION BASED ON SEPARATED TIME-AWARE COLLABORATIVE POISSON FACTORIZATION

SHUHUI CHEN[1] YUSHUN FAN[2] WEI TAN[3] JIA ZHANG[4]
BING BAI[1] ZHENFENG GAO[1]

[1]*Department of Automation, Tsinghua University*
Beijing, China
{*chensh13, bb13, gzf13*}*@mails.tsinghua.edu.cn*

[2]*Department of Automation, Tsinghua University*
Beijing, China
fanyus@tsinghua.edu.cn

[3]*IBM Thomas J. WatsonResearch Center*
Yorktown Heights, New York
wtan@us.ibm.com

[4]*Department of Electrical and Computer Engineering, Carnegie Mellon University*
Silicon Valley, California
jia.zhang@sv.cmu.edu

Received February 26, 2017
Revised May 11, 2017

With the booming of web service ecosystems, finding suitable services and making service compositions have become an principal challenge for inexperienced developers. Therefore, recommending services based on service composition queries turns out to be a promising solution. Many recent studies apply Latent Dirichlet Allocation (LDA) to model the queries and services' description. However, limited by the restrictive assumption of the Dirichlet-Multinomial distribution assumption, LDA cannot generate high-quality latent presentation, thus the accuracy of recommendation isn't quite satisfactory. Based on our previous work, we propose a Separated Time-aware Collaborative Poisson Factorization (STCPF) to tackle the problem in this paper. STCPF takes Poisson Factorization as the foundation to model mashup queries and service descriptions separately, and incorporates them with the historical usage data together by using collective matrix factorization. Experiments on the real-world show that our model outperforms than the state-of-the-art methods (e.g., Time-aware collaborative domain regression) in terms of mean average precision, and costs much less time on the sparse but massive data from web service ecosystem.

Keywords: service recommendation, service composition, Time-aware, Poisson Factorization
Communicated by: D. Schwabe & R. Mizoguchi

1. Introduction

With the extensive use of Service-Oriented Architecture (SOA), reusing web services has be-

come a popular way to develop new software services [1]. Composing services to form mashups helps creating value-added software efficiently and effectively [2]. However, as the number of services published on the web gets larger, it becomes a challenge for inexperienced developers to find suitable web services to create service compositions. Under such circumstances, service recommendation becomes a promising solution for assisting service composition's creation.

Existing approaches for service recommendation can be divided into two types: (1) Recommending services based on the non-functional features, i.e., QoS [3–5]. (2) Recommending services based on the functionalities [6–9]. These approaches are generally based on some specific performance indices. But for many inexperienced developers, a natural way to receive services recommendation is to submit a textual query for the desired service composition. For example, a service composition developer's query can be described as "*Search local auctions and classified listings and show them on a map view*". Since this service composition inherently requires services with different functions, the developer will finally choose services from two functions domains: Google maps from mapping domain and ebay services from e-commerce domains. In this process, there are two problems that need to be settled. The first one is that the domain information shall be extracted and purified from the requirement description; the second one is that the most appropriate service shall be selected from different domains and applied to the development. Undoubtedly, these two problems are challenging especially for inexperienced developers. Thus an intelligent algorithm is in desire to recommend relevant services according to developer query.

In recent years Machine learning turns into an effective and efficient way for web service recommendation [10]. Many existing works use LDA (Latent Dirichlet Allocation, [11]) to model user queries and service descriptions, which are in the form of nature language. However, LDA is not good enough to generate accurate latent presentation because of the restrictive assumption of the Dirichlet-Multinomial distribution [12]. As it puts a limitation on the quality of the extracted features from description content and queries, the accuracy of recommendation isn't quiet satisfying.

In previous work [13], we proposed a Time-aware Collaborative Poisson Factorization (TCPF) to solve the problems. Nevertheless we ignore the subtle difference between similar parameters for a more compact model. Moreover, the depth of research on certain issue is not thorough enough subject to the space limitation of the paper [13]. In this paper we (1)improve the model by separating the similar parameters for a higher accuracy; (2)fully explore the principle and mechanism of the mode; (3)introduce the complete formula derivation process and (4)analyze the performance of the parameters. We call the modified model as Separated Time-aware Collaborative Poisson Factorization (STCPF). Same as [13], STCPF also adopts Poisson Factorization as the foundation. Poisson Factorization, which gives Gamma-Poisson distribution assumptions to the components of the dateset, has been proved to be extremely effective and efficient in modeling document latent vectors in text analysis [14–17]. We consider applying Poisson Factorization to the following aspects: firstly, the probability topic model has extremely obvious advantages in treatment process. As one of probability topic models, Poisson Factorization can naturally and easily extract domain information from the texts described by the natural language. Secondly, the Poisson Factorization is different from other probability topic models (such as LDA), that is, when the observed quantity matrix to be treated as the sparse one, the Poisson Factorization has incomparable superiority in the

aspect of calculated performance, which can largely reduce the recommendation time. In this paper, we exploit the idea and propose STCPF model as a better service recommendation to queries. Compared with the previous service recommended algorithms, the precision of the recommended algorithm generated by taking advantage of Poisson Factorization is largely increased.

In addition, similar to [18], information evaporation is also taken into consideration in STCPF. The key idea of information evaporation is that the service heat changes over time. For the concrete manifestation, the services preferred by the developers of service combination will also be changed with the time. Information evaporation is also verified through the experiment in reference [19]. In this paper we introduce the information evaporation by applying the release time of mashups into the model, and giving a larger trust to newer usage records.

The contributions of this paper are as follows:

- By using Poisson Factorization as basic units, we propose the Separated Time-aware Collaborative Poisson Factorization for service recommendation.
- The STCPF model uses multi-factor including text descriptions of both services' and service compositions' historical usage data. By incorporating them together STCPF can get more accurate and eventually better recommendation results.
- Comprehensive experiments on the real-world dataset from ProgrammableWeb.com show that STCPF has faster calculation speed and higher accuracy than the state-of-the-art method in service recommendation.

The rest of this paper is organized as follows. Section II formulates the problem we studied and then gives a brief introduction to related methods. Section III introduces the proposed model including the generative process and set-method of Auxiliary variables. In section IV, the training and recommendation algorithms are described in detail. Section V reports the experimental results. Section VI discusses the related work and section VII draws a conclusion for the whole paper.

2. Background

In this section, we first provide some necessary definitions and formulate the service recommendation for service composition creation problem. Then a brief introduction of two related methods are presented.

2.1. *Problem formulation*

Our views on service recommendation are that a whole recommendation algorithm is consist of two processes, namely establishing a model and using the model for recommending. In the following section we formally define the problem from the centric view.

Definition 1: Model Topology. The set $G = (M, \mathbf{MD}, MT, S, \mathbf{SD}, \mathbf{R})$ is used to describe the topology of the problem. Each symbol of the set is defined as follows: $M = \{m_1, m_2, \ldots, m_i\}$ represents the set of service compositions. $S = \{s_1, s_2, \ldots, s_j\}$ denotes the set of services. $\mathbf{MD}_m = \{w_{m1}, w_{m2}, \ldots, w_{mn_m}\}$ represents the service composition $m \in M$

described by the set of n_m words. Similarly, $\mathbf{SD}_s = \{w_{s1}, w_{s2}, \ldots, w_{sn_s}, \}$represents the service $s \in S$ described by the set of n_s words. $MT = \{t_1, t_2, \ldots, t_M\}$ records the time stamp of service composition release time and has the same amount of service compositions in each time interval. $\mathbf{R} = (r_{ms})_{m=1,s=1}^{M \times S}$ records the historical usage between service compositions and services. If service composition m invokes service s then $r_{ms} = 1$, else $r_{ms} = 0$.

Definition 2: Service Recommendation for service composition Creation. For service composition Creation, a new requirement $l \in \mathbf{QD}$ described as a collection of words $\mathbf{QD}_l = \{w_{l1}, w_{l2}, ...w_{ln_l}\}$. For each query l, the goal is to return a ranked list of services r_{ls} based on G and services with higher probability to be involved will be recommended to the developer.

With these two definitions, we can conclude our problem as how to offer a useful service list r_{ls} for service composition developer while a new requirement l is proposed. Naturally, how to compute the list r_{ls} is the core of the recommendation problem. In order to construct an more effective predictive model than previous one, some essential new mathematic models must be adopted and we particularly introduce them in the following passage.

