A Novel Approach for API Recommendation in Mashup Development

Chune Li, Richong Zhang, Jinpeng Huai, Hailong Sun
School of Computer Science and Engineering
Beihang University
Beijing, China
Email: {lichune, zhangrc, huaijp, sunhl}@act.buaa.edu.cn

***Abstract*—Mashing up web services and RESTful APIs is a novel programming approach to develop new applications. As the number of available resources is increasing rapidly, to discover potential services or APIs is getting difficult. Therefore, it is vital to relieve mashup developers of the burden of service discovery. In this paper, we propose a probabilistic model to assist mashup creators by recommending a list of APIs that may be used to compose a required mashup given descriptions of the mashup. Specifically, a relational topic model is exploited to characterize the relationship among mashups, APIs and their links. In addition, we incorporate the popularity of APIs to the model and make predictions on the links between mashups and APIs. Moreover, the statistical analysis on a public mashup platform shows the current status of mashup development and the applicability of this study. Experiments on a large service data set confirm the effectiveness of this proposed approach.**

***Keywords*-mashup development; API recommendation; relational topic model.**

I. Introduction

As the growth of service computing in the modern services industry, web services have evolved to web-based "Internet services" [1], which consist of not only classical SOAP/WSDL web services but also RESTful services [2], data, open source codes and any programmable application components accessible through the Internet. On one hand, more and more companies and institutions have provided programmatic access (application programmable interfaces, APIs) to their data and services, the popular ones among which include Google's mapping services, Twitter's microblogging services, Amazon's advertising services and so on. On the other hand, developers are pleased to reuse and mash up the existing data resources and services to create additional value. And in the smart Internet era [3], friendly end user development (EUD) support environments give a boost to the integration process [4]. The applications that combine multiple web services and content from multiple sources are called *mashups*. In this paper, we refer to the generalized services in various protocols and styles as *APIs* and we model a mashup as a composition of APIs. As shown in the real web API directory, ProgrammableWeb.com [5], the number of web APIs and mashups grows sharply recently and up to Jan. 10^{th} 2014, there have been 10737 APIs and 7289 mashups.

As the rapid increase of APIs, finding the required ones is becoming more difficult for mashup developers. Much work on API discovery and recommendation has been proposed to improve the efficiency of mashup development. The common keyword search techniques, as ProgrammableWeb.com adopts, loss the semantics in services, which may be inaccurate. Complex relationships among mashups, APIs and users also have been taken into consideration in different work [6] [7] [8]. These relationships help to discover composition patterns of APIs and imply users' preference in API selection. However, the lessons learned from rough statistics on API usages in mashup develop history are not applicable for new APIs that have not any interactions with other APIs or mashups; while users' latent preference can only serve as supplements to their explicit requests for desired APIs.

Rather than using detailed words matching or coarse-grained services statistics, we choose to model the connections among APIs, mashups and words through topics. The topics are in the form of vectors of words with the probability of occurrence, which are expected to capture the functionality or business semantics to improve the precision of recommendation results. We propose to incorporate the annotation relationship between services and tags and the composition relationship between mashups and APIs to discover the latent topics. Besides, the popularity of APIs is also taken into account to characterize the users' preference towards APIs besides functional consideration. Based on the topic relevances and popularity of APIs, the probability of existing linkage between mashups and APIs is evaluated and related APIs are suggested to build a desired mashup. We collect a service data set from ProgrammableWeb.com and discuss the statistical properties of the current mashups and APIs. At the end, we conduct experiments on this data set and results show the effectiveness of our method in comparison with existing approaches.

The remainder of this paper is organized as follows. We start with the problem definition in section II. Then we introduce a probabilistic model and the topic-based approach to API recommendation in mashup development in section III. In section IV, we discuss the statistical properties of the current mashups and APIs in a real API repository and we report the experimental evaluation of our approach on this API data set in section V. The related work in

978-1-4799-5054-6/14 $31.00 © 2014 IEEE
DOI 10.1109/ICWS.2014.50

API recommendation is discussed in section VI. Finally, we summarize this paper in section VII.

II. Problem Definition

A mashup m consists of a set of APIs, $\mathcal{A}^{(m)} = \{a_1^{(m)}, a_2^{(m)}, ..., a_{|\mathcal{A}^{(m)}|}^{(m)}\}$, where $|\mathcal{A}^{(m)}|$ is the number of APIs that mashup m includes.

At the same time, there is a set of annotations or tags $\mathcal{W}^{(m)} = \{w_1^{(m)}, w_2^{(m)}, ..., w_{|\mathcal{W}^{(m)}|}^{(m)}\}$, describing the functionality of the mashup, while $|\mathcal{W}^{(m)}|$ is the number of the tags of mashup m.

