Journal of Web Engineering, Vol. 16, No.7&8 (2017) 676-692


# A HYBRID APPROACH FOR AUTOMATIC MASHUP TAG RECOMMENDATION

MIN SHI JIANXUN LIU DONG ZHOU

*Department of Computer Science and Engineering*

*Hunan University of Science and Technology*

*{toshimin132, ljx529}@gmail.com, dongzhou1979@hotmail.com*

Received May 25, 2016

Revised May 21, 2017

Tags have been extensively utilized to annotate Web services, which is beneficial to the management, classification and retrieval of Web service data. In the past, a plenty of work have been done on tag recommendation for Web services and their compositions (e.g. mashups). Most of them mainly exploit tag service matrix and textual content of Web services. In the real world, multiple relationships could be mined from the tagging systems, such as composition relationships between mashups and Application Programming Interfaces (APIs), and co-occurrence relationships between APIs. These auxiliary information could be utilized to enhance the current tag recommendation approaches, especially when the tag service matrix is sparse and in the absence of textual content of Web services. In this paper, we propose a hybrid approach for mashup tag recommendation. Our hybrid approach consists of two continuous processes: APIs selection and tags ranking. We first select the most important APIs of a new mashup based on a probabilistic topic model and a weighted PageRank algorithm. The topic model simultaneously incorporates the composition relationships between mashups and APIs as well as the annotation relationships between APIs and tags to elicit the latent topic information. Then, tags of chosen important APIs are recommended to this mashup. In this process, a tag filtering algorithm has been employed to further select the most relevant and prevalent tags. The experimental results on a real world dataset prove that our approach outperforms several state-of-the-art methods.

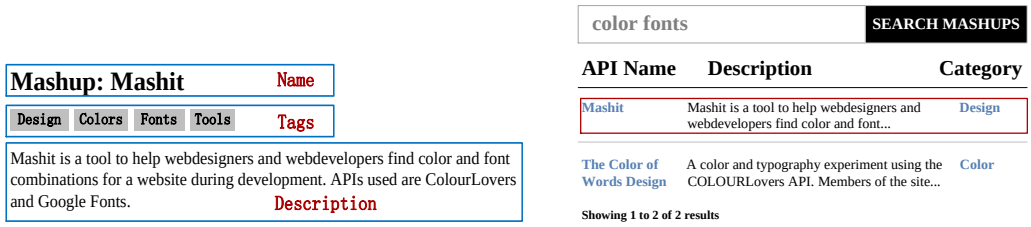
*Key words*: tags, mashups, APIs, tag recommendation, topic model, PageRank
*Communicated by*: M. Gaedke & Q. Li

## 1 Introduction

With the advent and development of Web 2.0, tag has been a research hotspot for its many merits, especially in the tasks of resource management, classification and retrieval [1-5]. Nowadays, many

---

This work is an extension of our previous study on mashup tag recommendation [15].
Jianxun Liu is the corresponding author (e-mail: ljx529@gmail.com).




(a) (b)

Figure 1. (a) An example mashup on ProgrammableWeb. (b) Search results with keywords “Color” and “Fonts”.

tagging systems such as ProgrammableWeb[1] and Biocatalogue[2] allow users to use several arbitrary and personalized words to annotate the mashups and APIs (called the behaviour of tagging). Tagging is the process of associating several meaningful and relevant keywords to a resource, e.g., the mashup and image. In most cases, these tags can reveal the semantic information the description documents[3] reflect. In the past few years, with the rapid development and growth of Web services and mashups, tags have been widely utilized to enhance the management of enormous Web service data. In addition, tags are also helpful for users to discover their desired resources. For example, Figure 1 (a) shows an example mashup on ProgrammableWeb[4]. The specific mashup “Mashit”, together with its textual description, have been annotated by the developer with four different tags. To better illustrate the usefulness of tags from facilitating the mashup resource retrieval aspect, we chose two tags “Color” and “Fonts” from the tag set as keywords to search the entire mashup repository on ProgrammableWeb. Figure 1 (b) demonstrates the search results. As it shows, the target mashup “Mashit” appears as the top relevant result.

However, people tend to create several relevant tags manually while registering a new mashup. In most cases, constructing a useful and meaningful tag set for a mashup is not an easy task. In addition, tags in the tagging system sometimes suffer from the lexical gap problem. For example, the new tag "firm" may corresponds to the already exited tag "company", which poses a challenge for the search engine to return all relevant results using only single searching keyword ("firm" or "company"). Therefore, automatic approaches are desired to ease the tagging process as well as regulate the tag vocabulary used to annotate a new mashup. As a result, majority of tag recommendation systems function with automatic tagging approaches [3][6-8]. Several of them have achieved promising performance, such as clustering-based methods [6][9] and co-occurrence based methods [10-11]. Since tags can be very emotional and subjective [4], it is crucial to mine the latent topic information for tag recommendation [12-13]. Therefore topic model-based methods have been widely adopted [2][14]. However, most tag recommendation approaches only exploit the tag service matrix and the textual description information of Web service [7][14]. In fact, rich relationships could be mined from the Web services data, such as composition relationships between mashups and Application Programming Interfaces (APIs), as well as annotation relationships between tags and APIs. These auxiliary

[1] http://www.programmableweb.com/category/all/mashups.

[2] http://www.biocatalogue.org/.

[3] description document: the textual description the author created for a resource, such as the description documents for mashups.

[4] http://www.programmableweb.com/mashup/Mashit.

information can be fully utilized to further improve the performance. We propose a hybrid approach for automatic mashup tag recommendation which exploits both the textual contents and the above discussed relationships. Our hybrid approach consists of two continuous processes: APIs selection and tags ranking. We first select the most similar APIs of a new mashup based on a novel probabilistic topic model. We then combine these chosen similar APIs and member APIs of mashup into a single API set. Subsequently, we select several important APIs (those possibly contain the relevant tags of the target mashup) from this API set based on a weighted PageRank algorithm. Finally, tags of chosen APIs are recommended to this mashup. To control the number of recommended tags, a tag filtering algorithm has been designed to select the most relevant and popular tags.

This work is an extension of our previous study on mashup tag recommendation [15]. The main difference between the current and the previous one is that we take member APIs of a mashup into account when finding the candidate recommended tags. More specifically, on some occasions, the relevant tags of a mashup are only included in its member APIs. Our previous work [15] only recommends tags extracted from the similar APIs (which sometimes contain few member APIs) of a mashup. We address this problem by combining similar APIs of a mashup obtained based on a topic model and member APIs of the current mashup. To further choose the most important APIs, we construct a Web service graph using various co-occurrence relationships (e.g. two APIs share identical tags). Then, a weighted PageRank algorithm is utilized to calculate the importance of each mashup.