2.2. *Poisson factorization*

Poisson Factorization(PF) model is proposed recently mainly for content based recommender system [15,20]. The graphical model of PF is shown in figure 1. Matrix $\mathbf{Y}$ can be decomposed into the form of $\mathbf{Y} = \Lambda\mathbf{X}$ with Matrix Factorization (MF).The key difference between PF and MF is that PF assumes that Λ follows the Gamma distribution and $\mathbf{X}$ follows Poisson distribution. For example, given the observed rating y_{ui} of user u to item i, the generative process of Poisson matrix factorization unit is formulated as follows:

$$\begin{aligned} \theta_i &\sim Gamma\left(\lambda_{ia}, \lambda_{ib}\right) \\ \eta_u &\sim Gamma\left(\lambda_{ua}, \lambda_{ub}\right) \\ y_{ui} &\sim Poisson\left(\eta_u^T, \theta_i\right) \end{aligned} \tag{1}$$

Where $\lambda_{\cdot a}$ is the shape parameter of the Gamma distribution, $\lambda_{\cdot b}$ is the rate parameter of the Gamma distribution and η_u^T,θ_i is the shape parameter of Poisson factorization. Specifically, the two distribution are defined as

$$\begin{aligned} Gamma\left(\theta_{\cdot,k}; \lambda_{\cdot a}, \lambda_{\cdot b}\right) &= \frac{{\lambda_{\cdot b}}^{\lambda_{\cdot a}}}{\Gamma\left(\lambda_{\cdot a}\right)} {\theta_{\cdot,k}}^{\lambda_{\cdot a}-1} e^{-\lambda_{\cdot b}\theta_{\cdot,k}} \\ Poisson\left(y_{ui}; \eta_u^T\theta_i\right) &= \left(\eta_u^T\theta_i\right)^{y_{ui}} e^{\left(-\eta_u^T\theta_i\right)/y_{ui}!} \end{aligned} \tag{2}$$

The goal of Poisson Matrix factorization is to get optimal θ_i and η_u to reconstruct original data [17].

In essence, PF is a modified version of matrix factorization [21]. For the user-item recommendation problem, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. It has been widely used in many recommender systems, but the biggest shortcoming is that there are unexplained negative values in the results.

Furthermore, all parameters of PF are Non-negative. This feature is similar to the NMF (Non-negative Matrix Factorization) proposed in [22]. NMF addresses the shortcoming of MF and has outstanding performance in clustering. However NMF needs much more time to convergence when dealing with large dataset.

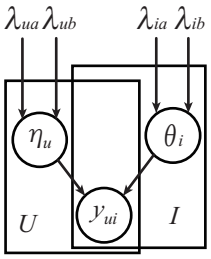

Fig. 1. Graphical model of the PF model. PF is a variant of probabilistic matrix factorization. The observed y_{ui} are modeled with a Poisson, parameterized by the inner product of η_u^T and θ_i.

From the perspective of generating process, PF is a little bit similar to Latent Dirichlet Allocation (LDA). The key difference between them is the choice of prior distribution assumptions of parameters. And with regard to computation amount, PF has more advantages than LDA. The assumption of gamma distribution has many preferential. If θ_{i_1} and θ_{i_2} are gamma variables with parameters $(\lambda_{i_1}, \lambda_{ib})$ and $(\lambda_{i_2}, \lambda_{ib})$ respectively, then their sum $(\theta_{i_1} + \theta_{i_2})$ is a gamma variable with parameters$(\lambda_{i_1} + \lambda_{i_2}, \lambda_{ib})$, this will contribute to great computation efficiency.

In this paper, we use collective matrix factorization to combine several basic PF units together and solve the mashup-side cold-start service recommendation problem as in [19].

2.3. *Collective matrix factorization*

Collective matrix factorization (CMF) is proposed to solve the optimal problems about several relational matrices with some shared elements [23]. For example, suppose there are two rational matrices $M1 \sim f(a, b)$ and $M2 \sim g(a, b)$ with the shared element b , collective matrix factorization combines the lose function together to test error. The modified hybrid objective function is defined as

$$\mathcal{L} = \alpha_1 \mathcal{L}_1 (M1; a, b) + \alpha_2 \mathcal{L}_1 (M2; b, c) \tag{3}$$

where α_1 and α_2 are relative weights and the value of them depends on the validation dataset.

There are several basic Poisson matrix factorization units in our model, and we use collective matrix factorization to synthesize them together.

3. Model Framwork

In this section, we will describe the construction of the two components in our approach. At first the overview of our proposed methodology is given to illustrate the whole business process, and then we will explain two processes of the methodology in details.

3.1. *Overview of methodology*

In our problem setting, the developer only offers a query for the new mashup(service composition). We cannot get the service usage of the mashup because it does not exist. So we cannot get the mashup-service to invoke historical usage because the new mashup does not

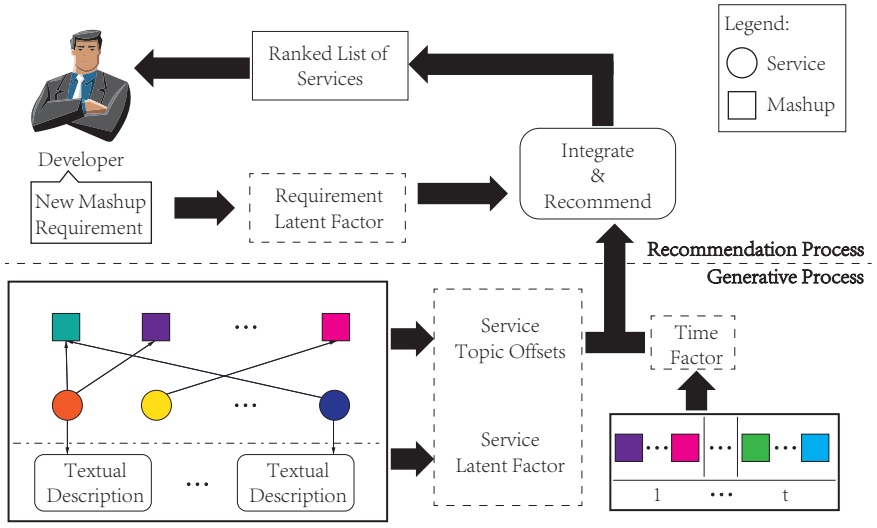

Fig. 2. Methodology of STCPF model. STCPF model utilizes service descriptions and the service composition(mashup) historical usage data separately, then combined with time factor for the more accurate recommendation results.

exist, and cannot directly associate the latent factor with each of them either. That is why we called the problem mashup-side cold-start recommendation [19]. Similar to [19], we use the content of queries to help overcome this cold-start problem.

As shown in figure 2, like TCPF, the whole methodology of STCPF also can be divided into two parts. In generative process, it is intuitive to utilize the service and mashup (service composition) introduction textual content for recommending. Existing mashup historical usage information is also taken into consideration in our model. After the above identified preprocessing we get the service-word, mashup-word and mashup-service matrices. From service-word matrix we get service latent factor and from mashup-service matrix we get service topic offsets. More specially, word latent factor can be obtained during this stage and then be prepared for the recommendation process. The phenomenon of information evaporation is taken into account and use to produce time factor. In the recommendation process, we tackle the query with the word latent factor and get the requirement latent factor. Then the recommendation process integrates the service comprehensive feature information from the generative process and use a poisson factorization-like method to provide service ranking list according to the new mashup requirement of the developer.

Here is an example for the whole process, suppose a service composition developer has a new idea like "regional pet sales market", but he has no idea which service can be used for this service composition. For this question, we first need to prepare the parameters such as service latent factor, mashup latent factor, etc with our generative process in the target service ecosystem, while the query is to input the model as textual content information, the recommendation process will offer the distribution of each services with the parameters calculated in the generative process. In the end, the developer can get a list as "Google Maps, Oodle, AddThis Analytics, CityGrid,..." for reference with our proposed model. Obviously it

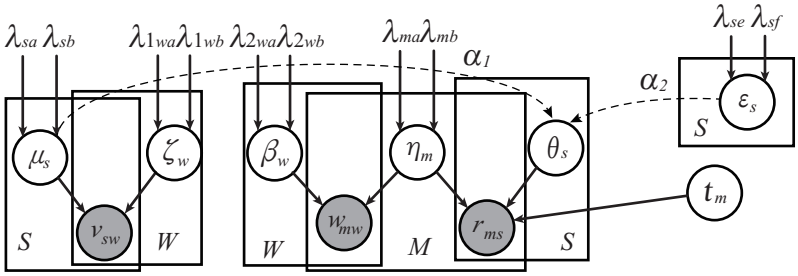

Fig. 3. Graphical model of the generative process. This process takes PF as the foundation, and incorporate them with collective matrix factorization. Comparing with TCPF model, parameter ζ is separated form parameter β.

is more helpful for the inexperienced service composition developer.

