SMS: A Framework for Service Discovery by Incorporating Social Media Information

Tingting Liang, Liang Chen*, *Member, IEEE,* Jian Wu, *Member, IEEE,* Guandong Xu, *Member, IEEE,* and Zhaohui Wu, *Senior Member, IEEE*

Abstract—With the explosive growth of services, including Web services, cloud services, APIs and mashups, discovering the appropriate services for consumers is becoming an imperative issue. The traditional service discovery approaches mainly face two challenges: 1) the single source of description documents limits the effectiveness of discovery due to the insufficiency of semantic information; 2) more factors should be considered with the generally increasing functional and nonfunctional requirements of consumers. In this paper, we propose a novel framework, called SMS, for effectively discovering the appropriate services by incorporating social media information. Specifically, we present different methods to measure four social factors (semantic similarity, popularity, activity, decay factor) collected from Twitter. Latent Semantic Indexing (LSI) model is applied to mine semantic information of services from meta-data of *Twitter Lists* that contains them. In addition, we assume the target query-service matching function as a linear combination of multiple social factors and design a weight learning algorithm to learn an optimal combination of the measured social factors. Comprehensive experiments based on a real-world dataset crawled from Twitter demonstrate the effectiveness of the proposed framework SMS, through some compared approaches.

Index Terms—Service discovery, social media, weight learning, social factors combination.

1 Introduction

With the increasing adoption of Service-Oriented Architecture (SOA), services in forms of Web services, cloud services, APIs, mashups, etc, are becoming popular in Web and mobile applications, and we can find a rapid growth in the number of services and their compositions. As a key component of SOA, service discovery is facing some challenges due to the explosive growth of services: (i) The single source of description documents limits the performance of service discovery caused by insufficient semantic information; (ii) More factors should be considered since consumers place more demands on nonfunctional aspects (e.g., popularity, personalization) compared with functional aspects.

Existing service discovery methods are mainly classified into several categories based on semantic similarity [1] [2], collaborative filtering (CF) technique [3] [4], quality of services (QoS) [5] [6], and social network [7] [8]. Most of these approaches are based on information that services originally contain or offered by service providers: functional similarity (semantics), nonfunctional feature (QoS), or social network generated by the collaboration or relationship among services, their compositions and consumers. Facing the rapid expanding of service scale and diversification of data sources, it is a trend to incorporate knowledge from other sources to facilitate service discovery [9] [10]. Recently, service providers tend to run promotions on social media, based on which, they can get direct feedback information and collective knowledge through the interactions on social media. Therefore, it is feasible to enrich available service data by leveraging social media information for improving the performance of service discovery.

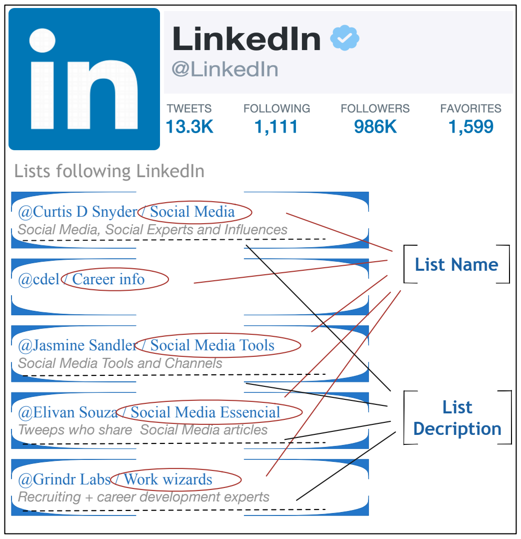
Introducing social media information to improve the performance of search methodology is a popular and novel strategy, and it has been proved effective in many other fields, such as information retrieval, social network, recommender system, etc. Ghosh et al. [11] exploited crowdsourcing information to discover topic experts in Twitter social network. In [12], the author showed the necessity of applying social tags in music information retrieval. In this paper, we intend to exploit collective knowledge mined from social media into the retrieval of RESTful services, i.e., APIs. Through the investigation of ProgrammableWeb[1], a well-known service ecosystem, we found that some APIs have official Twitter accounts managed by their providers. Thus, we can acquire social media information related to APIs from Twitter, the worldwide popular online social network.

There exist various social information in Twitter. Here we firstly introduce an important and practical characteristic in Twitter called *Twitter Lists*. *Twitter Lists* was proposed in late 2009 to help users organize the Twitter accounts they follow [13]. It has been widely used to group sets of users into topical categories. Generally, a list is created with a name (required) and description (optional), and the creator can manage the list members. For a Twitter account, the names and descriptions of lists including it can be considered as its semantic labels. Fig.1 shows several lists that contain LinkedIn, the twitter account of LinkedIn API linked in ProgrammableWeb. It can be obviously found that most

• T. Liang, J. Wu, and Z. Wu are with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, 310027. E-mail: {liangtt,wujian2000,wzh}@zju.edu.cn
• L. Chen and G. Xu are with the Advanced Analytical Institute, University of Sydney Technology, Australia. L. Chen is the corresponding author. E-mail: jasonclx@gmail.com, guandong.xu@uts.edu.au

Manuscript received xx x, xxxx; revised xx x, xxxx.