The reminder of this paper is organized as follows. Section 2 presents the current practice and research on tag recommendation. Section 3 describes our approach in detail. In Section 4, we describe the experimental settings and analyze results obtained. Finally, Section 5 concludes the paper.

## 2 Current Practice and Research

In the era of Web 2.0, tag has been used as a common and efficient tool to organize and index Web resources. A significant number of tag recommendation approaches have been proposed, especially in the social tagging systems [1][16-19]. In general, those techniques can be classified into three categories [20]: content-based methods; co-occurrence based methods; and hybrid methods. Content-based methods exploit the textual content of documents for tag recommendation [16][21]. This group of methods ignore the semantic ambiguity of tags, hence the recommendation results are somewhat unsatisfied [1][12]. Co-occurrence based methods employ tag co-occurrences to expand tag set of documents [10][21]. However, these methods only use the co-occurrence data, hence may exist the problem of topic drift [17]. Hybrid methods consider both co-occurrence statistics and content information for tag recommendation [13]. In hybrid methods, learning latent topics of items is vital to improve the recommendation accuracy [13]. Therefore, in the past, topic model-based methods have been widely utilized [2-3][14]. For example, Krestel et al. propose an approach for tag recommendation based on the LDA model [14]. They expand tag set of resources with tags whose topic assignments are same with the existed tags. Si et al. propose a Tag-LDA model for tag recommendation. The model extends the standard LDA by adding a tag variable in the generative process [2]. Although Si et al.'s model has some advantages than the basic LDA model, it does not consider the composition relationships between mashups and APIs. Chang et al. develop a RTM model [22-23] which also extends the LDA model, their model considers not only the content of each resource, but also the links between resources, such as the citation relationships in scientific papers or

hyperlinks of web pages. However, annotation relationships between tags and APIs are not considered when utilizing RTM directly for tag recommendation.

Tag recommendation for Web services is also well studied [3][6-8]. In particular, clustering-based methods are widely utilized [6][8-9]. For example, Lu et al. propose an automatic Web service tagging approach consisting of two strategies: tag enriching and tag extraction. The authors first cluster Web services using WSDL documents, and then recommend tags for a service extracted from tags of other services in the same cluster [6]. Unfortunately, methods based on WSDL matching suffer from the vocabulary problem [24] and the intention gap [25]. Different machine learning techniques have also been adopted for automatic Web services tagging [3][8]. For example, Aznag et al. propose a topic model based approach to automatically tag Web services according to the existing manual tags [3]. They first select candidate tags from Web services based on a topic model. Those tags are then used to train a classifier. Based on the classifier, they finally recommend the best tags for new Web services. However, most tag recommendation approaches utilizing probabilistic topic models [3][14] are purely based on words of documents. However, in the Web service tagging systems, there are rich relationships such as composition relationships between mashups and APIs as well as the annotation relationships between tags and APIs. This auxiliary information can be fully utilized to further promote the recommendation accuracy.

## 3 Automatic Tag Recommendation for Mashup

In this section, we firstly introduce our motivation and define the problem. Then we briefly describe the probabilistic topic model proposed and the weighted PageRank algorithm used in this paper. Lastly, we present our hybrid approach for tag recommendation in detail.

### *3.1. Motivation*

To motivate our work, we begin the discussion by exploiting the relationships among mashups, Web APIs and tags. Figure 2 visualizes example relationships. Composition relationship (dashed arrow) indicates the API-to-mashup relationship, representing that a mashup is composed by a number of APIs (member APIs) it points to. Annotation relationship (solid line) denotes the tag-to-API and tag-to-mashup relationships, representing that a mashup or an API is annotated by various tags. As can be seen from the figure, some mashups and APIs are annotated by some identical tags. It is also shown in the figure that in most cases, if a mashup and an API share identical tags, there is usually a composition relationship between them. It has been previously shown that if mashups and APIs have composition relationships between them, they normally have similar functionalities, thus have similar function description documents [26]. Based on the observation, our procedure to recommend tags for a newly created mashup works as follows, we firstly gather APIs whose description documents are similar to that of a target mashup. But these similar APIs may contain only few member APIs of this mashup. Based on the observation (see the case study in Section 4), some relevant tags of a mashup are only included in tags of its member APIs. Therefore, those chosen similar APIs and member APIs of this mashup are combined into a single API set. Finally, we recommend tags extracted from this API set to the target mashup. Taking the mashup named “Where Aml.At” in Figure 2 as an example, we first select API named “Shooping.com” as a similar API, then this API together with another two member APIs named “Google maps” and “Flickr” are combined into a single API set. We finally recommend several relevant tags extracted from this API set.

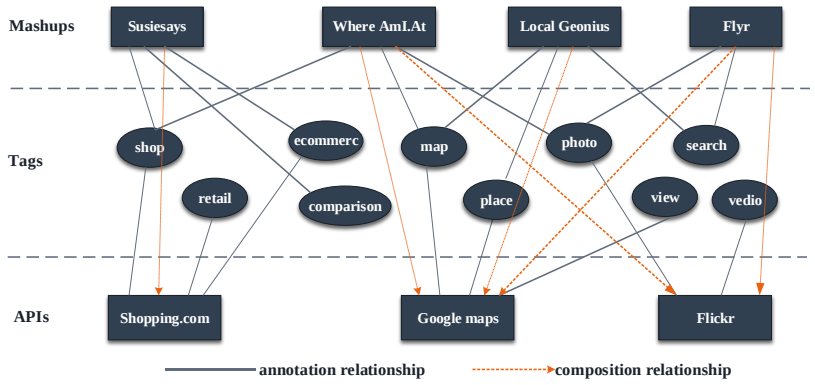



Figure 2. Relationships among Mashups, Tags and APIs.

According to the statistics (in section 4) on tag distribution of mashups, 99% of the mashups have been associated with less than 6 tags, which means we should further select top *n* most important APIs from the above combined API set to fetch their tags to recommend. As a result, we first construct a directed graph leveraging all the APIs in the dataset, where we exploit various co-occurrence relationships in Figure 2, such as the co-occurrence relationship between two member APIs of a mashup. Then we choose the top *n* most important APIs based on a weighted PageRank Algorithm.

### *3.2. Problem Definition*

Formally, a description document of a mashup $m$ is represented as $W^{(m)} = \{w_1, w_2, \ldots, w_{|W^{(m)}|}\}$, where $w_i$ is the *i*th word of the document, and $|W^{(m)}|$ is the number of words the document contains. Tags annotated with a mashup *m* are defined as $T^{(m)} = \{t_1, t_2, \ldots, t_{|T^{(m)}|}\}$, where $t_i$ is the *i*th tag of the mashup, and $|T^{(m)}|$ is the number of tags. Analogously, a description document of an API $a$ is denoted as $W^{(a)} = \{w_1, w_2, \ldots, w_{|W^{(a)}|}\}$, where $w_i$ is the *i*th word of the document, and $|W^{(a)}|$ is the number of words the document contains. There is also a set of tags attached to an API $a$, denoted by $T^{(a)} = \{t_1, t_2, \ldots, t_{|T^{(a)}|}\}$, where $t_i$ is the *i*th tag of the API, and $|T^{(a)}|$ is the number of tags. For a newly developed mashup $m'$ with a description document $W^{(m')}$, our system aims to automatically recommend adequate relevant tags to this mashup.