The detail of these processes will be explained in below.

3.2. *Generative process*

In this process, we will train the model for recommending. From a high-level perspective, STCPF model can be perceived as the combination of poisson factorization and collective matrix factorization, in which poisson factorization model acts as the basic unit and collective matrix factorization solves several poisson factorization models simultaneously. The generative process of our proposed model is illustrated in figure 3.

As the graphical model shown above, with the preprocessed service-word, mashup-word and mashup-service matrices, we specify the values of v_{sw}, w_{mw} and r_{ms} to be generated from Poisson distributions. Specifically, they are defined to be

$$\begin{aligned} v_{sw} &\sim Poisson\left(\mu_s^T \zeta_w\right) \\ w_{mw} &\sim Poisson\left(\eta_m^T \beta_w\right) \\ r_{ms} &\sim Poisson\left(\eta_m^T \theta_s\right) \end{aligned} \tag{4}$$

In formula 4, each parameter of Poisson distributions is the inner product of two variables. Here β and ζ represent the factor of word, η is the mashup factor. Variables θ is set with the summation of two parts μ and ε. μ_s is the factor of service and ε, which is gained from mashup-service usage, represents the service topic offsets. We incorporate relative weights between μ and ε to adjust their final contributions. Formally, the formula of computing θ is expressed as

$$\theta_s = (\alpha_1 \mu_s + \alpha_2 \varepsilon_s) \tag{5}$$

where α_1 and α_2 are relative weights and the values of them depend on validation datasets.

Because mashup is composed by services, so the word factor ζ drawn from service-word matrix is associated with β which is drawn from mashup-word matrix. If we combine these two parameters together, we can get a compact model as TCPF. But actually, there are some differences between these two parameters. the introduction textual content of mashup is not the set of introduction textual contents of services which are invoked by mashup. For example, in ProgrammableWeb, API "readmill" has a strong emphasis on its social function,

but mashups which invoke it rarely mention about that in the introduction textual contents. Many cases like this lead to the difference between two word latent factors. In STCPF we separate them for a higher accuracy.

From the perspective of the algorithm, we give the assumption that all these variables are subject to Gamma distribution. the hypothesis is reasonable because Gamma distributions are the conjugate priors for the shape parameters of Poisson distributions [17]. Particularly, these variables are defined as

$$\begin{gathered}\zeta_w \sim Gamma\left(\lambda_{1wa}, \lambda_{1wb}\right), \beta_w \sim Gamma\left(\lambda_{2wa}, \lambda_{2wb}\right) \\ \mu_s \sim Gamma\left(\lambda_{sa}, \lambda_{sb}\right), \eta_m \sim Gamma\left(\lambda_{ma}, \lambda_{mb}\right) \\ \varepsilon_s \sim Gamma\left(\lambda_{se}, \lambda_{sf}\right)\end{gathered} \tag{6}$$

Where $\lambda_{.a}(\lambda_{.e})$ and $\lambda_{.b}(\lambda_{.f})$ represent the rate parameter and the shape parameter of Gamma distribution respectively.

The information evaporation theory is also used in recommendation. The core idea is that the recent records are worth more than the past ones. Time factor t_m is adopted to scale as in [19].

In general, the generative process of STCPF model is described as follows,

1. For each word w,
   (a) Draw latent factor $\zeta_w \sim Gamma\left(\lambda_{1wa}, \lambda_{1wb}\right)$ from service-word matrix.
   (b) Draw latent factor $\beta_w \sim Gamma\left(\lambda_{2wa}, \lambda_{2wb}\right)$ from mashup-word matrix.
2. For each service s,
   (a) Draw latent factor $\mu_s \sim Gamma\left(\lambda_{sa}, \lambda_{sb}\right)$.
   (b) For each word w in the introduction content, draw word occurrence count, $v_{sw} \sim Poisson\left(\mu_s^T \beta_w\right)$.
3. For each mashup m,
   (a) Draw latent factor $\eta_m \sim Gamma\left(\lambda_{ma}, \lambda_{mb}\right)$.
   (b) For each word w in the introduction content, draw word occurrence count, $w_{mw} \sim Poisson\left(\eta_m^T \beta_w\right)$.
4. For mashup-service pair (m, s),
   (a) For historical usage information, draw service topic offsets $\varepsilon_s \sim Gamma\left(\lambda_{se}, \lambda_{sf}\right)$.
   (b) Draw the preference response, $r_{ms} \sim Poisson\left(\eta_m^T \theta_s\right)$.
   (c) The adjusted rating $\tilde{r}_{ms} \sim t_m r_{ms}$ according to information evaporation theory, where the time factor t_m is defined as

$$t_m = \lambda_\eta \frac{e^{-\lambda_t\left(t_{current} - t_m\right)}}{\frac{1}{M}\sum_m e^{-\lambda_t\left(t_{current} - t_m\right)}} \tag{7}$$

3.3. *Recommendation process*

When a mashup query l is submitted by the developer, we treat it as a bag of words and get a single-dimensional array query-word w_{ls} as the alternative of the query. Because we have gotten the word latent factor β_w during the generative process, so we only need to calculate the query latent factor η_l with the Poisson Factorization-like model

$$w_{lw} \sim Poisson\left(\eta_l^T \beta_w\right) \tag{8}$$

where the parameter β_w is fixed and $\eta_l \sim Gamma\left(\lambda_{la}, \lambda_{lb}\right)$.

When we have worked out η_l from the query, we compute the expectation of the dot product with η_l and θ_s (learned from STCPF). Finally, for the specified mashup requirement l, a ranked list r_{ls} which represents the score of services will be provided for the developer.

$$r_{ls} = E\left[\eta_l^T \theta_s\right] \tag{9}$$

4. Recommendation Approach

In this section, we illuminate how to use the methodology described above to carry out the factual model to solve the real-world problem. At first we describe the auxiliary variables which are indispensable for Variational inference. Then how the optimal parameters of the model are obtained is illustrated. At last we explain how to use this well-trained model for a real-world recommendation at length.

4.1. *Auxiliary variables*

With the preprocessed matrices v,w and r, our goal is to infer the word latent factor ζ and β, the mashup latent factor η, the service latent factor μ and service topic offsets ϵ. The exact posterior distribution can be described as

$$q\left(\zeta, \beta, \eta, \mu, \epsilon | v, w, s\right) \tag{10}$$

As many Bayesian models, this probability is intractable to compute directly because there are some coupling relationships between variables. In the above formula, each Gamma variable is independent but every Poisson variable is formed from two Gamma variables. Here the Variational Bayesian Inference is used to address this issue.

Variational Bayesian Inference is an optimization-based strategy for approximating posterior distributions in complex probabilistic models [15, 24]. This algorithm assumes a family of distributions over the target latent variable. The supposed family of distributions indexed by free "Variational" parameters and by adjusting the parameters we find a member of this family closet to the true posterior in Kullback-Liebler (KL) divergence. With this algorithm the inference problem turns to an optimization problem which is easier to accomplish.

To implement Variational Bayesian Inference, we augment STCPF with auxiliary variables as in [15]. For each mashup-word pair (m, w), we add K latent variables

$$w_{mw,k} \sim Poisson\left(\eta_{m,k}^T \beta_{w,k}\right) \tag{11}$$

Table 1. STCPF:latent variable, complete conditionals and variational parameters.