1. http://www.programmableweb.com


LinkedIn
@LinkedIn

TWEETS 13.3K FOLLOWING 1,111 FOLLOWERS 986K FAVORITES 1,599

Lists following LinkedIn

@Curtis D Snyder / Social Media
Social Media, Social Experts and Influences

@cdel / Career info

@Jasmine Sandler / Social Media Tools
Social Media Tools and Channels

@Elivan Souza / Social Media Essencial
Tweeps who share Social Media articles

@Grindr Labs / Work wizards
Recruiting + career development experts

Fig. 1. Some examples of lists containing LinkedIn

of list names are related to 'Social Media' and 'Career' along with the corresponding list descriptions, which distinctly reflect the topic of LinkedIn. Except for list information, Fig.1 shows some other social features of LinkedIn, such as tweet number, follower number, etc. All of these features reflect status of LinkedIn. For instance, followers of LinkedIn are users interested in it and the number of followers indicates the popularity of LinkedIn in this social network.

In this paper, considering the large scale and various types of information in social media, we try to address the following issues:

- How to reasonably model each social factor for services?
- How to design an effective combination strategy for social factors to generate service retrieval results with high accuracy?

In order to address the proposed issues, we propose a novel framework for Social Media based Service Discovery called SMS. Based on our previous work [14], list meta-data (list names and descriptions) offers significant semantic cues to the topics of involved Twitter accounts. To model the social factor of semantic information provided by collective knowledge, we apply Latent Semantic Indexing (LSI) model to infer services' semantics by analyzing meta-data of those lists including services' Twitter account. LSI maps both the features extracted from lists and the given queries into a conceptual space to calculate similarity between services and queries. Essentially, semantic information mined from Twitter lists is a kind of functional knowledge, since the contents of lists could reflect users' awareness of the functionality of the involved services. To satisfy nonfunctional demand of service discovery, we introduce another three social factors respectively generated from tweet number, follower number and list number: popularity, activity and decay factor.

To effectively combine the semantic similarity and three social factors, we design a weight learning method within the proposed framework SMS. The weight learning method assumes that the target query-service matching function is a linear combination of all the social factors. Then a collection of training triplets are generated based on relative matching degree of two different services to a query. Finally the learning method utilizes the training triplets to learn the optimal combination weights. The learned weights can facilitate the performance of service search for a coming query.

Experiments on a real-world dataset crawled from Twitter are implemented for the demonstration of the effectiveness of SMS. For evaluation, we select 52 sample queries from 12 categories and introduce P@k, nDCG@k and AR@k for the accuracy evaluation. Except for the accuracy evaluation, we introduce an external data source about service providers' ranking to evaluate the popularity mined from social media.

In particular, the main contributions in this paper are summarized as follows:

- We propose a social media based service discovery framework called SMS, where four social factors are modeled from Twitter for the improvement of service retrieval.
- We design a weight learning algorithm to train the optimal weight assignment for the combination of four social factors.
- We conduct comprehensive experiments on a real-world dataset crawled from Twitter to demonstrate the effectiveness of our proposed framework SMS.

The rest of this paper is organized as follows. Section 2 shows some data analysis of social information in Twitter. Section 3 introduces the proposed service discovery framework SMS, including the architecture of SMS, modeling method for each social factor, and the weight learning algorithm for social factors combination. Section 4 presents the detailed evaluation process of SMS. A case study of the proposed SMS is showed in Section 5. Section 6 surveys the related works and Section 7 makes a conclusion of this paper and discusses the future work.

2 DATA ANALYSIS

This section shows some data analysis of social information of services managed by Programmable in Twitter, including list information, tweet number, follower number.

In this work, we crawl 3877 APIs' official Twitter accounts with lists including them, and some related social information. Since there are few work about Twitter list so far, this paper reports some results of the analysis over lists to show the feasibility of our proposed approach. Fig.2 (a) shows the distribution of numbers of APIs based on the number of lists including them. One API could be involved in multiple lists and the most reaches 84511. About 30.8% of total APIs are organized by more than 100 lists, and 70.1% are included by at least 10 lists, which reveals that the list information of most APIs is full of richness. Fig.2 (b) illustrates the distribution of the number of authors based on the number of lists created by them. The total number of authors who created lists for the 3877 APIs is 658,878, of which only 3.8% created more than 10 lists. It is obvious that most authors designed lists less than 10 and the sources of