### *3.3. Probabilistic Topic Model and PageRank Algorithm*

We first describe the Mashup-API-Tag model (MAT). Then we give a brief introduction of the PageRank algorithm used for APIs selection task in this paper. The model is a combination of Tag-LDA[2] and Relation Topic Model (RTM)[22]. The reason why we propose the MAT model is that RTM can model annotation relationships, but it cannot capture semantics of tags. On the other hand, Tag-LDA can model annotation relationships, but it neglects the composition relationships. MAT incorporates both the composition relationships and annotation relationships to elicit the latent topic information, which is best suitable for our mashup tag recommendation task.

The plate notation of MAT model is shown in Figure 3. We denote the number of topics by *T*, the number of words in corpus dictionary of mashups and APIs by *N*, and the number of all the tags of

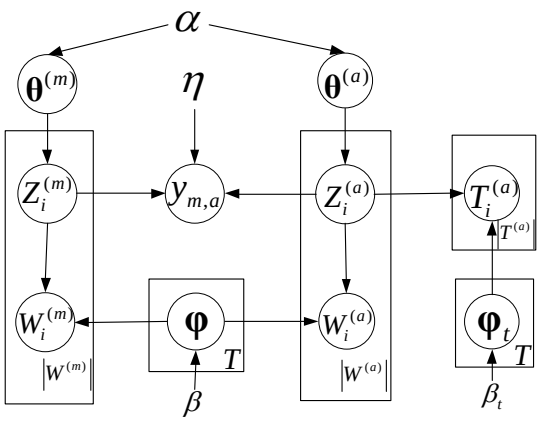



Figure 3. Plate Notation of Our Novel Probabilistic Topic Model.

APIs by $N_t$. $\boldsymbol{\theta}^{(m)}$ and $\boldsymbol{\theta}^{(a)}$ are length $T$ vectors denoting the proportions over all the topics for description documents $W^{(m)}$ and $W^{(a)}$; $\boldsymbol{\varphi}$ is a length $N$ vector denoting the distribution over all words; $\boldsymbol{\varphi}_t$ is a length $N_t$ vector denoting the distribution over all tags. $y_{\mathrm{m,a}}$ is an observed variable representing the composition probability or semantic similarity between a mashup and an API. The model contains the variable $y_{\mathrm{m,a}}$ for each pair of description documents for mashups and APIs, generated on the topic assignments for the participating documents $W^{(a)}$ and $W^{(m)}$. We model the description documents for mashups, APIs and tags of APIs in a more unified way to obtain the embedding representations of each part. Thus we can calculate the similarity between mashups and APIs according to their topic distribution vectors. In addition, we can also capture semantics of tags in the form of topic distribution, which is helpful for the tag filtering algorithm in this paper. The generative process of MAT is summarized in Table 1.

Table 1 The generative process of MAT

1. For each topic that the description document of mashup or API contains, draw a distribution over words $\boldsymbol{\varphi} \sim Dirichlet(\beta)$.
2. For each topic that the tags of API contain, draw a distribution over tags $\boldsymbol{\varphi}_t \sim Dirichlet(\beta_t)$.
3. For each description document for mashup, draw a vector of topic proportion for the document $\boldsymbol{\theta}^{(m)} | \alpha \sim Dirichlet(\alpha)$.
4. For each description document for API and tags of this API, conduct the same process as step 3.
5. For each word $w_i^{(m)}$ in the description document for mashup:
   Conditional on $\boldsymbol{\theta}^{(m)}$ choose a topic $z_i^{(m)} | \boldsymbol{\theta}^{(m)} \sim Multinomial(\boldsymbol{\theta}^{(m)})$;
   Conditional on $z_i^{(m)}$ choose a word $w_i^{(m)} | z_i^{(m)}, \boldsymbol{\varphi}_{1:T} \sim Multinomial(\boldsymbol{\varphi}_{z_i^{(m)}})$.
6. For each word $w_i^{(a)}$ in the description document for API and each tag $T_i^{(a)}$ in tag set of this API, conduct the same process as step 5.

Taking all the words of description documents for mashups and APIs, the composition relationships between them, and the tags of those APIs as inputs, the posterior distribution of various latent variable $\boldsymbol{\theta}^{(m)}$, $\boldsymbol{\theta}^{(a)}$, $\boldsymbol{\varphi}$ and $\boldsymbol{\varphi}_t$ can be approximated by Gibbs sampling method [27]. Then we can thus obtain the most similar APIs of each mashup. In the training process, a Markov chain is

established and the topic assignment samples are taken from the chain which in turn change the state of the chain. The update rule for the latent topics for the tags of APIs is as follow:

$$p(z_i^{(a)} = j|\mathbf{w}, \mathbf{z}_{\neg i}^{(a)}) \propto \frac{C_{j,\neg i}^{(T_i^{(a)})} + \beta_t}{C_{j,\neg i}^{(\cdot)} + N_t\beta_t} \times \frac{C_{j,\neg i}^{(T^{(a)})} + \alpha}{C_{\cdot,\neg i}^{(T^{(a)})} + T\alpha} \quad (1)$$

Where $\mathbf{z}_{\neg i}^{(a)}$ denotes the vector of all topic assignments excepting $z_i^{(a)}$. $\mathbf{w}$ denotes the vector of all tags of APIs. $C_{j,\neg i}^{(T_i^{(a)})}$ denotes the number of times tag $T^{(a)}$ has been observed with topic *j*. $C_{j,\neg i}^{(\cdot)}$ denotes the number of times tags are assigned to topic *j*. $C_{j,\neg i}^{(T^{(a)})}$ denotes the number of times tags of API are assigned to topic *j*, and $C_{\cdot,\neg i}^{(T^{(a)})}$ is the number of all tags. $C_{\cdot,\neg i}^{(\cdot)}$ indicates that the token *i* is excluded from the corresponding tags of API or topic.