Similarly, each API a has a set of tags, denoted by $\mathcal{W}^{(a)} = \{w_1^{(a)}, w_2^{(a)}, ..., w_{|\mathcal{W}^{(a)}|}^{(a)}\}$, while $|\mathcal{W}^{(a)}|$ is the number of tags of API a.

In the mashup development process, a user specifies a mashup m^* as a set of words, $\mathcal{W}^{(m^*)} = \{w_1^{(m^*)}, w_2^{(m^*)}, ..., w_{|\mathcal{W}^{(m^*)}|}^{(m^*)}\}$, describing the functionality of the mushup to be created. The API recommendation system takes $\mathcal{W}^{(m^*)}$ as a query and returns a list of APIs $\tilde{\mathcal{A}}^{(m^*)} = \{a_1^{(m^*)}, a_2^{(m^*)}, ..., a_{|\tilde{\mathcal{A}}^{(m^*)}|}^{(m^*)}\}$ that may be used to compose the desired mashup. The ideal result is that the recommended API set $\tilde{\mathcal{A}}^{(m^*)}$ contains all the relevant APIs, with no more and no less ones.

Since there are already a large number of mashups and APIs existed in service repositories, we can learn the interaction patterns between mashups and APIs from both the composition relation and tags correlation and then make recommendations based on these interaction patterns. In the following section, we will introduce how to solve the API recommendation problem in details.

III. Approaches

The key point of our approach is to model relationships among mashups, APIs and tags through latent topics, which is achieved through the relational topic model. In this section, the model and a probabilistic approach to API recommendation based on the model are presented.

A. Relational Topic Model

The relational topic model(RTM) [9] [10], first introduced by Jonathan Chang and David M. Blei in the year 2009, is a hierarchical probabilistic model taking into account both the network structure and node attributes. It is an extension of the commonly-used topic model, latent Dirichlet allocation (LDA) [11], which is a generative probabilistic model for collections of discrete data such as text corpora. It views each text document as a set of words and assumes that each document exhibits multiple topics with a statistical probability, while each topic is a distribution over a fixed vocabulary. The RTM model reuses the assumptions behind LDA. However, it considers not only the content of each document, but also the links between documents, such as citation relationships in scientific papers or hyper links of web pages. In this model, a binary random variable is used to connect every two documents. The generative process of all documents and their links in a corpus is as follows:

1) For each topic j in 1:T, draw a distribution over words $\phi_j \sim Dirichlet(\beta)$
2) For each document d in the corpus:
   (a) Draw a vector of topic proportions for the document $\theta^{(d)}|\alpha \sim Dirichlet(\alpha)$
   (b) For each word $w_n^{(d)}$ in the document:
      i. Draw a topic assignment from those topic proportions for this document $z_n^{(d)}|\theta^{(d)} \sim Multinomial(\theta^{(d)})$
      ii. Draw a word from the corresponding topic distribution over words $w_n^{(d)}|z_n^{(d)}, \phi_{1:T} \sim Multinomial(\phi_{z_n^{(d)}})$
3) For each pair of documents $d, d^{'}$, draw a binary link indicator $y_{d,d'}|\mathbf{z}^{(d)}, \mathbf{z}^{(d^{'})} \sim \psi(.|\mathbf{z}^{(d)}, \mathbf{z}^{(d^{'})}, \eta)$.

Denote the number of topics by T and the number of words in the corpus dictionary by W, then ϕ_j is a length W vector denoting the distributions over all words for the j^{th} topic; $\theta^{(d)}$ is a length T vector denoting the proportions over all the topics for document d. The hyper parameters α and β in Dirichlet distribution specify the mean shape and sparsity of the priors on θ and ϕ. $z_n^{(d)}$, a index of topics taking values from 1 to T, is the topic assignment for $w_n^{(d)}$, the nth word of document d; and $\mathbf{z}^{(d)}$ is the vector of topic assignments of all words in document d, $\mathbf{z}^{(d)} = [z_1^{(d)}, z_2^{(d)}, ..., z_{|\mathcal{W}^{(d)}|}^{(d)}]$.