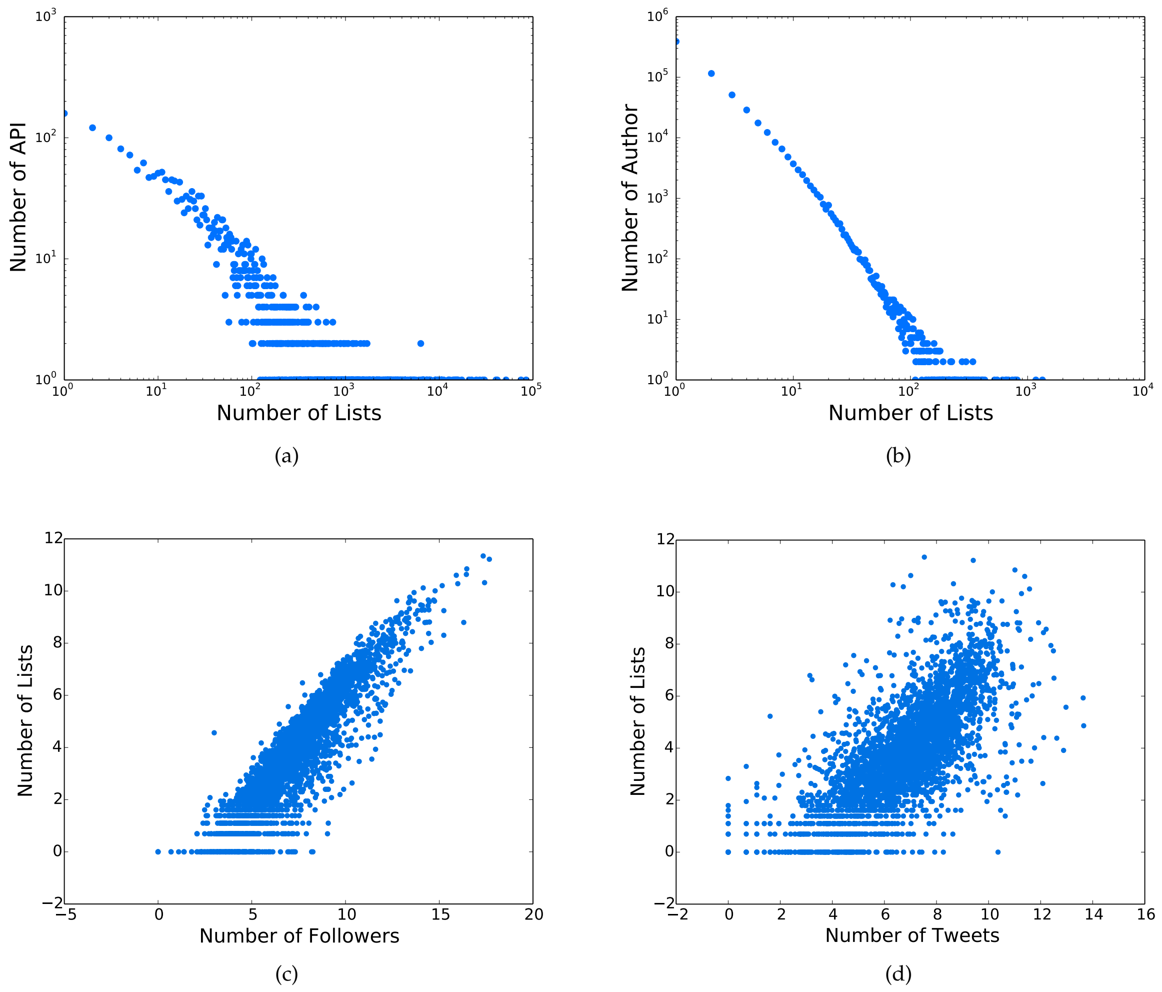


Fig. 2. Data Analysis on Social Information of Services Collected from Twitter

lists are in a wide range, which indicates the diversity of list semantics.

In addition, we show some statistics about the social information collected from Twitter. Fig.2 (c) shows the correlation between the number of lists by which an API is included and the number of followers that an API has. Generally, an API considered to be attractive to many consumers tends to be included in many lists since creating a list can subscribe status of its members, just like following an API. Fig.2 (c) shows the strong correlation between the number of followers and the number of lists. In Fig.2 (d), we show the correction between the number of lists an API is included and the number of tweets that an API posts. Different from the strong relativity showed in Fig.2(c), there only exists weak correlation between list number and tweet number. It means that a popular API might not have much dynamic information, indicating that the API might not be practical or even has been outdated. From the above, the number of an API's followers and the number of tweets posted by an API could be considered as indicators of API popularity and activity, respectively.

3 METHODOLOGY

3.1 Problem Formulation

In this section, we formally formulate the problem of social media based service search.

Definition 1. Query-Service Matching Modeling: *Given a collection of candidate services $\mathcal{S}$ and a set of queries $\mathcal{Q}$, the objective of query-service matching modeling problem is learning a function $f : \mathcal{Q} \times \mathcal{S} \rightarrow \mathbb{R}_+$, such that $f(q_i, s_j)$ measures the matching degree between query q_i and service s_j.*

In our problem, a service is considered to be matched with a given query if it satisfies the requirements of consumers from multiple aspects. Specifically, we take functional similarity between the service and given query as a prior consideration. Based on this, we assume that consumers prefer services that are more popular and more active, since popularity and activity respectively indicate the practical applicability and availability of services.

A crucial element in Definition 1 is a modeled service $s_j \in \mathcal{S}$, which is defined as follows.

Definition 2. Modeled Service: *A service $s_j \in \mathcal{S}$ is structured as a k-dimensional tuple $s_j = [m_{j1}, m_{j2}, ..., m_{jk}]$, where $m_{jl}(1 \leq l \leq k)$ is a social feature of service s_j.*

In this paper, we intend to exploit multiple social features to facilitate service discovery. Specifically, we use four social features of services extracted from Twitter: list content, follower number, tweets number, and list number. These features in general reflect social-level characteristics of services, including semantic similarity (functional), popularity, activity, and decay factor (nonfunctional).