Based on the topic assignments, topic proportions of tag sets and topic distributions over tags can be calculated by the following Equations.

$$p(z_i^{(a)} = j|T^{(a)}) = \frac{C_j^{(T^{(a)})} + \alpha}{C_{\cdot}^{(T^{(a)})} + T\alpha} \quad (2)$$

$$p(T_i^{(a)}|\, z_i^{(a)} = j) = \frac{C_j^{(T_i^{(a)})} + \beta_t}{C_j^{(\cdot)} + N_t\beta_t} \quad (3)$$

Next, the sampling process takes the composition relationships between mashups and APIs into account. The update rule for this purpose is as follow:

$$p(z_i^{(m)} = j|\, \mathbf{z}_{\neg i}^{(\mathrm{m})}, \mathbf{w}, \mathbf{y}) \propto \prod_{a\in A} \exp\left(\frac{\eta}{|W^{(m)}|} \cdot z_j^{(a)}\right) \cdot p(z_i^{(m)} = j|\mathbf{w}, \mathbf{z}_{\neg i}^{(m)}) \quad (4)$$

and

$$p(z_i^{(m)} = j|\mathbf{w}, \mathbf{z}_{\neg i}^{(m)}) = \frac{C_{j,\neg i}^{(w_i^{(m)})} + \beta}{C_{j,\neg i}^{(\cdot)} + N\beta} \times \frac{C_{j,\neg i}^{(w^{(m)})} + \alpha}{C_{\cdot,\neg i}^{(w^{(m)})} + T\alpha} \quad (5)$$

where **w** denotes the vector of all words, **y** denotes the vector of all composition relationships between mashups and APIs. *A* represents all member APIs of mashup *m*. $z_j^{(a)}$ represents the probability value with topic *j* of *a*. Smoothing parameter $\eta$ characterizes the degree of importance of the composition relationship between *m* and *a*. The above update rule is the same with latent topics for description documents of APIs. In this situation, *A* represents all the mashups that have composition relationships with the API *a*. Then, the topic proportions of description documents for mashups and APIs and the topic distributions over words can also be calculated by Equations (2) and (3).

Based on the MAT model, similar APIs of a new mashup can be obtained. Suppose the topic distributions of mashup *m* is $Z^{(m)} = (z_1^{(m)}, z_2^{(m)}, \ldots, z_T^{(m)})$, the topic distributions of API *a* is $Z^{(a)} = (z_1^{(a)}, z_2^{(a)}, \ldots, z_T^{(a)})$. We utilize an enhanced cosine similarity measurement for calculating the composition probability or similarity between *m* and *a* as follows:

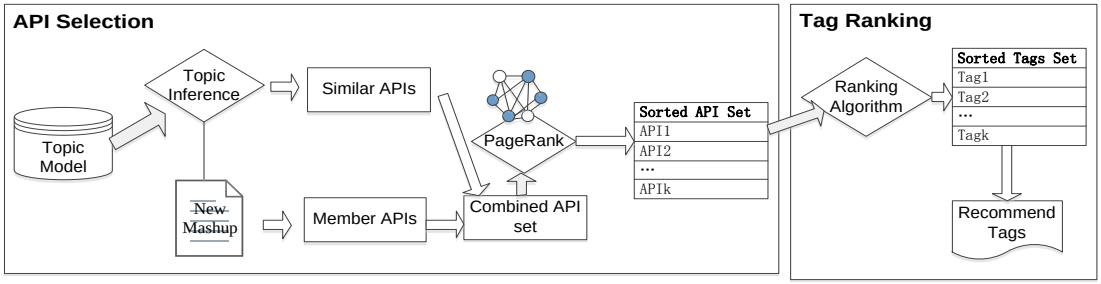



Figure 4. The execution framework of our hybrid approach for tag recommendation.

$$Sim(m,a) = \frac{\sum_{k=1}^{T} \frac{1}{e^{\lambda | z_k^{(m)} - z_k^{(a)} |}} z_k^{(m)} \cdot z_k^{(a)}}{\sqrt{(z_1^{(m)})^2 + \ldots + (z_T^{(m)})^2} \sqrt{(z_1^{(a)})^2 + \ldots + (z_T^{(a)})^2}} \tag{6}$$

A penalty term for dissimilarity between two vectors is introduced into this formula. We assume that, based on the cosine similarity calculation, high similarity score does not means that two documents have similar latent topic distributions, and the penalty term is introduced to address this problem. Parameter $\lambda$ is the penalty severity degree of their dissimilarity. Equation (6) reduces to cosine similarity calculation when we set $\lambda = 0$.

These similar APIs together with the member APIs of this mashup are then combined into a single API set. Next, we describe the PageRank algorithm [28], which is used to further select the most important APIs in order to fetch their tags to recommend. Our purpose is to find out top $n$ APIs according to their importance scores (i.e. PageRank values). PageRank plays an important role in determining the importance of web pages [28], and has been the most commonly used tool for retrieving in web search engines. PageRank can be thought of as a model of user behaviour, where there is a "random suffer" who is given a web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its PageRank value [28]. In order to reduce the effect of dead-ends or endless cycles the surfer will occasionally jump to a random page with a small probability $\varepsilon$, or when on a page with no out-links. Given a network of web pages, PageRank value of a page $u$ is given as:

$$PR(u) = (1-\varepsilon) + \varepsilon \sum_{v \in B_u} \frac{PR(v)}{|F_v|} \tag{7}$$

where $B_u$ is the set of pages that link to page $u$, and $|F_v|$ is the number of pages that $v \in B_u$ links to. Because Equation (7) is recursive, it must be iteratively evaluated until $PR(u)$ converges.

### *3.4. Hybrid Approach for Tag Recommendation*

Figure 4 gives an general presentation of our recommendation approach based on the latent topics obtained. It consists of two steps: APIs selection, and tags ranking. Suppose the topic distribution of $j$th API $a_j$ in the corpus is $Z^{(a_j)} = (z_1^{(a_j)}, z_2^{(a_j)}, \ldots, z_T^{(a_j)})$, calculated from its description document. For a new mashup $m'$ without tags, we denote its topic distribution as $Z = (z_1^{(m')}, z_2^{(m')}, \ldots, z_T^{(m')})$. By Equation (6), the composition probability or similarity between $m'$ and $a_j$ can be computed. Then

Table 2 The generative procedure of the directed graph using all APIs

1. The vertexes of graph are composed by all APIs in the dataset.
2. For two arbitrary APIs $A_u$ and $A_v$ $(u \neq v, u \in E, v \in E)$, construct a link from $A_u$ to $A_v$ if they have one of the following relationships:
   - Co-occurrence relationship between two member APIs of a mashup.
   - Co-occurrence relationship between two APIs annotated with the identical tags.
3. Assign the edge from $A_u$ to $A_v$ a weight based on the following Equation:

$$weight(A_u, A_v) = \frac{sim(A_u, A_v)}{\sum_{t \in P_u} sim(A_u, A_t)} \quad (8)$$

APIs can be sorted in decreasing order according to their similarity value. We finally select several top similar APIs of mashup $m'$, denoted by $A = \{a_1, a_2, a_3, a_4, \dots\}$.