The binary link indicator $y_{d,d'}$ is drawn from a function of the topic assignments of all words from the corresponding document d and $d^{'}$. For document d with $|\mathcal{W}^{(d)}|$ words, let $n_j^{(d)}$ denote the number of words from document d which are assigned to the jth topic. Then the length T vector $\bar{\mathbf{z}}^{(d)} = \frac{1}{|\mathcal{W}^{(d)}|}[n_1^{(d)}, n_2^{(d)}, .., n_T^{(d)}]$ is a statistical representation of topic proportions of document d. In the paper where RTM is introduced, the authors explore four specific possibilities for the link probability function ψ, all of which can be viewed as a function of whether the cosine similarity or the Euclidean distance of $\bar{\mathbf{z}}^{(d)}$ and $\bar{\mathbf{z}}^{(d^{'})}$. In other words, it assumes that documents which have similar topic distributions link to each other more likely. The link probability function in exponential form is defined as follows:

$$\psi(y_{d,d'} = 1) = \exp(\eta^T(\bar{\mathbf{z}}^{(d)} \circ \bar{\mathbf{z}}^{(d^{'})}) + \nu). \quad (1)$$

where the $\circ$ notation denotes the Hadamard product of two vectors, the global regression parameter η controls the importance of links information, and the intercept v forces the function to range in $[0, 1]$. Note that, when $\eta = 0$, the ψ will be a constant such that RTM will be reduced to LDA.

Fig. 1 is the graphical model representation of the RTM that shows the relationships among two documents, their words and latten topics clearly. In this figure, left and right plates represent replications on all words of two documents,

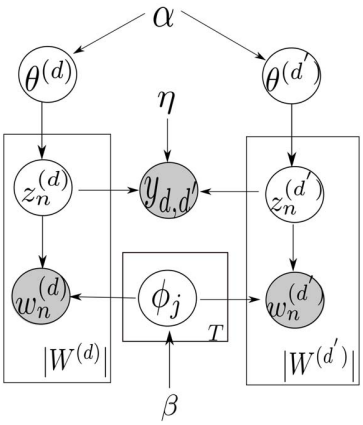

Figure 1. A two-document segment of the RTM.

and the middle plate indicates replications on T topics. Circles are latent variables while the shaded are observed variables.

Given the observations of all the words $\mathbf{w}$ and links $\mathbf{y}$, the posterior distribution of the latent variables $\mathbf{z}, \theta, \phi$ can be approximated by Markove Chain Monte Carlo (MCMC) sampling methods [12] or variational inference [10]. In MCMC, a Morkov chain is created and topic assignment samples are taken from the chain which in turn change the state of the chain. In the RTM model where links are taken into account, the topic assignment of a word then not only depends on the corpus-level topic distributions and topic proportions of the document the word occurs, as the LDA does, but also depends on the topic preference the linked documents have. The update rule is defined as follows.

$$p(z_n^{(d)} = j|\mathbf{z}_{-(n,d)}, \mathbf{w}, \mathbf{y}) \propto \prod_{d'} \exp(\frac{\eta}{|\mathcal{W}^{(d)}|}\bar{\mathbf{z}}_j^{(d')}) \\ p(z_n^{(d)} = j|\mathbf{z}_{-(n,d)}, \mathbf{w}) \tag{2}$$

while

$$p(z_n^{(d)} = j|\mathbf{z}_{-(n,d)}, \mathbf{w}) \propto \frac{\#_j^{(w_n^{(d)})} + \beta}{\#_j^{(\cdot)} + W\beta}(\#_j^{(d)} + \alpha) \tag{3}$$

where $\mathbf{z}_{-(n,d)}$ denotes the vector of all the topic assignments except $z_n^{(d)}$; $d^{'}$ is a document linked with document d. $\#_j^{(\cdot)}$ denotes the number of times words are assigned to topic j; $\#_j^{(w)}$ denotes the number of times word w is assigned to topic j; and $\#_j^{(d)}$ denotes the number of times words from document d are assigned to topic j.

Based on the topic assignments, the topic distributions over words and topic proportions of documents can be calculated by the following equations.

$$\phi_j^{(w)} = \frac{\#_j^{(w)} + \beta}{\#_j^{(\cdot)} + W\beta} \tag{4}$$

$$\theta_j^{(d)} = \frac{\#_j^{(d)} + \alpha}{|\mathcal{W}^{(d)}| + T\alpha} \tag{5}$$

B. Topic-based API Recommendation Incorporating API Popularity

In the API and mashup background, we realize that the invocation relationship between mashups and APIs to some extent forms a document network, where the documents are made up of tags or description words of mashups and APIs, and the link between a mashup and an API means that the mashup consists of the API. Mashup and APIs are relevant through tags and topics, making it possible to make prediction of their links. For a to-be-created mashup without API links, the APIs linked by the predicted links are the recommended results to create the mashup.

Given a number of existing mashups and APIs with words and links, the topic proportions of mashups and APIs and topic assignments of their tags can be discovered. In the API recommendation process, the mashup developer gives a set of words describing the mashup to be created then the RTM model samples the topic assignments z for these words. And then by equation (1), the likelihood this mashup has links with each API is computed. APIs can be sorted in decreasing order according to this probability and the most probable APIs for the new mashup can be generated.