3.2 Architecture of SMS

Fig.3 illustrates the architecture of the social media based service discovery approach SMS. The social media information for service retrieval is crawled from Twitter after being matched with services crawled from ProgrammableWeb. Generally, the whole process of SMS could be divided into four parts:

- The first part is measuring the first social factor, service semantics mined from Twitter lists. This part consists of data pre-processing, semantic similarity calculation by utilizing LSI model based on list meta-data, including list names and descriptions.
- The second part is building functions to measure the rest three nonfunctional social factors including popularity, activity and decay factor.
- All the social factors modeled in the first two parts intend to be used for a proposed weight learning algorithm in the third part. The goal of the weight learning algorithm is to find an optimal combination strategy for social media information.
- In the final part, for a coming query, the new values of modeled four social factors could be integrated based on the learned optimal weight assignment, to generate a list of ranked results.

3.3 Lists based Semantic Similarity

In this section, we model the functional social factor of semantic similarity based on list information of services. We first introduce the content and form of Twitter *Lists*. Then the preprocessing of semantic similarity modeling is described. Finally we show the LSI based approach for text similarity computation.

3.3.1 Leverage Twitter Lists

Lists is introduced for users to manage their followings and the tweets they post. Users can group a set of accounts whom they think have the similar topics by creating a list. A list includes a list name (limited to 25 characters) and a list description that approximately shows the content of the list.

Table 1 shows illustrative examples of lists containing APIs, selected from our dataset (to be detailed in Section 4). As the examples present, Flickr API is included in lists with the name “Photography & design”, “Photography”, “Photography websites” and corresponding descriptions. LinkedIn and lastfm are in the similar situations. It can be observed that the list names and descriptions provide valuable semantic cues to the topics of the APIs in the lists. For instance, according the list meta-data, we can associate Flickr API with photography or photo topic, LinkedIn API with social or profession topic, lastfm with music topic. Thus, lists offer semantic annotations of the APIs included by them. And the semantic feature is generated by arbitrary users, which means it reflects the collective knowledge of crowds. The modeling of the first social factor mainly considers extracting the semantics contained in the crowd-sourced lists.

3.3.2 Data Pre-processing

Due to the arbitrariness of list meta-data (names and description), a pre-processing of lists is needed before applying LSI model. The detailed strategy of extracting features from list meta-data consists of the following steps:

1. **Tokenization**: As list names are limited to 25 characters, users often combine words using *CamelCase* (e.g., PlayStation). It is essential to separate these into individual words by tokenization algorithm.
2. **Stop Words Removal**: Words that are not meaningful for presenting APIs, such as *a*, *the*, *about*, etc, should be removed. In addition to the common stop words, a set of domain-specific stop words also should be filtered out (e.g., Twitter, Google, List).
3. **Nouns & Adjectives Identification**: It has been analyzed in previous work [15] that nouns and adjectives in list names and descriptions are particularly useful for semantics extraction. Thus we identify nouns and adjectives using a tagger tool.
4. **Stemming**: Extracting stem words is another important process. The stem words can be obtained by removing the commoner morphological and inflectional endings from words. For example, *travel*, *traveler*, *traveling* will be replaced with the same stem *travel*.
5. **Low Frequency Words Filtration**: As the definition of list names and descriptions is arbitrary, some words with low frequency are regarded as the impurity in the presentation of APIs and filtered out.

3.3.3 Latent Semantic Indexing

Generally, information is retrieved by literally matching words in documents with those of the given query. However, lexical matching may be inaccurate since some words share the same concept (synonymy) and most words have multiple meanings (polysemy). Moreover, for list meta-data, the drawbacks of literal word matching become more obvious due to personal style and individual differences in word usage. Thus a better method should permit users to search information based on conceptual topics or semantics of documents.

Latent Semantic Indexing (LSI) [16] is a model that maps queries and documents into a space with latent semantic dimensions. In a latent semantic space, high cosine similarity of a query and a document of which words are conceptually similar can be obtained even if they do not share any words. LSI method consists of the following four main steps, of which the first two steps are also applied in vector space models and the step of dimension reduction is the critical part.

A large collection of text is usually represented as a term-document ($t \times d$) matrix X, with each position x_{ij} corresponding to the frequency with which a term (a row i) occurs in a document (a column j). Note that the order