After the above steps, we have selected the most similar APIs of mashup $m'$, but these APIs may contain only few member APIs of this mashup. Next, these similar APIs together with the member APIs of mashup $m'$ are combined into a single API set, denoted by $A' = \{a_1, a_2, a_3, a_4, \dots, b_1, b_2, \dots\}$. Then, we further select top *n* most important APIs of mashup $m'$ from that combined API set based on a weighted PageRank algorithm. We construct a directed graph by using all APIs in the dataset. The generation procedure of this graph is presented in Table 2, where $E$ represents the collection of all APIs. $P_u$ is the API set that API $A_u$ links to. $sim(A_u, A_v)$ indicates the similarity value between $A_u$ and $A_v$ according to Equation (6). Based on above generation procedure, we obtain a graph consisting of 9122 vertexes and 394581edges. Figure 5 presents an example graph. By assigning each edge a weighting value, the PageRank of each API is given as:

$$PR(u) = (1 - \varepsilon) + \varepsilon \sum_{v \in B_u} \frac{PR(v)}{|F_v|} \cdot weight(A_u, A_v) \quad (9)$$

By Equation (9), we can get the PageRank value of each API in API set $A'$ (the shaded vertexes in Figure 5). APIs are then be sorted in decreasing order according to their PageRank values. Finally, we extract all tags of top *n* APIs as the candidate tags to be recommended, denoted by $S = \{t_1, t_2, t_3, t_4, \dots\}$.

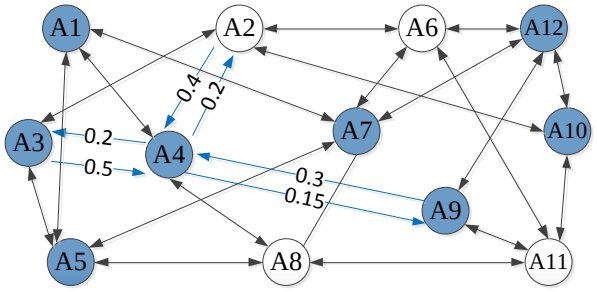



Figure 5. A weighted directed graph among APIs.

In the above step, we have selected the most important APIs to fetch their tags. However, this tag set is still larger than that we can obtain from the real dataset (see the next section). In the following we describe a procedure in Table 3 to further filter the most relevant tags. In Table 3, $t_{i,j}$ is the probability of tag $i$ belongs to topic $j$. $z_j^{(m')}$ denotes element in vector $Z'$ which belongs to topic $j$. $p_j$

Table 3 The procedure of filtering the most relevant tags

1. Order $Z = (z_1^{(m')}, z_2^{(m')}, \ldots, z_T^{(m')})$ to get a sorted list $Z' = (z_3^{(m')}, z_1^{(m')}, z_k^{(m')}, \ldots, z_T^{(m')})$. The elements in the list are sorted in descent.
2. Calculate a recommendation score for each tag in the tag set $S$ by:
$$Score_{t_i} = (1 - \sin(\frac{\mu p_j}{p_j+1} \cdot \frac{\pi}{2})) \cdot z_j^{(m')} \cdot t_{i,j} \quad (10)$$
3. Sort by the recommendation score to get a list of tags $S' = \{t_2, t_1, t_6, t_3, \ldots\}$. We finally recommend top $k$ tags for mashup $m'$.

is the position of $z_j^{(m')}$ in vector $Z'$. For example, the position of element $z_3^{(m')}$ is 1. Variable $\mu$ is a smoothing parameter, to coordinate the degree that the topics of mashup affect the final recommendation score of tags.

Moreover, some tags have been used frequently to annotate similar mashups. These tags have a higher probability to be used again in the future. Each tag has a prevalence value calculated by [4]:

$$P(t) = \frac{U_{t+c}}{U_{t+1}} \quad (11)$$

where $U_t$ is the used times of tag in history by all APIs. By substituting Equation (11) into Equation (10), we obtain the following Equation by considering the popularity of tags:

$$Score_{t_i} = (1 - \sin(\frac{\mu p_j}{p_j+1} \cdot \frac{\pi}{2})) \cdot z_j^{(m')} \cdot t_{i,j} \cdot P(t) \quad (12)$$

By using Equation (6) and (12), we finally recommend tags which are both relevant and prevalent.

## 4 Evaluation

In this section, we conduct experiments to evaluate our recommendation approach from different dimensions. We start the section by discussing the experimental settings, and then we present and analyze the results obtained.

### *4.1. Dataset Description*

The dataset for evaluation was crawled from the programmableweb.com during June 2013. In total, we collected 6673 mashups, 9121 APIs and 13613 links between the mashups and APIs [1].

Figure 6 presents the statistics of tag distribution on the crawled mashups. As can be seen from the figure, in the real world, more than 99% of mashups have been associated with 1 to 6 tags. Therefore, in the experiments described in this section, we report results obtained by recommending 1 to 6 tags. In our evaluation settings, a fivefold cross-validation is performed. All the mashups in the dataset have been roughly split into 5 equal subsets, and each fold in the subsets is used as testing set (i.e. we manually removed all the tags of these mashups and used them as relevant tags when evaluating). The other 4 subsets are combined into a single training dataset. Then the results of each fold are summed up and the averages are reported.

[1] http://blog.csdn.net/shimin520shimin/article/details/51209322.

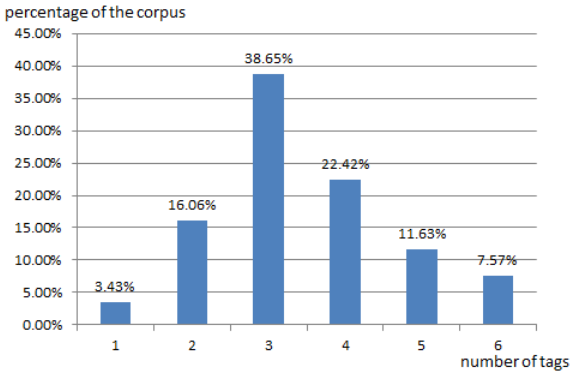



Figure 6. Tag distribution of crawled 6673 mashups.

### *4.2. Metrics*

We use the following metrics to evaluate various tag recommendation algorithms.

- Recall@*k*: the Recall of *top-k* recommendation results. That is the fraction of tags among the real tag set for a mashup that are recommended. It is defined as:

$$Recall@k = \frac{|tags_r \cap tags_m|}{|tags_m|} \tag{13}$$

where $tags_r$ indicates the recommendation result set, and $tags_m$ is the relevant tag set for a mashup.