The tags and topic links capture the properties of mashups on the whole. However, we find that practical mashups are biased towards some particular APIs. For example, in the ProgrammableWeb.com, both Google Maps and Microsoft Bing Maps have the same tags: 'mapping', 'places', 'viewer', 'display'; and they are almost published at the same time. But the Google Maps API is used by 2530 mashups while the Microsoft Bing Maps API is used by only 174 mashups. This phenomenon may be resulted from many factors, such as the popularity and reliability of APIs. Taking this into account, we change the link indicator function in the RTM model.

Let $pop(a)$ denotes the popularity of API a, which is defined according to X_a, the number of mashups that consist of this API:

$$pop(a) = \frac{X_a + \lambda}{X_a + 1} \tag{6}$$

Where λ is a smooth efficient ranging in [0,1] which controls the effect of API usage times in history on API popularity. When λ equals 0, the APIs which have not been used by any mashups will be deeply distrusted by new mashups. When λ equals 1, all APIs have the same popularity. Figure 2 shows the popularity function when $\lambda = 0.1$.

Then the link function reinforced with popularity is defined as follows:

$$\psi(y_{ma} = 1) = \exp(\eta^T pop(a)(\bar{\mathbf{z}}^{(m)} \circ \bar{\mathbf{z}}^{(a)}) + \nu). \tag{7}$$

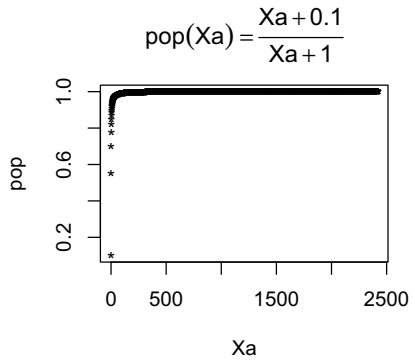

Figure 2. Popularity function when $\lambda = 0.1$.

The update rule for the enhanced RTM model is as follows:

$$p(z_n^{(m)} = j|\mathbf{z}_{-(n,m)}, \mathbf{w}, \mathbf{y}, \mathbf{X}) \propto \prod_a \exp(\frac{X_a + \lambda}{X_a + 1} \frac{\eta}{|\mathcal{W}^{(m)}|} \bar{\mathbf{z}}_j^{(a)}) p(z_n^{(m)} = j|\boldsymbol{z}_{-(n,m)}, \mathbf{w}) \quad (8)$$

RTM models the relevance between mashups and APIs in terms of functionality and domain semantics via topics, which is one of the most important factors that dominate the selection of APIs in mashup creation. Compared to simple keyword search techniques, topic based model has the advantage of capturing the latent correlations of words which improves the matching between the requirement of mashups and the functionality of existing APIs. It ensures the precision of API selection. By taking the popularity of APIs into account, our enhanced RTM captures another factor which also affects the selection of APIs greatly in practise. Besides, popular ones are usually with higher quality. In this way, we try to recommend APIs which are both relevant and with high quality.

IV. Statistical Analysis on a Real Data Set

We crawled the ProgrammableWeb.com in November 2012 and collected a data set with 7974 APIs, 6815 mashups and 14127 links between APIs and mashups.

Statistics on the usage of APIs show that most APIs (up to 85.6 percentages) have not been used by any mashups. This indicates that APIs still have great potential to compose new mashups. To make full use of these APIs to create additional value, recommendation approaches suitable for never used new APIs will become promising. Word tags are an important properties for new APIs and we argue that the latent topics found from existed mashups are helpful for the discovery of new APIs.

At the other extreme, the most popular API, Google Maps, is used by 2416 mashups in this data set. It is predictable that the popular ones will probably be used in the future.

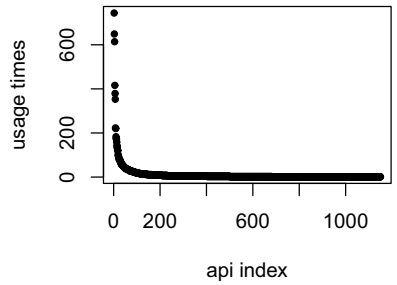

Figure 3. The times each API is used by mashups. (The most frequently used API and those APIs that have never been used are omitted in this figure.)

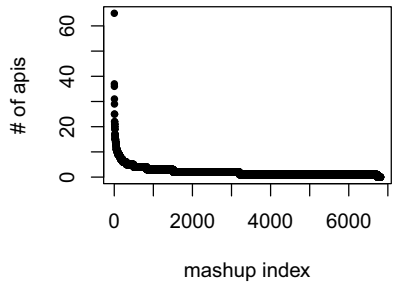

Figure 4. The number of APIs each mashup consists of.