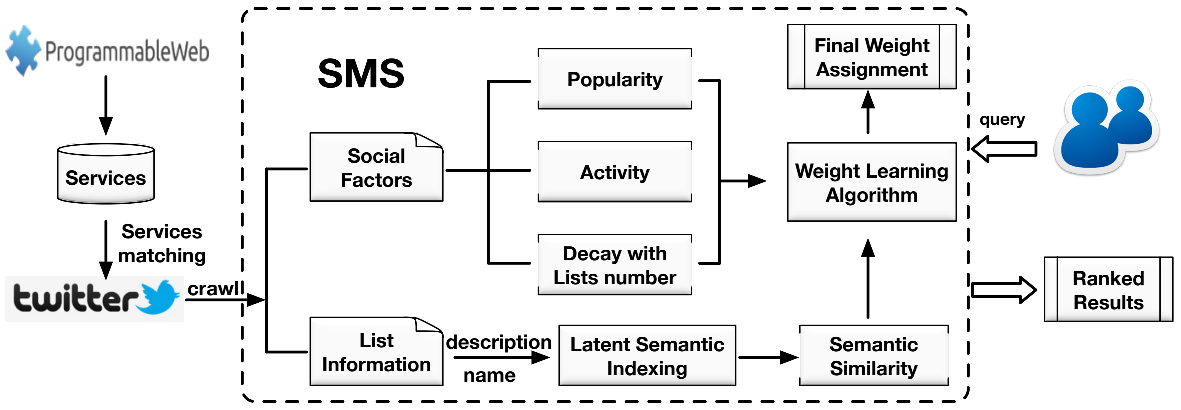


Fig. 3. Architecture of Social Media based Service Discovery

TABLE 1
Example of APIs, the name and description of lists containing them

API Name	List Name	List Description
Flickr	Photography & design	Visual inspiration for my eyes to swallow
	Photography	The art of drawing with light
	Photography websites	Twitter accounts related to Photography you should be following
LinkedIn	Social Media	Bloggers and social media networks
	Professional-Resources	Writing and Design for Interactive Media
	Recruiters	Job Tweets
lastfm	Music	Flagging events or new music
	DJs, Producers, Labels	Music worth listening to
	Home Taping Is Killing Music	Bands and Music; go on, rock my world

of terms in the document is unconsidered in matrix X on the basis of "bag of words". Matrix X is typically sparse since most documents contain only a small percentage of total number of unique terms in the whole corpus.

Instead of dealing with raw term frequencies, the entries in the term-document matrix X are often transformed. An appropriate weighted technique is TF-IDF which is widely used as a weighting factor in text mining and other related fields. TF-IDF value of each word increases with its frequency in corresponding document, while offsets by its frequency in the whole collection. In traditional vector retrieval systems, documents and queries are denoted as vectors in n-dimensional space, where n is the number of indexed terms in the collection. And to apply LSI for document searching, Singular Value Decomposition (SVD) is used to decompose the term-document matrix $X_{t\times d}$ into the product of three matrices, $T_{t\times n}$, a term by dimension matrix, $S_{n\times n}$, a singular value matrix, $D_{d\times n}$, a document by dimension matrix:

$$X_{t\times d} = T_{t\times n} S_{n\times n} {D_{d\times n}}^T,$$

where t denotes the number of terms, d is the number of documents, $n = min(t, d)$, T and D have orthonormal columns:

$$TT^T = I, DD^T = I.$$

Suppose the rank of matrix X is r, the diagonal elements of S are ordered by magnitude,

$$S = diag(\sigma_1, \sigma_2, ..., \sigma_n),$$

where $\sigma_i > 0$ for $1 \leq i \leq r$, $\sigma_i = 0$ for $j \geq r + 1$.

LSI uses a truncated SVD, keeping the k largest singular values and their associated vectors, so

$$\hat{X}_{t\times d} = T_{t\times k} S_{k\times k} {D_{d\times k}}^T,$$

where k is the dimensionality of the reduced space and $k << n$. The resulting reduced-dimension SVD representation is the best k-dimensional approximation to the original matrix in the least-squares sense. The purpose of dimension reduction is to reduce the noise in the latent space, leading to a richer term relationship structure that reveals latent semantics. Through SVD, each term and document is represented as a k-dimensional vector in the latent semantic space. The rows in $T_{t\times k}$ are the term vectors and the rows in $D_{d\times k}$ are the document vectors in latent semantic space.

When LSI is used for retrieval, a query must be represented as a vector in k-dimensional latent semantic space that the document collection is represented in. A query is seen as a set of words like a document, and it can be represented by multiplying the transpose of the query's term vector with $T_{t\times k}$ and ${S^{-1}}_{k\times k}$:

$$\hat{q} = q^T T_{t\times k} {S^{-1}}_{k\times k},$$

where term $q^T T_{t\times k}$ reflects the sum of these k-dimensional term vectors, and the right multiplication with $S^{-1}_{k\times k}$ weights the separate dimensions distinguishingly. By representing query in this way, the distance between documents and given query can be measured using cosine similarity.

Thus, the functional social factor measured by semantic similarity of service s_i and a given query q inferred from list meta-data, which could be defined as:

$$SF^1(q, s_i) = \frac{\sum_j v_{q,j} v_{i,j}}{\sqrt{\sum_j {v^2}_{q,j}} \sqrt{\sum_j {v^2}_{i,j}}}, \tag{1}$$

where $\mathbf{v}$ denotes the term vector of a document or query in latent semantic space.

3.4 Modeling of Nonfunctional Social Factors

Essentially, the social factor of semantic information offered by Twitter lists is a kind of functional knowledge. Based on this, this paper intends to incorporate other social factors to reflect the nonfunctional information of services. In this subsection, we introduce three nonfunctional social factors and build functions to measure them for each service. Without loss of generality, all values are within the range of $[0, 1]$.