- Precision@*k*: the Precision of *top-k* recommendation results. That is the fraction of recommendation tags that are among the real tag set for a mashup. It is defined as:

$$Precision@k = \frac{|tags_r \cap tags_m|}{|tags_r|} \tag{14}$$

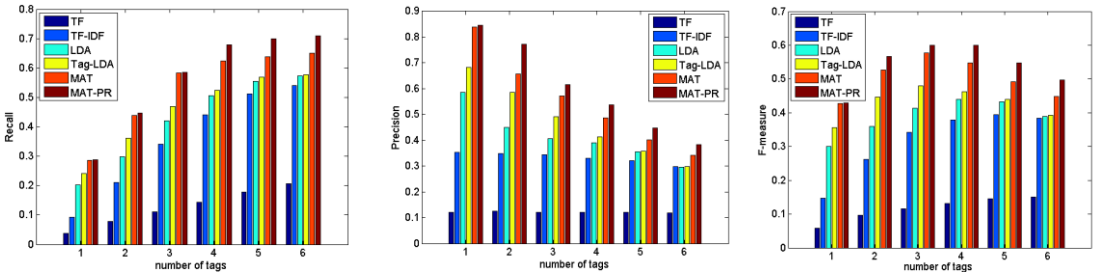
- F-measure@*k*: a single measure that trades off Precision versus Recall. It is defined as:

$$F - measure@k = \frac{2 * Recall * Precision}{Recall + Precision} \tag{15}$$

### *4.3. Approaches Comparison*

We take the following six methods as baselines to evaluate our approach.

- TF [2]: This method recommends tags from highly frequent terms extracted from description document of the mashup itself. We define highly frequent as terms with high *term-frequency*.
- TF-IDF: This method recommends tags of APIs whose description documents are similar to the mashup to be recommended. The similarity calculation between documents is based on term frequency and inverse document frequency (TF-IDF).
- LDA [14]: This method uses the standard LDA model. It recommends tags of similar APIs by enhanced cosine similarity and tag filtering.
- Tag-LDA [2]: Compared with LDA, this method simultaneously takes the annotation relationships between APIs and tags into account, but neglects the composition relationships between mashups and APIs. It recommends tags of similar APIs by enhanced cosine similarity and tag filtering.




(a) (b) (c)

Figure 7. Recommendation performance for different approaches. (a) Recall. (b) Precision. (c) F-measure

- MAT [15]: Compared with LDA, this method further takes the composition relationships between mashups and APIs as well as the annotation relationships between APIs and tags into account, separately. As described in Section 3, it recommends tags of similar APIs by enhanced cosine similarity and tag filtering.
- MAT-PR: Our proposed hybrid approach based on the MAT model and the weighted PageRank algorithm, using enhanced cosine similarity and tag filtering.

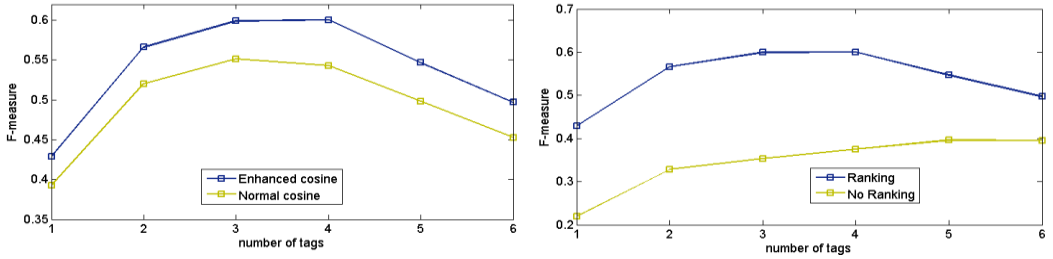
There are various parameters in MAT, such as the topic number $T$, and the prior parameters $\alpha$ and $\beta$. The selection of parameters is a tough task. To train the MAT model, we tested several groups of parameters, and finally selected the following settings that generates the best results. The number of topics $T$ is set to 10, $\alpha$ and $\beta$ are set to 2.0, 0.1 respectively [15]. And the smoothing parameter $\eta$ have been set to 3 and $c$ has a unique value of 0.1 (Equation 10) [26]. In addition, top $n$ important APIs have been selected in our approach, we obtain the best performance by setting $n$=6.

### *4.4. Performance Evaluation*

In this section, we first present the results of all recommendation methods. Then we report the effectiveness of the enhanced cosine similarity calculation used for finding similar APIs of a mashup. Finally, we study the impact of various parameters.

Figure 7 shows the comparisons of recommendation performance for different approaches. We report results by recommending 1 to 6 tags for a mashup. As can be seen from the figure, topic model-based methods including MAT-PR, MAT, Tag-LDA and LDA perform better than frequency-based methods including TF-IDF and TF for all metrics evaluated. As it is known in the community, tags normally have two intrinsic features, the arbitrary of annotation and semantic ambiguity among them [1]. The frequency-based methods only consider lexical matching between documents to obtain the similarity, this may miss some important information. The semantic-oriented methods are superior because they match the documents though latent semantics. The results are quite consistent with previous researches [2][14]. Our proposed hybrid approach performs better than all other methods, with all metrics and with various numbers of tag. The average F-measure of our approach has 7.1% improvement over MAT, 25.5% improvement over Tag-LDA, 39.2% improvement over LDA, 84.6% improvement over TF-IDF, and 402.1% improvement over TF.

To further illustrate the effectiveness of our hybrid approach for mashup tag recommendation, we consider the following three groups of method for comparisons: LDA vs. Tag-LDA, Tag-LDA vs.

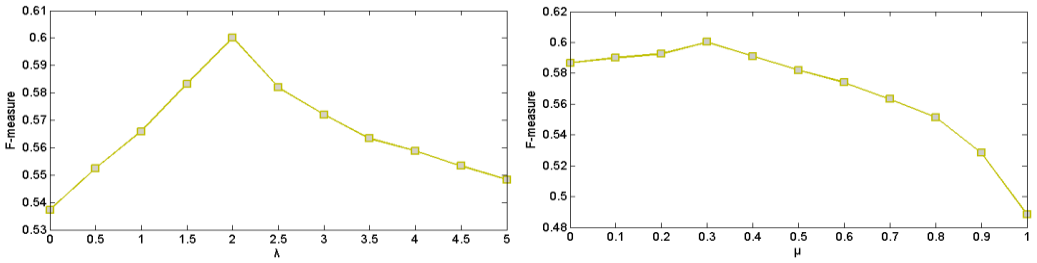



(a) (b)

Figure 8. (a) Comparison of F-measure for normal and enhanced cosine similarity measurements. (b) Comparison of F-measure for filtered tags or no filtering.