Latent Variable	Type	Complete Conditional	Variational Parameters
Model Training			
$\zeta_{w,k}$	Gamma	$\lambda_{1wa}+\sum_s \nu_{sw,k},\ \lambda_{1wb}+\sum_s \mu_{s,k}$	$\tilde{\zeta}^{shp},\ \tilde{\zeta}^{rte}$
$\beta_{w,k}$	Gamma	$\lambda_{2wa}+\sum_m w_{mw,k},\ \lambda_{2wb}+\sum_m \eta_{m,k}$	$\tilde{\beta}^{shp},\ \tilde{\beta}^{rte}$
$\eta_{m,k}$	Gamma	$\lambda_{ma}+\sum_w w_{mw,k}+\sum_s \left(r^a_{ms,k}+r^b_{ms,k}\right),$ $\lambda_{mb}+\sum_w \beta_{w,k}+\sum_s \left(\alpha_1\mu_{s,k}+\alpha_2\varepsilon_{s,k}\right)$	$\tilde{\eta}^{shp},\ \tilde{\eta}^{rte}$
$\mu_{s,k}$	Gamma	$\lambda_{sa}+\sum_w v_{sw,k},\ \lambda_{sb}+\sum_w \zeta_{w,k}$	$\tilde{\mu}^{shp},\ \tilde{\mu}^{rte}$
$\varepsilon_{s,k}$	Gamma	$\lambda_{se}+\sum_m r^b_{ms,k},\ \lambda_{sf}+\sum_m \eta_{m,k}$	$\tilde{\varepsilon}^{shp},\ \tilde{\varepsilon}^{rte}$
$w_{mw,k}$	Mult	$\log\left(\eta_{m,k}\right)+\log\left(\beta_{w,k}\right)$	ϕ_{mw}
$v_{sw,k}$	Mult	$\log\left(\mu_{s,k}\right)+\log\left(\beta_{w,k}\right)$	ψ_{sw}
$r_{ms,k}$	Mult	$\begin{cases}\log(\alpha_1)+\log\left(\eta_{m,k}\right)+\log\left(\mu_{s,k}\right), & if k\le K\\ \log(\alpha_2)+\log\left(\eta_{m,k}\right)+\log\left(\varepsilon_{s,k}\right), & if K<k\le 2K\end{cases}$	ξ_{ms}
Recommending			
$\eta_{l,k}$	Gamma	$\lambda_{la}+\sum_w w_{lw,k},\ \lambda_{lb}+\sum_w \beta_{w,k}$	$\tilde{\eta}^{shp},\ \tilde{\eta}^{rte}$
$w_{lw,k}$	Mult	$\log\left(\eta_{l,k}\right)+\log\left(\beta_{l,k}\right)$	ϕ_{lw}

which are integers and satisfy $w_{mw}=\sum_k w_{mw,k}$. This formula-transform process utilizes the probability addible attribute of Poisson distribution. Each variable $w_{mw,k}\,(\forall k\in K)$ can be regarded as the contribution from component k to the observed data w_{mw}. Similar operation is applied for $v_{sw,k}$. A tiny difference appearing while the formula-transform process to mashup-service pair (m,w) for $\theta_{s,k}$. The parameter $r_{ms,k}$ can be broken into two parts as

$$\begin{aligned} r^a_{ms,k} &\sim Poisson\left(\alpha_1\eta^T_{m,k}\mu_{s,k}\right)\\ r^b_{ms,k} &\sim Poisson\left(\alpha_2\eta^T_{m,k}\varepsilon_{s,k}\right)\end{aligned} \tag{12}$$

The summation of them should satisfy the equation $r_{ms}=\sum_k \left(r^a_{ms,k}+r^b_{ms,k}\right)$. In response, each Gamma variables are changed into K latent factors. the expanded varaibles are shown in the first column of table 1.

Based on changed variables, now we define the mean-field variational family as below

$$\begin{aligned} &q\left(\zeta,\beta,\eta,\nu,\delta,\varepsilon,w,v,c,r\right)=\\ &\prod_{w,k} q\left(\zeta_{w,k}\right)q\left(\beta_{w,k}\right)\prod_{m,k} q\left(\eta_{m,k}\right)\prod_{s,k} q\left(\mu_{s,k}\right)q\left(\varepsilon_{s,k}\right)\\ &\prod_{mw,k} q\left(w_{mw,k}\right)\prod_{sw,k} q\left(v_{sw,k}\right)\prod_{ms,k} q\left(r_{ms,k}\right)\end{aligned} \tag{13}$$

In formula 13, the latent variables $\varepsilon_{s,k}$, $\mu_{s,k}$, $\eta_{m,k}$, $\beta_{w,k}$, $\zeta_{w,k}$ follow Gamma distribution. As shown in the forth column of table 1, we denote the shape parameter with the superscript "*shp*" and the rate parameter with the superscript "*rte*". The variables w_{mw}, v_{sw} are multinomials. r_{ms} is also a multinomial but its variational parameter ξ_{ms} is a point in $2K$-simplex.

There are two categories of latent variables we need to optimize. The first category is Gamma variable and the other is Poisson variable. Here we select one variable from each category respectively to illuminate the iterative algorithm. For Gamma variable, we take ε_s as an example. For each $\varepsilon_{s,k}\,(\forall k \in K)$, we extract the relevant terms from formula 13. Then we get the equation as follow,

$$\begin{aligned} & q\left(\varepsilon_{s,k}|r,\eta,\lambda_{se},\lambda_{sf}\right) \\ & \propto \varepsilon_{s,k}^{\lambda_{se}-1} e^{-\lambda_{sf}\varepsilon_{s,k}} \prod_m \left(\eta_{m,k}\varepsilon_{s,k}\right)^{r_{ms,k}} e^{-\eta_{m,k}\varepsilon_{s,k}} \\ & \propto \varepsilon_{s,k}^{\lambda_{se}+\sum_m r_{ms,k}-1} e^{-\left(\lambda_{sf}+\sum_m \eta_{m,k}\right)\varepsilon_{s,k}} \\ & = Gamma\left(\lambda_{se}+\sum_m r_{ms,k}, \lambda_{sf}+\sum_m \eta_{m,k}\right) \end{aligned} \tag{14}$$

This formula can be considered as the probability of Gamma distribution with "Variational" parameters. The update for the "variational" parameters is

$$\left(\tilde{\varepsilon}_{s,k}^{shp}, \tilde{\varepsilon}_{s,k}^{rte}\right) = \left(\lambda_{se}+\sum_m r_{ms,k}^{b}, \lambda_{sf}+\sum_m \frac{\tilde{\eta}_{m,k}^{shp}}{\tilde{\eta}_{m,k}^{rte}}\right) \tag{15}$$

where $\tilde{\varepsilon}^{shp}$ means the shape of Gamma distribution and $\tilde{\varepsilon}^{rte}$ means the rate of Gamma distribution. $\frac{\tilde{\eta}_{m,k}^{shp}}{\tilde{\eta}_{m,k}^{rte}}$ in the formula is the mean of the gamma variable $\eta_{m,k}$ and can be labeled as $E\left[\eta_{m,k}\right] = \frac{\tilde{\eta}_{m,k}^{shp}}{\tilde{\eta}_{m,k}^{rte}}$.

The Gamma distribution has two parameters $(\lambda_{se}, \lambda_{sf})$. We can initialize these two parameters separately for higher accuracy. And we also can initialize the parameter

$$\lambda_{sf} = c \cdot \lambda_{se} \tag{16}$$

For convenience as in [16]. The scale c is update as $c^{-1} = \frac{1}{SK}\sum_{s,k} E\left[\varepsilon_{s,k}\right]$.

For Poisson variable, we choose w_{mw} as an example. Each $w_{mw,k}\,(\forall k \in K)$ is a Poisson distribution and $w_{mw} = \sum_k w_{mw,k}$. If we choose the appropriate parameters $\phi_{mw,k}$ and hold the following formula

$$w_{mw,k} = w_{mw}\phi_{mw,k} \tag{17}$$

where w_{mw} can also be considered as a multinomial and $\sum_k \phi_{mw,k} = 1$. Here we assume that there is a probability Poisson distribution p and approach it to the true probability distribution with Variational Bayesian inference method. And we get the complete conditional distribution $\phi_{mw,k}$ as below,

$$\begin{aligned} & \phi_{mw,k} = \frac{\eta_{m,k}\beta_{w,k}}{w_{mw}} \\ & \propto \exp\left(\log\left(E_p[\eta_{m,k}\beta_{w,k}]\right)\right) \\ & = \exp\left(E_p\left[\log(\eta_{m,k}\beta_{w,k})\right]\right) \\ & \propto \log(\eta_{m,k}) + \log(\beta_{w,k}) \end{aligned} \tag{18}$$

Then the update for the multinomial ϕ_{mw} is

$$\phi_{mw,k} \propto \exp\Big\{\Psi(\tilde{\eta}^{shp}_{m,k}) - \log(\tilde{\eta}^{rte}_{m,k}) + \Psi(\tilde{\beta}^{shp}_{w,k}) - \log(\tilde{\beta}^{rte}_{w,k})\Big\} \tag{19}$$

where $\Psi(\cdot)$ is the digamma function and $E_p\left(\log \eta_{m,k}\right) = \Psi(\tilde{\eta}^{shp}_{m,k}) - \log(\tilde{\eta}^{rte}_{m,k})$.