And on average, each API is used by 1.772 mashups. Figure 3 shows the distribution details. Each point in the curve is one API and the vertical axis represents the number of mashups that consist of this API. Note that we draw API points in the descend order of usage times and APIs that have never be used and the Google Maps API are not shown in the figure. It can be seen that the curve follows a negative exponential distribution.

On the other hand, we count the number of APIs each mashup consists of. More than half of the mashups (51.8 percentages) have only one API while the average number of APIs contained by mashups is 2.073. Figure 4 shows the details of this analysis. Each point in the curve is one mashup and the vertical axis represents the number of APIs it makes up of. This curve also follows a negative exponential distribution.

Another factor we focus on is the tags of mashups and APIs. In this data set, each API and mashup is attached with one to nine tags, see Figure 5. (We have filtered out ten services without any tags in data preparation, since they

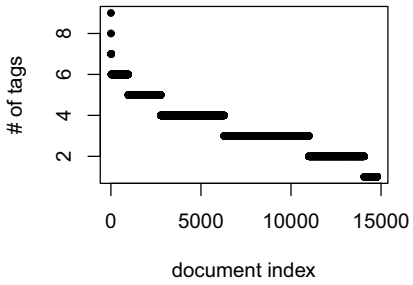

Figure 5. The number of tags each API or mashup consists of.

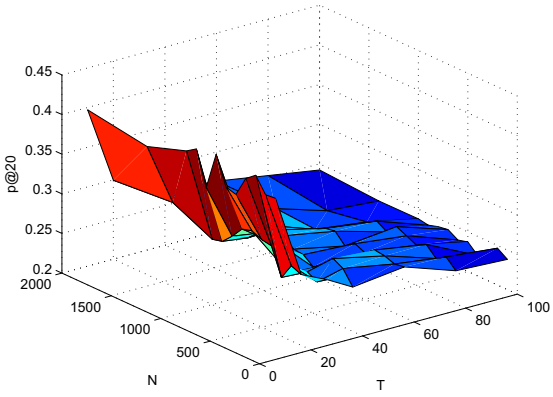

Figure 6. Precision at top 20 results (p@20) changes over topic numbers (T) and iteration times (N) for the enhanced RTM model.

can be regarded as incomplete.) The top 10 popular tags are 'mapping', 'deadpool', 'social', 'search', 'shopping', 'photo', 'video', 'mobile', 'reference', 'tools'. Most tags describe the functionality for both mashups and APIs.

V. Experiments

In this section, we verify our topic-based approaches to API recommendation on the data set described in section IV.

A. Model Training

Our model is fitted based on the *lda* package[1], an implementation of collapsed Gibbs sampling methods for topic models in R language.

To choose befitting parameters for our enhanced RTM model, 84 groups of experiments are done, where the topic number T ranges from 10 to 100, and sampling iteration times N ranges from 100 to 2000. Other parameters are set as constants in all groups: $\alpha = 0.1, \beta = 0.1, \eta = 3, \nu = -3, \lambda = 0.1$.

In each group, a ten-fold cross validation on API recommendation is used. In each fold, we get a test set by sampling 10 percentages of mashups and removing their API links. These APIs that a test mashup is linked to serve as the ground truth of recommend results for this mashup. The rest mashups and all APIs are training data. Then we iteratively sample topic assignments z according to equation (8) for words in both training set and test set until each variable reaches a stable state. Finally, for each test mashup, we rank all the APIs according to the probability of each API linking to the mashup calculated by equation (7). The recommendation results are measured by $p@20$.

- $p@20$: the precision of top-20 results. That is the ratio of right APIs (whose links to the test mashup are removed before sampling topic assignments) in top 20 recommended results to the number of all right APIs.

[1]http://cran.r-project.org/web/packages/lda/

Table I
Topic examples got by enhanced RTM

TOPIC 1		TOPIC 5		TOPIC 7	
mobile	0.1466	social	0.1558	video	0.1243
telephony	0.1466	deadpool	0.0788	music	0.1072
messaging	0.1242	search	0.0670	photo	0.0973
sms	0.0831	microblogging	0.0516	deadpool	0.0591
voice	0.0464	news	0.0451	search	0.0486
deadpool	0.0442	photo	0.0354	social	0.0466
social	0.0416	twitter	0.0309	movies	0.0213
global	0.0222	bookmarks	0.0299	games	0.0184

For each recommendation for a test mashup, we calculate the $p@20$ value. Average the value on all the mashups in a fold, we get the result of the fold. And then average the 10 folds results, we get the measurement of one group. Figure 6 shows the results of 84 groups of parameter settings. It turns out that $T = 10$ and $N = 700$ is the best permutation of parameters.