MAT, and MAT vs. MAT-PR. As can be observed from Figure 7, Tag-LDA works better than LDA with all three metrics. The reason is that Tag-LDA simultaneously takes the annotation relationships between APIs and tags into account, which can capture the semantics of tags. Thus compared with LDA, Tag-LDA prefers to recommend relevant tags in similar topics with a new mashup. MAT performs better than Tag-LDA in different metrics, the reason is that compared with Tag-LDA, MAT simultaneously takes the composition relationships between mashups and APIs as well as the annotation relationships between APIs and tags into account. Typically, two documents with relation between them are most likely to be about the similar topics [29-30]. Based on the MAT model, if a mashup and an API have composition relationship, they would have similar topic distributions, which will result in a larger proportion of member APIs included in the similar API set of a target mashup. This can improve the recall performance especially in some situations where the relevant tags of a mashup can only be found in its member APIs. On the other hand, the chosen similar APIs contains a high proportion of non-member APIs, which reflects the fact that a lot of non-member APIs provide very similar functionalities with that of the member APIs, and these non-member APIs could be good alternatives while creating mashups. We can also observe from Figure 7 that MAT-PR performs better than MAT. The reason is that MAT still fails to recommend relevant tags only contained by some member APIs, and it also proves that combining the similar APIs and member APIs would further improve the performance.

In the step of similarity calculation between APIs and mashups, we utilize an enhanced cosine similarity. We now examine the effectiveness of this calculation. Figure 8 (a) presents the results of our method (enhanced cosine) compared with the cosine similarity measurement (normal cosine). As shown from the figure, our method performs better than the normal cosine similarity measurement with different number of tags. F-measure of two methods first rises and then drops along with the tag number increases. This phenomenon might be caused by the tag number distribution of mashups. Our method reaches its peak when recommending 4 tags, and it has an average improvement of 7.5% in terms of F-measure than normal cosine similarity. The reason is that in our method, a penalty term for dissimilarity between two vectors is introduced. As we know, for two vectors having large absolute differences only in several corresponding positions, they still have high similarity score based on the normal cosine similarity calculation. But the outcome may be misleading. For example, suppose that mashup $m$ and API $a$ have similar topic probability values on all topics except for topic $t_1$ and topic $t_2$, they would still have a high similarity score based on the normal cosine calculation. While it is obvious




(a) (b)

Figure 9. Impact of parameters. (a) Impact of $\lambda$. (b) Impact of $\mu$.

that mashup *m* and API *a* may have different latent topic distributions (because *m* has a extremely high probability value on topic $t_1$, and *a* has a extremely high probability value on topic $t_2$). Our enhanced cosine calculation can avoid this problem by introducing a penalty term for dissimilarity between two vectors. We also compare performance between methods filtering the candidate recommended tags or not. From Figure 8 (b), we can observe that filtering the tags can highly improve the performance, as the candidate tag set is larger than the tag set we can observe from the real dataset. We obtain 54.3% improvement in average with the recommended tags from 1 to 6.

### *4.5. Impact of Parameters*

In the enhanced cosine similarity calculation, we introduce smoothing parameter $\lambda$ which represents the penalty degree of dissimilarity between two vectors. To study the effect of this parameter, we set it from value 0 to 5 with the increasing interval of 0.5. We can observe from Figure 9 (a) that the F-measure firstly rises and subsequently drops with the changing of $\lambda$, and it obtains the best result when we set $\lambda = 2$. We also test the variable in Equation (10), and Figure 9 (b) shows that F-measure of recommendation result changes along with the growth of value of $\mu$, with the number of recommended tags setting to 4. From the picture, we observe the performance reaches its peak when setting $\mu = 0.3$.

### *4.6. Case Study*

Figure 10 presents an example of tag recommendation for a mashup named "Tag Fight!". The recommendation involves the following processes. Firstly, we select six most similar APIs of this mashup based on the MAT model. Next, the above chosen similar APIs together with two member APIs of this mashup are combined into a single API set. Subsequently, we further select the most important APIs (including 6 APIs) based on a weighted PageRank algorithm. Finally, tags of these most important APIs are recommended to "Tag Fight!" according to their recommendation scores. As can be observed from the picture, tags of this mashup exist either in its most similar APIs or in its member APIs. For example, the tag "game" is only included by the most similar APIs while the tag "photo" is only included by its member APIs. Therefore, we should combine those two API sets to guarantee the recall performance of recommendation.

## 5 Conclusions and Future Work

In this paper, we propose a hybrid approach for automatic mashup tag recommendation. Our hybrid approach consists of two continuous processes: APIs selection and tags ranking. We select the most important APIs to fetch their tags to recommend based on a probabilistic topic model and a weighted

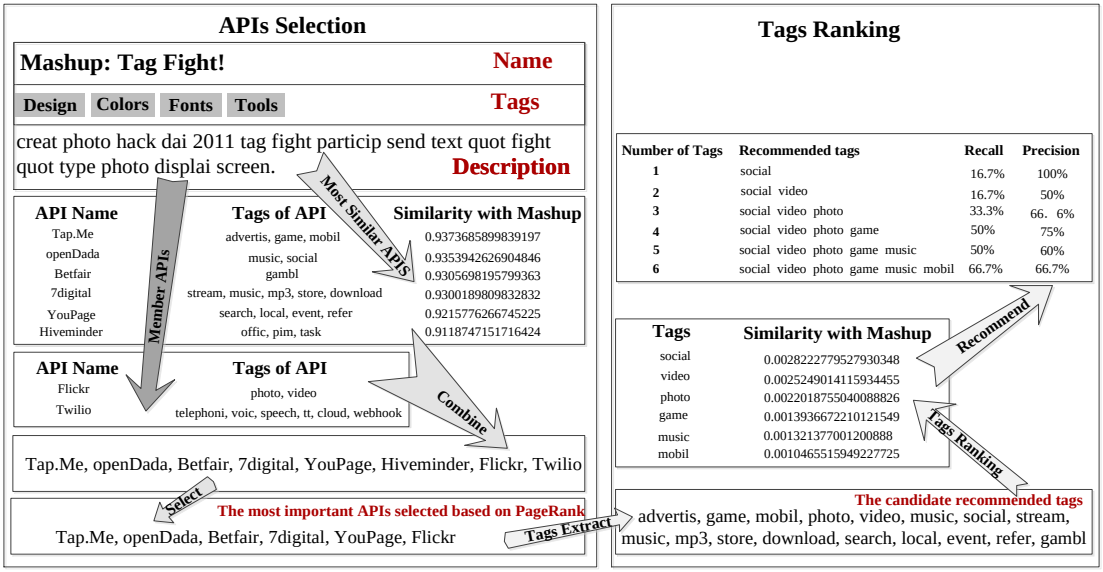



Figure 10. An example tag recommendation for a mashup

PageRank algorithm. In order to reduce the number of recommended tags, we develop a tag filtering algorithm to further select the most relevant tags. We conduct experiments to evaluate our approach. The experimental results also prove that topic model-based methods have a better performance than frequency-based methods. Our hybrid approach outperforms other state-of-the-art topic model-based methods. We further test the effectiveness of the enhanced cosine similarity measure. The experimental results prove that the enhanced cosine similarity measure works better than normal cosine similarity measure. Moreover, combining similar APIs and member APIs to select the most important APIs based on the weighted PageRank algorithm can promote the performance.