Similar to formula 15 and 19, we integrate list of conditional distributions for all latent variables as the third column in Table 1.

4.2. *Model training*

The remaining problem is to find the complete conditional distribution [15] of each latent variable. In coordinate ascent we can optimize one variational parameter with the others fixed, and We iterate this process repeatedly until we get the satisfactory results near to the optimal values.

Note that Poisson matrix factorization is non-negative. With formula 17 we can easily deduce that all $\phi_{mw,k}\,(\forall k \in K)$ can set equal to zero if w_{mw} is zero. Similar process is applied to initialize ψ_{sw} and ζ_{ms}. It is very useful for decreasing the complexity of variational parameter space and the computation of parameter learning.

In summary, the model training stage is described as algorithm 1.

The model training stage can be seen as an offline stage. It only need to be conducted once at the start of each time interval. There are also some techniques for improving the performance of algorithm. while doing the experiment, the resulting topics and proportions of LDA can be used to initialize the means of Gamma distribution. This method can help to reduce the convergence time of the algorithm.

4.3. *Recommending*

As described in section , we need to calculate the expectation of the dot product with η_l and θ_s. But because of Variational Bayesian Inference we replaced it with the expectation under the probability distribution of p. Formally, r_{ls} can be expressed as

$$r_{ls} = E\left[\eta_l^T \theta_s\right] \approx E_p\left[\eta_l^T \theta_s\right] \tag{20}$$

The recommending stage is described as algorithm 2. This stage can be seen as a online stage and it need to be calculate every time when receiving a query.

5. Experiments

In this section, we explain how we applied our STCPF service recommendation approach to service composition creation on a real-world dataset. First we introduce the real-world dataset used in the experiments. Next we illuminate the evaluation metrics, adoption baseline methods and the parameter setting of the proposed model. Then, the recommendation result of the methods is given for comparisons. Last, we summarize some interesting features of STCPF.

5.1. *Dataset preparation*

Algorithm 1: Model training for STCPF

Input:

1) v: Service-word matrix
2) w: Mashup-word matrix
3) r: Mashup-service matrix
4) K: latent factor number
5) $(\lambda_{1wa,k}, \lambda_{1wb,k}), k = 1 : K$: Parameters of Gamma distribution about word latent factor ζ
6) $(\lambda_{2wa,k}, \lambda_{2wb,k}), k = 1 : K$: Parameters of Gamma distribution about word latent factor β
7) $(\lambda_{sa,k}, \lambda_{sb,k}), k = 1 : K$: Parameters of Gamma distribution about service latent factor μ
8) $(\lambda_{ma,k}, \lambda_{mb,k}), k = 1 : K$: Parameters of gamma distribution about mashup latent factor η
9) $(\lambda_{se,k}, \lambda_{sf,k}), k = 1 : K$: Parameters of gamma distribution about service topic offsets ϵ
10) (α_1, α_2): relative weights about service latent factor

Output:

1) β: word latent factor drawn from mashup-word matrix
2) θ: service latent factor

Procedure:

For latent factor $k = 1 : K$
  With $(\lambda_{1wa,k}, \lambda_{1wb,k})$, calculate $\zeta_{w,k}$
  With $(\lambda_{2wa,k}, \lambda_{2wb,k})$, calculate $\beta_{w,k}$
  With $(\lambda_{ma,k}, \lambda_{mb,k})$, calculate $\eta_{m,k}$
  With $(\lambda_{sa,k}, \lambda_{sb,k})$, calculate $\mu_{s,k}$
  With $(\lambda_{se,k}, \lambda_{sf,k})$, calculate $\epsilon_{s,k}$
End
Repeat
  For each service description $v_{sw} > 0$
    According to the fomula in table 1, for each $k \in K$ update $\psi_{sw,k}$
  End
  For each mashup description $w_{mw} > 0$
    According to the fomula in table 1, for each $k \in K$ update $\phi_{mw,k}$
  End
  For each mashup-service pair $r_{ms} > 0$
    According to the fomula in table 1, for each $k \in K$ update $\xi_{ms,k}$
  End
  For latent factor $k = 1 : K$
    $w_{mw,k} = w_{mw}\phi_{mw,k}$
    $v_{sw,k} = v_{sw}\psi_{sw,k}$
    $r_{ms,k} = r_{ms}\xi_{ms,k}$
  End
  According to the formulas in table 1,update $\zeta_{w,k}$, $\beta_{w,k}$, $\eta_{m,k}$, $\mu_{s,k}$ and $\epsilon_{s,k}$ with parameters $w_{mw,k}$, $v_{sw,k}$ and $r_{ms,k}$
Until convergence
$\theta = \alpha_1\mu + \alpha_2\epsilon$

Algorithm 2: Recommending for STCPF

Input:
1) β: word latent factor
2) θ: service latent factor
3) w_{lw}: query-word matrix form developer
4) $(\lambda_{la,k}, \lambda_{lb,k}), k = 1 : K$: Parameters of gamma distribution about query latent factor η

Output:
1) r_{ls}: Ranked list of services for developer

Procedure:
01. **For** latent factor $k = 1 : K$
02. With $(\lambda_{la,k}, \lambda_{lb,k})$, calculate $\eta_{m,k}$
03. **End**
04. **Repeat**
05. **For** query $w_{lw} > 0$
06. According to the fomula in table 1, for each $k \in K$ update $\phi_{lw,k}$
07. **End**
08. return r_{ls} by equation 20

To the best of our knowledge, PromgrammableWeb.com is a successful business web with a large number of web services(APIs) and their service compositions(mashups). We crawled the data of APIs and mashups from September 2005 to February 2015 from PrgrammableWeb.com. Each service contains metadata such as service name and textual description. Each mashup has the information of mashup name, creation date, textual description and the invoke relationship of services. At last we record the words that have ever used in the service and mashup textual description. After removing some meaningless mashups which never invoke service, the basic properties of the dataset summarized as table 2.

Table 2. Dataset on ProgrammableWeb.com

Number of services	12711
Number of services used in at least one mashup	1120
Number of mashups	6120
Size of vocabulary	21328

5.2. *Evaluation metric*

Mean Average Precision at top N (MAP@N) is a widely accepted measure in information retrieval and recommendation system. It is defined as follows,

$$MAP@N = \frac{1}{|M_q|} \sum_{m \in M_q} \frac{1}{N_q} \sum_{s \in S_m} \left[\frac{n(m,s)}{r(m,s)} \cdot I(m,s) \right] \tag{21}$$

Where $|M_q|$ denotes the number of mashups queries. $N_q = \min(N, |s_m|)$ and s_m represent the set of actually used services in queries $m \in M_q$. For each service $s \in \mathbf{s}$, $n(m,s)$ means

the ranking position of s for s both in the S_m and recommendation list while $r(m, s)$ refers to the ranking position of s which is only in the recommendation list.

To avoid the randomness of the test sample, we test several groups of testing data and get a mean value for the last MAP@N results. A higher number of MAP@N means a better accuracy of the recommendation method.

5.3. *Baseline methods*

To evaluate the performance of STCPF, we compare it with TCPF and other five types of common recommendation methods as baseline methods: Matrix factorization and Service-description-based Matching are popularity-based method. these two methods are universally applicable with the advantage of handy calculation and convenient implement. Both of them work coordinating with LDA. Collaborative Poisson factorization is based on our proposed model but ignore the informatin evaporation. Mashup-description-based collaborative filtering is introduced by Y. Zhong [18] in 2014 and time-aware collaborative domain regress was introduced by B. Bai [19] in 2015 as the state-of-the-art models at the time.

5.3.1. ***Baseline method 1:*** *LDA + Matrix Factorization (MF)*

In this method we first use LDA to extract the latent factors of mashup description and the developer's query. Then use MF to factorize the matrix R_{ms} for the optimized latent factors of services. The recommendation rating defined as follows,

$$MF(q, s) = \mu_q v_s \tag{22}$$

Where μ_q denotes the topic proportions of query and v_s denotes the topic proportions of service $s \in S$.