B. Topic Results

After getting the values of the topic number and iteration times, we run our enhanced RTM model on all the APIs and mashups. Then according to equation (4), topics are got. Table I shows three discovered topics and corresponding words as an example. Top 8 most probable words of each topic are listed. By analyzing the words in these topics, we can see TOPIC 1 is about mobile communication, mainly via sms and voice; TOPIC 5 is about social networking. The main media are blogs, news and photos while the main interactive operations are search and tagging. Besides, since the 'deadpool' is an important tag in this topic, we can guess this domain changes quickly and many social services are unavailable now. TOPIC 7 is on multimedia, including photos, videos, music and movies. The main operations on these media are search or interaction in social network. We should say the results are quite heuristic and reasonable.

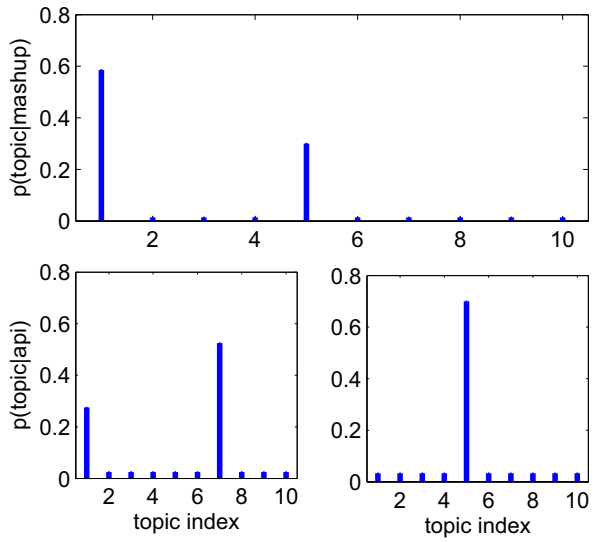

Figure 7. Topic proportions of one mashup (top) and its two APIs(bottom).

According to equation (5), we give the topic proportions of one mashup and its two APIs in Figure 7. In each subgraph, the horizontal axis represents 10 topics while the heights of the bars represent the proportions of the corresponding topics in the mashup or API. The mashup is VoiceMap, which records location voice notes on the cell phone and plays them on the web or publishes them on Twitter. It consists of two APIs: WHERE GPS and Twitter. As showed in the distribution histograms, The main topics of VoiceMap are mobile communication and social services, while besides these two topics, its APIs also include TOPIC 7. This example shows the effectiveness of topics further.

C. Approaches Comparison

In this section, we compare our model with several different approaches in three aspects.

1) The effect of linkage.
   Mashups and their APIs are related to each other in functionality and modeling the links helps to discover the latent functionalities and then improves the prediction results. To verify this idea, we compare the original RTM model and the LDA model.
   - RTM: The original RTM model, that is to train the model with equation (2) and make prediction with equation (1).
   - RTM2LDA: The RTM model with link parameter $\eta = 0$. This setting reduces the RTM model to LDA which does not consider the links.
2) The effect of popularity of APIs.
   The reinforced RTM and the original RTM model are compared to show the advantage of incorporating the API popularity in API recommendation.
   - ERTM: Our enhanced RTM model considering the popularity of APIs, that is to train the model by equation (8) and make prediction by equation (7).
   - ERTM2RTM: The ERTM model with popularity parameter $\lambda = 1$. This setting reduces the ERTM model to RTM.
3) Topic-based approaches v.s. word-based approaches.
   Besides these topic-based approaches, we also exam other ranking methods according to words directly (rather than fitting the model and then ranking according to the similarities got from latent topics).
   - WVSM: The vector space model represents a mashup or an API as a length-W binary vector $\mathbf{v}$ where each component $\mathbf{v}_n$ indicates whether the nth word of the corpus dictionary occurs in the mashup or API or not. Then WVSM ranks APIs according to the cosine similarity of the corresponding two binary vectors multiplying the popularity of the API as a weight.

$$p_{wvsm}(m,a) = pop(a)\frac{\mathbf{v}^{(m)}\mathbf{v}^{(a)}}{||\mathbf{v}^{(m)}||||\mathbf{v}^{(a)}||} \quad (9)$$

   - WJaccard: The Jaccard similarity is defined as the size of the intersection divided by the size of the union of the word sets of a mashup and an API. The popularity of the API is also multiplied to the ratio.

$$p_{wvsm}(m,a) = pop(a)\frac{|\mathcal{W}^{(m)} \bigcap \mathcal{W}^{(a)}|}{|\mathcal{W}^{(m)} \bigcup \mathcal{W}^{(a)}|} \quad (10)$$