In the future, we plan to use our method for social tagging system where the resources have links among them. Moreover, to reduce the time of training process of the topic model in big data environment, employing parallel computing frameworks such as MapReduce to improve the training process is also an interesting direction for future work.

## Acknowledgements

The work described in this paper was supported by the National Natural Science Foundation of China under Project No. 61572187, 61300129 and 61272063, China National Key Technology R&D Program under Grant No. 2015BAF32B01, Scientific Research Fund of Hunan Provincial Education Department of China under Grant No. 16K030, Hunan Provincial Natural Science Foundation of China under Grant No. 2017JJ2101, and the Hunan Provincial Innovation Foundation For Postgraduate under Grant No. CX2016B573.

## References

1. Font F., Serrà J., and Serra X. Analysis of the impact of a tag recommendation system in a real-world folksonomy. ACM Transactions on Intelligent Systems and Technology, 2015, 7(1): 6.
2. Si X.H., Sun M.S. Tag-LDA for scalable real-time tag recommendation. Information & Computational Science, 2009, 6(1): 23-31.

3. Aznag M. Multilabel learning for automatic web services tagging. Advanced Computer Science and Applications, 2014, 5(8): 4910– 4914.
4. Xie, H., Li, X., Wang, T., Lau, R. Y., Wong, T. L., Chen, L. and Li, Q. Incorporating sentiment into tag-based user profiles and resource profiles for personalized search in folksonomy. Information Processing & Management, 2016, 52(1): 61-72.
5. Xie, H., Li, Q., Mao, X., Li, X., Cai, Y. and Zheng, Q. Mining latent user community for tag-based and content-based search in social media. The Computer Journal, 2014, 57(9):1415-1430.
6. Lu F., W L.，Li M and Zhao J.F. Towards automatic tagging for web services//Proceedings of the 19th International Conference on Web Services. Honolulu, HI , 2012: 528- 535.
7. Gawinecki M., Cabri G., Paprzycki M. and Ganzha M. WSColab: Structured collaborative tagging for web service matchmaking// Proceedings of the International Conference on Web Information Systems and Technologies. Valencia, Spain, 2010: 70-77.
8. Lin, M., and Cheung D.W. Automatic tagging web services using machine learning techniques//Proceedings of the International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). Washington, DC, USA, 2014: 285-265.
9. Azmeh Z., Falleri J.R., Huchard M. and Tibermacine C. Automatic web service tagging using machine learning and wordNet synsets// Proceedings of the 6th Web Information Systems and Technologies. Valencia, Spain, 2010: 46-59.
10. Menezes G. V., Almeida J. M. and Belém F. Demand-driven tag recommendation. Machine Learning and Knowledge Discovery in Databases, 2010: 402-417.
11. Chen L., Wang Y., Yu Q., Zheng Z. and Wu J. WT-LDA: User tagging augmented LDA for web service clustering. Service-Oriented Computing, 2013: 162-176.
12. Wang H., Chen B., and Li W. Collaborative topic regression with social regularization for tag recommendation//Proceedings of the 23rd International Conference on Artificial Intelligence, Beijin, China, 2013: 2719–2725.
13. Wang H., Shi X.J and Yeung D. Relational stacked denoising autoencoder for tag recommendation//Proceedings of the 29th AAAI Conference on Artificial Intelligence. Austin, Texas, USA, 2015: 3052-3058.
14. Ralf K., Fankhauser P. and Nejdl W. Latent dirichlet allocation for tag recommendation//Proceedings of the third ACM conference on Recommender systems. New York, USA, 2009: 61-68.
15. Shi M, Liu J.X and Zhou D. A Probabilistic Topic Model for Mashup Tag Recommendation//Proceedings of International Conference on Web Services. Anchorage, AK, 2016: 444-451.
16. Liu Z.Y, Chen X.X. and Sun M.S. A simple word trigger method for social tag suggestion//Proceedings of the Association for Computational Linguistics, Stroudsburg, PA, USA, 2011: 1577-1588.
17. Zhao W, Guan Z, and Liu Z. Ranking on heterogeneous manifolds for tag recommendation in social tagging services. Neurocomputing, 2015(148): 521-534.
18. Gu B., Sheng V.S., and Li S. Bi-parameter space partition for cost-sensitive SVM//Proceedings of the 24th International Conference on Artificial Intelligence, Buenos Aires, Argentina, 2015 :3532-3539.

19. Ma T., Zhou J., and Tang M. Social Network and Tag Sources Based Augmenting Collaborative Recommender System. IEICE Transactions on Information and Systems, 2015, 98(4) :902-910.
20. Wang M., Ni B., Hua X.-S., and Chua, T.-S. Assistive tagging: A survey of multimedia tagging with human-computer joint exploration. ACM Computing Surveys, 2012, 44(4): 25.
21. Belém F.M., Martins E.F, Almeida J.M, Gonçalves M.A. Personalized and object-centered tag recommendation methods for Web 2.0 applications. Information Processing & Management, 2014, 50(4): 524-553.
22. Chang Z. and Blei D.M. Relational topic models for document networks//Proceeding of the 12th international conference on Artificial Intelligence and Statistics. Florida, USA, 2009: 81-88.
23. Chang J. and Blei D. M. Hierarchical relational models for document networks. The Annals of Applied Statistics, 2010, 4(1): 124–150.
24. Wang, Y. and Stroulia, E. Semantic Structure Matching for Assessing Web-Service Similarity.// Proceeding of the 1st international Conference on Service-Oriented Computing, 2003 , 14 (4) :194-207.
25. Fernandez A., Hayes C., Loutas N., Peristeras V., Polleres A., and Tarabanis K. A. Closing the service discovery gap by collaborative tagging and clustering techniques//Proceedings of the 7th International Semantic Web Conference. Karlsruhe, Germany, 2008: 115-128.
26. Li C., Zhang R., Huai J.P. and Sun H.L. A novel approach for api recommendation in mashup development//Proceedings of International Conference on Web Services. Anchorage, AK, 2014:251–258.
27. Heinrich G. Parameter estimation for text analysis. Technical report, vsonix.GmbH and University of Leipzig, Germany, 2004.
28. Brin S. and Page L. The anatomy of a large-scale hypertextual web search engine. The Journal of Computer networks and ISDN systems, 1998, 30(1): 107–117.
29. Li W.J., Yeung D.Y., and Zhang Z.H. Probabilistic relational PCA//Proceedings of the 23rd Advances in neural information processing systems, Hyatt Regency, Vancouver Canada, 2009: 1123–1131.
30. Li W.J., Zhang Z.H., and Yeung D.Y. Latent wishart processes for relational kernel learning//Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, Clearwater Beach, Florida, USA, 2009: 336–343.