5.3.2. ***Baseline method 2:*** *Collaborative Poisson Factorizaiton (CPF)*

CPF is based on our proposed model and sets time factor t_m to the constant value 1. It means that CPF ignores the information evaporation.

5.3.3. ***Baseline method 3:*** *Service-description-based Matching (SDM)*

The core idea of SDM is that the query of a new mashup is semantically similar to the description of services. The steps of this algorithm are described as below. First, we use the LDA model for extracting the latent factors of services' description and developer's query respectively. Then for $s \in S$ we calculate the cosine similarities $sim(q, s)$ between them as the result for recommendation rating.

$$SDM(q, s) = sim(q, s) \tag{23}$$

5.3.4. ***Baseline method 4:*** *Mashup-description-based Collaborative Filtering (MDCF)*

MDCF is based on the idea of traditional neighborhood-based collaborative filtering, assuming that similar mashups have a higher possibility to use similar service. MDCF uses LDA model to calculate the topic proportions of each mashup description $m \in M$ and the developer's

query q respectively. Then the model calculates the cosine similarity $sim\,(q,m)$ with time factor t_m applied. The equation defined as bellows,

$$MDCF\,(q,s) = \frac{\sum\limits_{m \in U(N,q)} [sim\,(q,m)\,t_m]}{\sum\limits_{m \in U(N,q)} sim\,(q,m)} \tag{24}$$

where $U\,(N,q)$ denotes the Top N similar mashups with the developer's query.

5.3.5. ***Baseline method 5:*** *Time-aware Collaborative Domain Regression (TCDR)*

TCDR is similar to our proposed model except that the PF model is replaced by LDA model. The detailed process is described in [19]. By applying LDA model TCDR learns the topic proportions of services $v_s\,(s \in S)$ and the topic proportions of developer's query ζ_q. Then the recommendation rating defined as follows, $TCDR\,(q,s) = \zeta_q v_s$

5.3.6. ***Baseline method 6:*** *Time-aware Collaborative Poisson Factorization (TCPF)*

TCPF combines the parameters ζ and β into one parameter for a more compact model. The rest of the model is similar to STCPF.

5.4. *Experiment settings*

We segment the data with thirty mashups as a group in chronological order instead of shredding the dataset monthly. we do this because in the real dataset There is very few mashups while adopting a monthly time granularity. And we can get a much higher recommending accuracy by adjusting the parameters in these months but it is useless for the whole dataset. By this way we avoid the tremendous differences of the amount within different months and get 204 groups.

In our experiment we select the last 30 groups and treat them as the test data while its previous data serves as the training data. As shown in figure 4, the dataset is divided by the moving cutoff timestamp. The last group is treated as training set and all of the previous data is regarded as a whole for testing. In addition, services which have never been used are removed while testing the model. And then we run the algorithms for the value of MAP@N. We repeat this process for all of the last 30 groups and get a mean value as the last result. Actually these 30 groups contain the mashups from May 2012 to February 2015. This means STCDF has been tested adequately since the test date has covered a wide range of time span.

The parameters of the model are set as follows. For all method we set topic number $K = 40$. For CPF, TCPF and STCPF, we set $\alpha = \begin{bmatrix} 1 & 1 \end{bmatrix}$. For CPF and TCPF, the shape parameter λ_{wa} is set to 0.07 and other shape parameters are set to 0.03. For STCPF the shape parameter λ_{1wa} is set to 0.27, λ_{2wa} is set to 0.07 and other shape parameters are set to 0.03. The rate parameters are set following the formula 16 for convenience. For LDA model we set the hyper-parameters $\alpha = 1.25$ and $\beta = 0.01$. For MDCF, N is set to 250. For t_m in TCPF, STCPF and TCDR, λ_η is set to 1 and λ_t is 0.08.

5.5. *Performance comparison*

The comparison results are shown in figure 5. From the result of SDM, we can get a conclusion

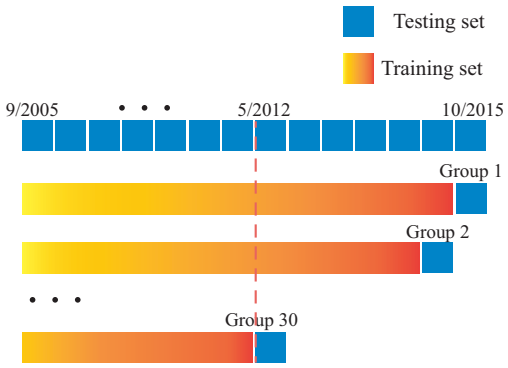

Fig. 4. Generation of training and testing data sets. Each testing set has thirty mashups. The training sets are treated as a whole and the color of the gradient means the different influence with the information evaporation.

that it is unreliable to judge whether a service should be invoked by a mashup or not merely based on semantic similarity. MDCF is a neighborhood-based method in essence. For taking the historical information into consideration, MDCF get a small performance promotion. MF is a better method than MDCF, for this model captures the signals encompassed in the mashup historical usage. TCDR continues to improve the performance by taking information evaporation into consideration. CPF adopts the prior hypothesis of the data and gets a better performance. TCPF synthesizes the merits of TCDR and CPF, promotes the accuracy of recommendation significantly, At last STCPF separates the parameters and outperforms all the baseline methods.

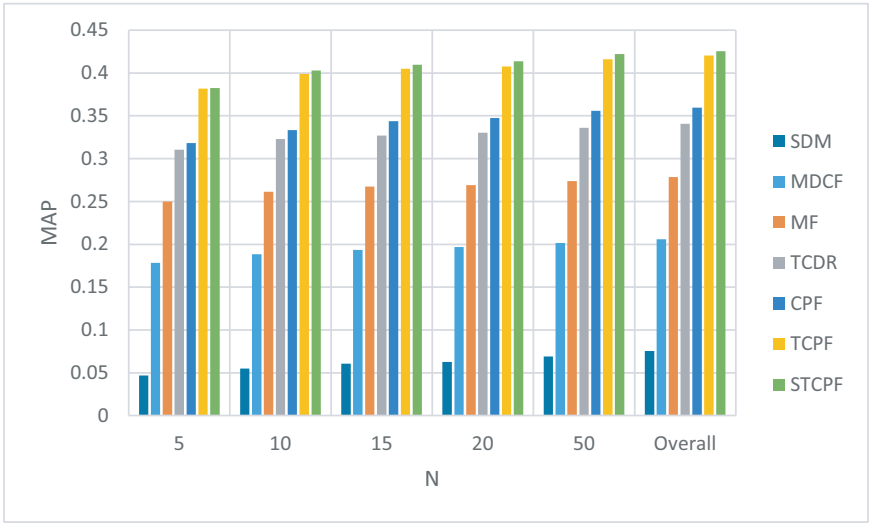

Fig. 5. MAP@N of the different methods. STCPF model outperforms all the baseline methods in accuracy.

The detailed performance is summarized in Table 3. Comparing with TCPF, after sepa-

Table 3. PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS ON MAP@20 AND OVERALL MAP

Recommendation Algorithm	MAP@5	MAP@20	Overall MAP
SDM	4.7%	6.25%	7.53%
MDCF	17.84%	19.34%	20.58%
MF	24.99%	26.9%	27.85%
TCDR	31.05%	32.71%	34.06%
CPF	31.82%	34.37%	35.96%
TCPF	38.17%	40.48%	42.05%
STCPF	**38.23%**	**40.96%**	**42.55%**

rating the parameters, STCPF promotes the performance of about 0.5% while topic number K is 40.

5.6. *Parametric analysis*

Actually, The service recommendation problem which is based on Poisson factorization is not a convex optimization problem. So it is of small significance to analyze how to set and adjust parameters for a better results. But it is so instructive to summarize the rules of how to optimize and adjust various parameters.

5.6.1. *Initialization of Gamma parameters*

As shown in algorithm 1, we need to initialize the parameters of Gamma distributions. Of course we can get the best possible result if we can initialize the parameters closer to the real values. But in practice we can not know the actual values in advance. Here we need some tracks to reduce the number of attempts.

Owing to the non-negative feature of Poisson factorization, the iterative technique of the model is adding some positive values to the parameters. So the better approach is setting the parameters to be smaller than their actual values deliberately. In our experiment we set the shape parameters as low as 0.03.