3.4.1 *Popularity Factor*

To satisfy consumers' demand of popular services, we introduce the popularity factor. We assume that, when faced with two services providing similar functions, consumers are more likely to select the more popular one. Based on Twitter information, we consider the number of a service's followers could reflect the degree of being concerned of the service. Due to the large gap between the follower numbers of different services, we normalize follower numbers with their logarithmic form. Thus, we define the popularity of service s_i according to its follower number as follows,

$$SF^2(s_i) = \frac{\log n_f^i - \log min_f}{\log max_f - \log min_f}, \tag{2}$$

where n_f^i denotes the follower number of service s_i, min_f and max_f respectively denote the minimal and maximal follower numbers.

3.4.2 *Activity Factor*

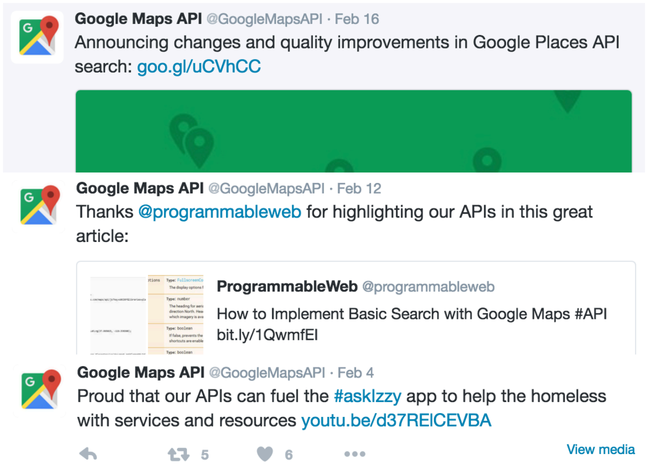
Service accounts could post tweets since they are users in Twitter. Generally, the tweets or retweets posted are about improvements or updates of services, news and trends about services, etc. as showed in Fig.4. Thus we consider a service is active if it posts tweets frequently. In this paper, we assume that active services are constantly invoked and more valuable to consumers. Analogous to popularity factor mentioned above, we define the activity of service s_i based on its tweet number, formally,

$$SF^3(s_i) = \frac{\log n_t^i - \log min_t}{\log max_t - \log min_t}, \tag{3}$$

where n_t^i denotes the number of tweets posted by service s_i, min_t and max_t respectively denote the minimal and maximal tweet numbers.

3.4.3 *Decay Factor*

Services included by lists with very high frequency are likely to be too general for consumers. We try to damp the contribution of candidate services that included by too many lists. We build the following decay function to generate social factor.


Google Maps API @GoogleMapsAPI · Feb 16
Announcing changes and quality improvements in Google Places API search: goo.gl/uCVhCC

Google Maps API @GoogleMapsAPI · Feb 12
Thanks @programmableweb for highlighting our APIs in this great article:

ProgrammableWeb @programmableweb
How to Implement Basic Search with Google Maps #API bit.ly/1QwmfEI

Google Maps API @GoogleMapsAPI · Feb 4
Proud that our APIs can fuel the #askIzzy app to help the homeless with services and resources youtu.be/d37REICEVBA

5 6 View media

Fig. 4. Instance of tweets posed by Google Maps API

$$SF^4(s_i) = \frac{k_d}{k_d + abs(k_d - \log n_l^i)}, \tag{4}$$

where n_l^i denotes the number of lists containing service s_i and parameter k_d is designed as the logarithmic form of average list number.

3.5 Learning Optimal Weights for Social Factors

In the previous subsections, we totally introduce four modeling functions (SF^1, SF^2, SF^3, SF^4) to measure different social factors for services. The next crucial problem is how to find the best way to integrate these social factors into the target query-service matching function. In this work, we propose a weight learning algorithm to discover the optimal combination of social information, in which the target query-service matching function f is assumed as a linear combination of the multiple modeling functions as follows,

$$K(q, s_i) = w_1 SF^1(q, s_i) + \sum_{k=2}^{n} w_k SF^k(s_i) \tag{5}$$

where s_i is the i-th service and $K(q, s_i)$ denotes the target query-service matching function f. SF^k represents the modeling functions for social factors defined under the the k-th ($1 \le k \le n$) social views of services, n is the total number of social factors, and $\mathbf{w} \in \mathbb{R}^n$ represents the weight vector.

One direct way to figure out the target query-service matching function is to manually assign the weights of different social factors. However, such method has poor performance in efficiency and hardly finds the best combination. Therefore, this paper utilizes online learning technique to learn the optimum combination weights $\mathbf{w}$ from streams of triplets. The main advantages of the weight learning algorithm are that (i) it can reduce the training cost during the learning process; and (ii) it can figure out more appropriate combination weights to improve the effectiveness of service retrieval.