Unless stated, all topic based models set the same parameters as those used in parameter selection process. And the 10-fold cross validation is used in all the comparison experiments. However, besides the precision of top-20 results p@20, the results are also evaluated by other two measurements:

- RKL: rank of the last matching case. In other words, it is the value of the rank at 100%-recall. Since in the mashup development, we hope to get all the APIs needed from the top results, the lower the value of RKL is, the better the recommendation result is.
- AP: average precision. For a mashup m and the recommended API list $\boldsymbol{L}$, the AP of $\boldsymbol{L}$ for m is defined by the following equation.

$$AP(m,\boldsymbol{L}) = \frac{1}{|\mathcal{A}^{(m)}|}\sum_{i=1}^{RKL(m,\boldsymbol{L})}\frac{is(\boldsymbol{L}_i,m)\#(\boldsymbol{L}_{1:i},m)}{i} \quad (11)$$

where $|\mathcal{A}^m|$ is the number of APIs of mashup m, $RKL(m,\boldsymbol{L})$ is the RKL value of $\boldsymbol{L}$ for mashup m.

$is(L_i, m)$ is an indicator function which values 1 if API L_i is included by m actually, otherwise values 0. $\#(\boldsymbol{L}_{1:i}, m)$ is the number of right APIs in top-i recommended APIs. The mean average precision of all mashups in all ten folds is higher, the recommendation is better.

Table II shows the results of all the six different approaches. The comparison of the RTM model and the RTM2LDA approach verifies the benefits of taking into account the invocation relationships among mashups and APIs. The precision at top 20 results, for example, are improved by 30% in the RTM model. However, the value is limited. It is considering the popularity of APIs that improves the performance of API recommendation greatly, showed by the comparison of ERTM and ERTM2RTM. The keyword-based approaches with popularity weights also have good results. However, the ERTM model is better than WVSM and WJaccard in all the three measurements, which proves the strengths of topic-based matching approaches over word-based approaches.

In the collected data set, mashups and APIs are represented as tags, which are mostly unambiguous words. In other situations where mashups and APIs are described by sentences, we believe that the advantages of topic-based approaches will be more obvious.

VI. Related Work

Much work related to mashup construction and Web API recommendation has been done in these years. MashupAdvisor [13] is an example of the first group which complement a partially constructed mashup automatically. Given a current partially constructed mashup, it first suggests possible outputs; when the user selects a recommended output, the system then outputs the best plan(composition of services) that will output the selected output. In this work, each service has a formal input and an actual output, both of which are described as concepts(words). In the recommended plan, one service's output is binded to another service's formal input, with an AI planner maximizing the total semantic similarities and statistic probabilities of these bindings. A similar work is done in [14], where both services' inputs/outputs and the user's goals are described by tags(words) and the system recommends a sequence of services while each service's outputs contain the next service's inputs and the outputs of all services contain the user's goals. These studies aim to plan composition solutions automatically. One main concern is the computation complexity owing to the large word space. Besides, to recommend a structured mashup, these plans require explicit information about the input and output of services or services' operations, which is not always available. Such as the case of ProgrammableWeb.com, most services are RESTful APIs, which have no standards to guide the description of their inputs and outputs, and tagging them is a tedious work.

With the complete information of inputs and outputs of APIs and also the structure information of the existing mashups, the work of [15] mines re-usable composition patterns from the existing mashups and recommends composition knowledge by pattern matching and retrieval.

As for single API recommendation, the work of R. Torres et. al. [7] is similar to ours. They propose to combine the API popularity to traditional keyword search techniques to retrieve required APIs. The popularity of an API is measured by the degree of the API node in an API collaboration network, which is an alternative to a positive correlation function of the degree of an API node in an API-mashup bipartite graph as used in our approach. In the keyword search process, they represent a user's query and APIs as documents in a VSM model. With a hierarchical classification, a cluster of APIs closest to the query is found using a cosine similarity. Then the APIs in the cluster are finally ranked according to both the APIs' keyword similarity with the query and APIs' popularity. As we have showed in the comparison between ERTM and WVSM, the topic-based matching approach has advantages than the keyword based ranking according to cosine similarity in vector space model. Another difference of our work is that, aiming to recommend services to compose a mashup, our work also try to model semantic relevance between APIs and mashups.

Semantic approaches for API discovery have also be studied. [16] is a representative of these works, which combines WADL and a learning ontology mechanism to describe RESTful services. These semantic approaches are expected to provide high performance in discovering similar services. The disadvantages lie in the absence of standards and the shortage of semantic description for most existing APIs.