5.6.2. *Factors contribution*

In our proposed model the service latent factor consists of two parts: one is drawn from service description content information and the other is drawn from the service historical usage information. Each of them can candidate their effectiveness for the problem alone. Here we discuss them separately for the more accurate quantitative analysis.

We calculate the recommender accuracy in the three conditions: using the service description content, using the service historical information and using both of them. The results are shown in figure 6. we can see that the result with using the service description information is better than that using the historical usage information. The reason is that people may prefer to select service due to functional requirements rather than imitate other developers. We get a higher accuracy while we use two kinds of data. It can be very intuitive to know that we can get higher accuracy if we utilize more data.

5.6.3. *Impact of Number of Latent Factors*

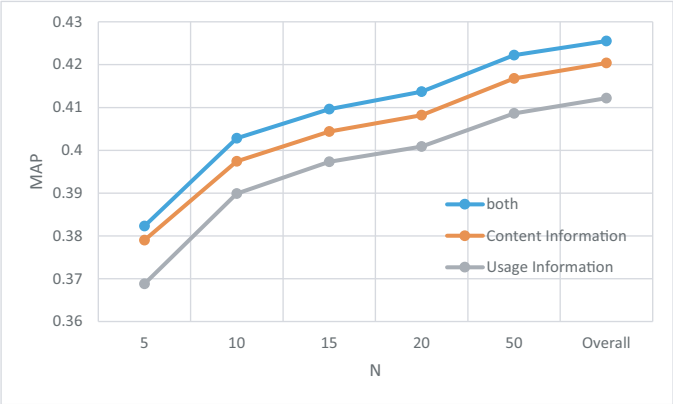

Fig. 6. MAP@N with service latent factor contain the information of different data. The bottom polyline represents the MAP while service latent factor only uses the service historical usage data. The middle one is the MAP while service latent factor only uses service description content, and the top polyline represents the MAP while service latent factor uses both two kinds of data.

As explained in section 2.2, we can easily get the formulas as

$$\begin{aligned} \theta_{i_1} &\sim Gamma\left(\lambda_{i_1}, \lambda_{ib}\right) \\ \theta_{i_2} &\sim Gamma\left(\lambda_{i_2}, \lambda_{ib}\right) \\ \theta_{i_1} + \theta_{i_2} &\sim Gamma\left(\lambda_{i_1} + \lambda_{i_2}, \lambda_{ib}\right) \end{aligned} \tag{25}$$

Let us suppose that the most appropriate number of latent factors is k. According to this character of Gamma distribution, we can consider that some latent factors split into small latent factors while k increase to k_1, and some similar latent factors merge into a bigger one while k reduce to k_2.

It is the very characteristic that we can get a good result with a small value of k for the convenience of computation. We demonstrate the MAP of the proposed methods with different values of k in figure 7. By contrasting two lines which are darwed by STCPF and TCPF model, we can easily see that there are little differences in results while latent factors k is small(e.g.40). But with the increase of the parameters k, the differences of latent factors of mashup-sets and service-sets enlarge gradually. This change leads to the difference of the accuracy of STCPF and TCPF. From the figure we can also see another interesting feature, that is although the most appropriate number of k is 150, there is little difference in results even through the number of latent factors varies dramatically, unless the number of latent factors is too small (e.g. 20). Furthermore, a small k means we can save a great lot of time while doing calculation. In our experiment k is set to 40 for both of computational complexity and accuracy.

5.7. *Time complexity analysis*

TCPF has some wonderful features inherited from PF model. The most outstanding feature is this method has a very low computational cost. As described in the Algorithm, the computational cost of the model mainly depends on the number of non-zero elements in matrices

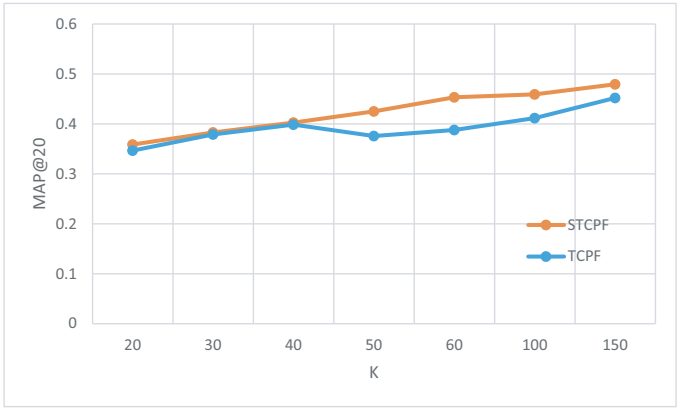

Fig. 7. MAP@20 of the proposed approach with the number of latent factors varied (all other parameters are fixed).

such as mashup descriptions matrix, service descriptions matrix and mashup-service historical usage matrix. TCPF makes calculation only when the elements of these matrices are not zero during iteration. Because of this character, TCPF is specifically more suitable for the massive, sparse and long-tail data comparing with other model. Coincidentally, the data we crawled from ProgammableWeb just conforms to this type. And, this character is conformed to the vast majority of the situation when users call for services. We analyze all observed matrices after data cleaning and get the results as table 4.

Table 4. EXPERIMENT DATA ANALYSIS

Statistic Info	Value	Sparsity
Size of vocabulary	21328	
Average # words in one service description	38.59	99.82%
Average # words in one mashup description	14.75	99.93%
Number of services for computation	1120	
Average # services in one mashup description	1.95	99.83%

We choose r_{ms} to illustrate the advantage of this character. The dimension of $[r_{ms}]$ is 5709×1120. Compared with other methods which need to store and calculate all the elements (zero ones and non-zero ones), TCPF and STCPF only consider the non-negative elements. So these two models only need to store and calculate the sparse matrix which the number of elements is about 5709×1.95. Same conclusion can be drawn for w_{mw} and v_{sw}. To quantifiably verify the efficiency of these two models, we compare the learning time cost of with TCDR. All of these algorithms set 50 iterations to get the training model. The results is provided in Table 5.

Table 5. THE COMPARISON OF TRAINING TIME

Model	Cost of one iteration	Total time of model training
TCPF	1.4s	70s
STCPF	1.4s	71s
TCDR	27.3s	1365s

As we can see, TCPF and STCPF cost much less time than TCDR, which confirms the

analysis above. The calculation time of STCPF model is approximately equaling to TCPF model. The reason is that the computational elements are not increasing while STCPF model separates parameter parameter β of TCPF model into two parameters ζ and β . the iterate formulas are listed as follows. formula 26 belongs to TCPF model and formula 27 belongs to STCPF model.

$$\left(\tilde{\beta}^{shp}, \tilde{\beta}^{rte}\right) = \left(\lambda_{wa} + \sum_{m} w_{mw,k} + \sum_{s} \nu_{sw,k}, \lambda_{wb} + \sum_{m} \eta_{m,k} + \sum_{s} \mu_{s,k}\right) \tag{26}$$

$$\begin{cases} \left(\tilde{\zeta}^{shp}, \tilde{\zeta}^{rte}\right) = \left(\lambda_{1wa} + \sum_{s} \nu_{sw,k}, \lambda_{1wb} + \sum_{m} \mu_{s,k}\right) \\ \left(\tilde{\beta}^{shp}, \tilde{\beta}^{rte}\right) = \left(\lambda_{2wa} + \sum_{m} w_{mw,k}, \lambda_{2wb} + \sum_{m} \eta_{m,k}\right) \end{cases} \tag{27}$$

6. Related work

6.1. *Service Recommendation*

As described in [1], existing service recommendation approaches mainly based on their functionality [6–9], non-functional features [4, 14], and related with social networks [3, 25, 26]. The functional approaches emphasize the services functionalities during the recommendation, while the non-functional approaches pay more attention to the Quality of the Service (QoS) and the social network-based approaches consider the recommendation from the relationships of the users.

With respect to the functional approaches, the top priority is recommending the service with most suitable functions to users rather than think about index like QoS or Ease of Use. Under this condition some functional approaches use content matching like key-words search [7, 8]. Other approaches use semantic-based search [6, 9, 27] to increase the accuracy of the recommendation. With the development of machine learning, researchers find more approaches for Web recommendation based on probabilistic models (such as Latent Dirichlet Allocation (LDA) [18, 19, 28] or others [29]).

In this paper, we import TCPF model based on PF for recommendation. To the best of our knowledge, we are the first to introduce PF model for service recommendation.