3.5.1 *Relative Matching Degree Learning*

In the training step, we assume a collection of training instances represented in the form of triplets as follows,

$$\mathcal{Z} = \{(q_i, s^+_{q_i}, s^-_{q_i}), i = 1, ..., m\}$$

where each triplet $(q_i, s^+_{q_i}, s^-_{q_i})$ indicates that service $s^+_{q_i}$ is better matched with the i-th query q_i compared with service $s^-_{q_i}$. m is the total number of queries in the training data. Our target is to learn the optimum combination weights $\mathbf{w}$, which lead a higher query-service matching score to service that is more relevant to the given query q_i,

$$K(q_i, s^+_{q_i}) > K(q_i, s^-_{q_i})$$

Based on such a triplet setting, it is only need to give the relative order of matching degree rather than an exact measure of matching score, which is more feasible in most scenarios.

3.5.2 *Weight Learning Algorithm for Social Factors*

Based on the side information in the form of triplet as defined above, the weight learning algorithm learns a query-service matching function $K(q, a_i)$, for all triplets in $\mathcal{Z}$ satisfying,

$$K(q_i, s^+_{q_i}) > K(q_i, s^-_{q_i}) + \varepsilon$$

where ε is a margin factor which should be positive. We set ε equal to 1.0 in our experiments. A hinge loss is defined for each triplet $(q_i, s^+_{q_i}, s^-_{q_i})$ as follows,

$$\begin{aligned} h(q_i, s^+_{q_i}, s^-_{q_i}) &= max\{0, \varepsilon - K(q_i, s^+_{q_i}) - K(q_i, s^-_{q_i})\} \\ &= max\{0, \varepsilon - \mathbf{w} \cdot \mathbf{d}^+_{q_i} + \mathbf{w} \cdot \mathbf{d}^-_{q_i}\} \end{aligned}$$

where $\mathbf{d}^+_{q_i} = [SF^1(q_i, s^+_{q_i}), SF^2(s^+_{q_i}), SF^3(s^+_{q_i}), SF^4(s^+_{q_i})]^T$ and $\mathbf{d}^-_{q_i} = [SF^1(q_i, s^-_{q_i}), SF^2(s^-_{q_i}), SF^3(s^-_{q_i}), SF^4(s^-_{q_i})]^T$.

We aim to obtain the optimal combination weights $\mathbf{w}$ that minimize the object function,

$$\min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{M} \sum h(q_i, s^+_{q_i}, s^-_{q_i}) \tag{6}$$

where M denotes the total number of training instances and λ is a regularization parameter.

In order to detailedly illustrate the core procedure of the proposed online weight learning algorithm for solving the optimization problem described in Eq.(6), we present the pseudo-code as showed in Algorithm 1. The inputs consist of a collection of training triplets $\mathcal{Z}$, a regularization parameter λ, the number of iterations T, and a learning rate $\beta^{(0)}$. Initially, we assign each social factor with the same weight. The combination weights $\mathbf{w}$ could be set as $[w^{(1)}_1, ...w^{(1)}_n]$, where each weight $w^{(1)}_k = 1/n, \forall k = 1, ..., n$, n is the number of social factors. During each training iteration t, we set the learning rate $\beta^{(t)} = \beta^{(0)}/(1 + \lambda\beta^{(0)}t)$. Then a triplet $(q^{(t)}_i, s^{+(t)}_{q_i}, s^{-(t)}_{q_i})$ is obtained from collection $\mathcal{Z}$ and we respectively compute $\mathbf{d}^{+(t)}_{q_i}$ and $\mathbf{d}^{-(t)}_{q_i}$. The objective function corresponding to this triplet is:

$$\mathcal{L}^{(t)}(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + h(\mathbf{w}; (q^{(t)}_i, s^{+(t)}_{q_i}, s^{-(t)}_{q_i})) \tag{7}$$

The sub-gradient of $\mathcal{L}^{(t)}(\mathbf{w})$ with respect to $\mathbf{w}$ is computed as:

$$\triangledown^{(t)} = \frac{\partial \mathcal{L}^{(t)}(\mathbf{w})}{\partial \mathbf{w}} = \begin{cases} \lambda \mathbf{w}^{(t)} & \text{if } \mathbf{w}^{(t)} \cdot \Delta\mathbf{d}^{(t)}_{q_i} \geq \varepsilon \\ \lambda \mathbf{w}^{(t)} - \Delta\mathbf{d}^{(t)}_{q_i} & \text{if } \mathbf{w}^{(t)} \cdot \Delta\mathbf{d}^{(t)}_{q_i} < \varepsilon \end{cases}$$

where $\Delta\mathbf{d}^{(t)}_{q_i} = \mathbf{d}^{+(t)}_{q_i} - \mathbf{d}^{-(t)}_{q_i}$.

Then the new combination weights can be updated by $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \beta^{(t)}\triangledown^{(t)}$, that is,

$$\mathbf{w}^{(t+1)} = \begin{cases} (1 - \beta^{(t)}\lambda)\mathbf{w}^{(t)} & \text{if } \mathbf{w}^{(t)} \cdot \Delta\mathbf{d}^{(t)}_{q_i} \geq \varepsilon \\ (1 - \beta^{(t)}\lambda)\mathbf{w}^{(t)} + \beta^{(t)}\Delta\mathbf{d}^{(t)}_{q_i} & \text{if } \mathbf{w}^{(t)} \cdot \Delta\mathbf{d}^{(t)}_{q_i} < \varepsilon \end{cases}$$

Through T iterations, the final learned weight vector $\mathbf{w}^{(T+1)}$ would be output as the optimal combination weights for multiple social factors.