The work of [8] suggests to leverage the social network of users and services to a step-by-step mashup completion. It captures the similarity between users and recommends services based on similar users. Buqing Cao et.al. [6] also focuses on personal recommendations. They model a user's interest as a vector of term weights based on the description documents of mashups the user has used in history. Given a mashup, other mashups similar to this one and described by words similar with the user's interest are recommended. In a service repository with enough users information and history mashup usage records, these approaches can supplement the recommendation totally based on services.

There are several mashup editors with friendly graphical user interface which support mashup develop for non-programmer end users. Grammel Lars and Storey Margaret-Anne [17] present a survey of six mashup development environments. In most of the tools, users must select and customize data sources and API operators to build the desired application. With the increasing number of APIs, automatic API recommendation and mashup formation techniques including our topic based approaches will place an

Table II
APPROACH COMPARISIONS

Approaches	RTM	RTM2LDA	ERTM	ERTM2RTM	WVSM	WJaccard
p@20	0.0072	0.0055	0.4120	0.0068	0.3842	0.3495
RKL	2303.4	3102.4	355.6	2256.6	1614.5	1616.5
AP	0.0073	0.0048	0.1620	0.0069	0.1569	0.1449

important role in these mashup develop environments.

VII. CONCLUSION

This study focuses on the web API recommendation in the process of mashup building to ease the burden of discovery of required data and service resources in the vast services repository. The relational topic model is used to capture the relationship among tags, mashups and APIs. Latent topics are discovered to model the semantics and domains of services. In addition, we take the popularity of APIs into account to make the model characterize the preference to popular APIs more precisely. Based on the topic relevance and API popularity, the model predicts links between mashups and APIs. Statistical analysis on a public mashup platform shows the great potential of APIs to create novel mashups. Experiments on a real mashup data set verify the effectiveness of our approaches.

ACKNOWLEDGMENT

This work was supported partly by National Natural Science Foundation of China (No. 61300070, No. 61103031), partly by China 863 program (No. 2012AA011203, No. 2013AA01A213), China 973 program (No.2014CB340300, No. 2014CB340304), partly by the State Key Lab for Software Development Environment (SKLSDE-2013ZX-16), partly by A Foundation for the Author of National Excellent Doctoral Dissertation of PR China(No. 201159), partly by Beijing Nova Program(2011022) and partly by Program for New Century Excellent Talents in University.

REFERENCES

[1] C. Pedrinaci and J. Domingue, "Web services are dead. long live internet services," *SOA4All White Paper*, vol. 8, 2010.

[2] L. Richardson and S. Ruby, *RESTful web services*. O'Reilly, 2008.

[3] M. Chignell, J. Cordy, J. Ng, and Y. Yesha, *The smart internet: current research and future applications*. Springer, 2010, vol. 6400.

[4] D. Benslimane, S. Dustdar, and A. Sheth, "Services mashups: The new generation of web applications," *Internet Computing, IEEE*, vol. 12, no. 5, pp. 13–15, 2008.

[5] "Mashup timeline," http://www.programmableweb.com/, Jan. 2014.

[6] B. Cao, J. Liu, M. Tang, Z. Zheng, and G. Wang, "Mashup service recommendation based on user interest and social network," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 99–106.

[7] R. Torres, B. Tapia, and H. Astudillo, "Improving web api discovery by leveraging social information," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 744–745.

[8] A. Maaradji, H. Hacid, R. Skraba, A. Lateef, J. Daigremont, and N. Crespi, "Social-based web services discovery and composition for step-by-step mashup completion," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 700–701.

[9] J. Chang and D. M. Blei, "Relational topic models for document networks," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 81–88.

[10] ——, "Hierarchical relational models for document networks," *The Annals of Applied Statistics*, vol. 4, no. 1, pp. 124–150, 2010.

[11] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[12] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," *Machine learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[13] H. Elmeleegy, A. Ivan, R. Akkiraju, and R. Goodwin, "Mashup advisor: A recommendation tool for mashup development," in *Web Services, 2008. ICWS'08. IEEE International Conference on*. IEEE, 2008, pp. 337–344.

[14] X. Liu, Q. Zhao, G. Huang, H. Mei, and T. Teng, "Composing data-driven service mashups with tag-based semantic annotations," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 243–250.

[15] S. R. Chowdhury, F. Daniel, and F. Casati, "Efficient, interactive recommendation of mashup composition knowledge," in *Service-Oriented Computing*. Springer, 2011, pp. 374–388.

[16] Y.-J. Lee and C.-S. Kim, "A learning ontology method for restful semantic web services," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 251–258.

[17] L. Grammel and M.-A. Storey, "A survey of mashup development environments," in *The smart internet*. Springer, 2010, pp. 137–151.