6.2. *Poisson factorization*

Poisson matrix Factorization (PF) originates from non-negative matrix factorization [30], where the objective function is equivalent to a factorized Poisson likelihood [20]. PF is similar to Gamma-Poisson (GaP) model [12]. For a document-term matrix "user-item", GaP places a Gamma prior to the user weights while PF goes further and places Gamma priors to both user and item weights. The most distinctive advantage of PF is that the Variational Bayesian Inference is employed for optimization [24]. By this way PF omits the zero elements in the observed matrices and reduces calculation time greatly.

In Reference [15] it is proved that PF is quite applicable to document-term matrix factorization. But because of good scalability and higher predictive accuracy, PF becomes very

popular in recent years and widespread in multiple areas. Reference [16] uses PF for automatic music tagging. Reference [17] develops CBPF method based on PF for cold-start local event recommendation. All the results of these papers indicate that it is a superior model especially for the content-based data sets. A promising tendency in the development of this model is using it as a basic unit to solve more difficult problems. In this paper we also treat PF as foundation and solve the mashup-side cold-start recommendation problem.

PF model has some changes in application during the evolutionary process. reference [20] develops hierarchical Poisson matrix factorization (HPF) based on PF model. By using PF model at different levels HPF addresses the hierarchical classification problem. Reference [31,32] put forward the model of using PF model in multidimensional matrix and proved Poisson Tensor Factorization. It is certain that PF model will have broader applications as development over time.

7. Conclusion

As the service system is constantly evolving with the rapidly increasing number of published services on the Internet, service recommendation becomes desired when developers are making service compositions.

Many recently proposed studies are based on LDA, however, the restrictive distribution assumptions of LDA put a strict limitation to the performance of modeling mashup queries and service descriptions. In this paper, we improve the TCPF model and propose Separated Time-aware Collaborative Poisson Factorization (TCPF) model to overcome the problem. Poisson Factorization serves as the foundation to model mashup queries and service descriptions separately, and then we incorporate them with the historical usage data by using collective matrix factorization. Experiments on the real-world ProgrammableWeb dataset show that our model outperforms the state-of-the-art methods even in calculation speed and accuracy.

In the future, we plan to take the comments information into consideration for a better performance of recommendation. For we believe the comments of mashups and services will play an increasingly important role in web service recommendation.

Acknowledgements

This research has been partially supported by the National Natural Science Foundation of China (No.61673230). Yushun Fan is the corresponding author.

References

1. Liu, Xumin, and I. Fulia (2015), *Incorporating User, Topic, and Service Related Latent Factors into Web Service Recommendation*, IEEE International Conference on Web Services, pp. 185-192.
2. J. Zhang, W. Tan, J. Alexander, I. Foster and R. Madduri (2011), *Recommend-As-You-Go: A Novel Approach Supporting Services-Oriented Scientific Workflow Reuse*, IEEE International Conference on Services Computing, pp. 48-55.
3. R Kalpana, K. Saruladha and J. Jayabharathy (2016), *Studying the performance of QoS specific web service recommendation system using virtual regions*, J. Web Engineering, Vol.16, pp. 361-396.
4. Q. Yu (2012), *Decision tree learning from incomplete qos to bootstrap service recommendation*, International Conference on Web Services IEEE, pp. 194-201.
5. Z. Zheng, H. MaM. R. Lyu and I. King (2011), *Qos-aware web service recommendation by collaborative filtering*, IEEE Transactions on Services Computing, pp. 140-152.

6. U. Chukmol, A. Benharkat and Y. Amghar (2011), *Bringing socialized semantics into web services based on user-centric collaborative tagging and usage experience*, IEEE Asia-Pacific Services Computing Conference, Jeju, Korea DBLP, pp. 450-455.
7. F. Liu, L. Wang and J. Yu (2012), *Context-aware similarity search of web service*, IEEE International Conference on Information Science and Technology, pp. 596-602.
8. C. Platzer and S. Dustdar (2005), *A Vector Space Search Engine forWeb Services*, European Conference on Web Services IEEE Computer Society, pp. 62.
9. T. Qiu, L. Li and P. Lin (2007), *Web Service Discovery with UDDI Based on Semantic Similarity of Service Properties*, International Conference on Semantics, Knowledge and Grid IEEE Computer Society, pp. 454-457.
10. K. C. Bhardwaj and R. K. Sharma (2015), *Machine learning in efficient and effective web service discovery*, J. Web Engineering Vol.14, pp. 196-214.
11. D. M. Blei, A. Y. Ng and M. I. Jordan (2003), *Latent dirichlet allocation*, Journal of Machine Learning Research, Vol.3, pp. 993-1022.
12. Canny and John (2004), *GaP: a factor model for discrete data*, International ACM SIGIR Conference on Research and Development in Information Retrieval ACM, pp. 122-129.
13. S. Chen, Y. Fan, W. Tan,J. Zhang, B Bai and Z. Gao (2016), *Time-Aware Collaborative Poisson Factorization for Service Recommendation*, IEEE International Conference on Web Services IEEE, pp. 196-203.
14. Y. Jiang, J. Liu, M. Tang and X. Liu (2011), *An effective web service recommendation method based on personalized collaborative filtering*, IEEE International Conference on Web Services IEEE Computer Society, pp. 211-218.
15. P. Gopalan, L. Charlin and D. M. Blei (2014), *Content-based recommendations with Poisson factorization*, Advances in Neural Information Processing Systems, Vol.4, pp. 3176-3184.
16. D. Liang, J. Paisley and DPW Ellis (2014), *Codebook-based Scalable Music Tagging with Poisson Matrix Factorization*, ISMIR, Vol.4, pp. 167-172.
17. W. Zhang and J Wang (2015), *A collective bayesian poisson factorization model for cold-start local event recommendation*, Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1455-1464.
18. Y. Zhong, Y. Fan, K. Huang and W. Tan (2015), *Time-aware service recommendation for mashup creation*, IEEE Transactions on Services Computing, Vol.8, pp. 356-368.
19. B. Bing, Y. Fan, K. Huang and W. Tan, B. Xia and S. Chen (2015), *Service Recommendation for Mashup Creation Based on Time-Aware Collaborative Domain Regression*, IEEE International Conference on Web Services, pp. 209-216.
20. P. Gopalan, J. M. Hofman and D. M. Blei (2014), *Scalable recommendation with poisson factorization*, Computer Science.
21. Y. Koren, R. Bell and C. Volinsky (2009), *Matrix Factorization Techniques for Recommender Systems*, Computer, Vol.42, pp. 30-37.
22. W. Xu, X. Liu and Y. Gong (2003), *Document clustering based on non-negative matrix factorization*, In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pp. 30-37.
23. A. P. Singh and G. J. Gordon (2008), *Matrix Factorization Techniques for Recommender Systems*, Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 650-658.
24. MD Hoffman, DM Blei, C Wang and J Paisley (2013), *Stochastic variational inference*, Journal of Machine Learning Research, Vol.14, pp. 1303-1347.
25. J. Mcdowall and L. Kerschberg (2012), *SLeveraging Social Networks to Improve Service Selection in Workflow Composition*, Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining, pp. 1278-1283.
26. W. Xu, J. Cao, L. Hu and J. Wang (2013), *A social-aware service recommendation approach for mashup creation*, IEEE 20th International Conference on Web Services, pp. 1107-114.
27. S. Kamath and AV S. (2016), *Semantic similarity based context-aware web service discovery using*

nlp techniques, J. Web Engineering, Vol.15, pp. 110-139.
28. K. Huang, J. Han, S. Chen and Z. Feng (2016). *A Skewness-Based Framework for Mobile App Permission Recommendation and Risk Evaluation*, Service-Oriented Computing. Springer International Publishing.
29. B. Jiang, XX Zhang, WF Pan and B. Hu (2013), *BIGSIR: A Bipartite Graph Based Service Recommendation Method*, IEEE Ninth World Congress on Services, pp. 363-369.
30. D. D. Lee and H. S. Seung (1999), *Learning the parts of objects by non-negative matrix factorization*, Nature, pp. 788-791.
31. A. Schein, J. Paisley, D. M. Blei and H. Wallach (2015), *Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts*, Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1045-1054.
32. C. Hu, P Rai, C. Chen, M. Harding and L. Carin (2015), *Scalable bayesian non-negative tensor factorization for massive count data*, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 53-70.