Algorithm 1 Weight Learning for Social Factors

```
Input:
    Collection of training triplets $\mathcal{Z}$;
    Regularization parameter $\lambda$, learning rate $\beta^{(0)}$;
    Iteration number $T$
Output:   Combination weights $\mathbf{w}^{T+1}$
 1: Initialize weights: $w^{(1)}_k = 1/n, \forall k = 1, ..., n$.
 2: for $t = 1, 2, ..., T$ do
 3:     $\beta^{(t)} = \beta^{(0)}/(1 + \lambda\beta^{(0)}t)$.
 4:     Get one triplet $(q^{(t)}_i, s^{+(t)}_{q_i}, s^{-(t)}_{q_i})$ from collection $\mathcal{Z}$.
 5:     Compute $\mathbf{d}^{+(t)}_{q_i}$ and $\mathbf{d}^{-(t)}_{q_i}$.
 6:     if $\mathbf{w}^{(t)} \cdot \mathbf{d}^{+(t)}_{q_i} - \mathbf{w}^{(t)} \cdot \mathbf{d}^{-(t)}_{q_i} \geq \varepsilon$ then
 7:         $\mathbf{w}^{(t+1)} \leftarrow (1 - \beta^{(t)}\lambda)\mathbf{w}^{(t)}$.
 8:     else if $\mathbf{w}^{(t)} \cdot \mathbf{d}^{+(t)}_{q_i} - \mathbf{w}^{(t)} \cdot \mathbf{d}^{-(t)}_{q_i} < \varepsilon$ then
 9:         $\mathbf{w}^{(t+1)} \leftarrow (1 - \beta^{(t)}\lambda)\mathbf{w}^{(t)} + \beta^{(t)}(\mathbf{d}^{+(t)}_{q_i} - \mathbf{d}^{-(t)}_{q_i})$.
10:     end if
11: end for
12: return $\mathbf{w}^{(T+1)}$
```

The time complexity of weight learning algorithm includes two parts. The first part is about the generation of triplets set for training, the time complexity is $O(|q|N^2)$, where $|q|$ is the size of query set and N is the number of APIs. Another part is weight learning process and its time complexity is $O(T)$, where T is the predefined iterations.

4 EVALUATION

4.1 Dataset and Setup

The datasets used in this paper are respectively crawled from Twitter and ProgrammableWeb, and named as D_p and D_t. First we crawl 10,850 APIs from PW, then 3,877 API official Twitter accounts are crawled from Twitter, matching with APIs in ProgrammableWeb. 414 out of total 3,877 APIs with no list are removed since the kernel of the modeling of two social factors is leveraging list meta-data. D_p contains API name, description, category, while D_t includes API name, list name, list description, follower number and tweet number of APIs. The number of lists in D_t reaches 1,015,128 created by totally 658,878 twitter users. Besides, we crawl a dataset about API provider rank from Alexa [2] as an evaluation criteria for the ranking of APIs.

2. Alexa: http://www.alexa.com/

TABLE 2
The 52 Sample Queries for Evaluation

Category	Sample queries
Music	audio, band, lyrics, singer, sound, music
Travel	hotel, tourism, flight, holiday, voyage, travel
Financial	stock, investment, price, finance, economics
eCommerce	shopping, eCommerce
Mapping	GIS, geography, GPS, mapping, traffic
Sports	NFL, fitness, sport, running, athlete
Photos	photo, image, picture, camera
Government	policy, congress, department, government, law
Game	player, game
Education	education, training, student, school
Enterprise	sales, hr, leader, enterprise
Social	social network, media, social

All the experiments in this paper are implemented with Python 2.7, conducted on a MacBook Pro with an *2.2 GHz Intel Core i7 CPU* and *16 GB 1600 MHz DDR3 RAM*, running *OS X Yosemite*.

4.2 Evaluation Methodology

For evaluation, a method of binary judgment should be applied to identify whether the returned API is relevant to the given query. The queries used for evaluation could be chosen from a given set of 52 sample queries that are extracted from topics of the 12 categories shown in Table 2. The idea of binary judgment in the evaluation is: a returned result for a given query can be judged as relevant once its category is identical with the category of the query. Besides binary judgment, we adopt provider rank which represents the ranking of API provider's site to estimate the influence of popularity mined from social media, following the intuition that an API generated by a provider whose site is ranked higher would be regarded as the relatively more popular one. We use the following three metrics to evaluate the proposed SMS.

4.2.1 Precision@k

Precision measures the proportion of returned documents that are relevant. Precision at cutoff k (P@k) is calculated as:

$$\text{P@}k(r) = \frac{1}{k}\sum_{i=1}^{k} r_i, \tag{8}$$

where r_i indicates the relevance of the ith ranked result scored as either 0 (not relevant) or 1 (relevant).

4.2.2 nDCG@k

Normalized discounted cumulative gain (nDCG) is an explicitly rank-weighted metric. For a ranked list retrieved for the user, the further down the list a result occurs, the more its value to the user is discounted. If binary relevance is determined, with R relevant results for a query, the formula of nDCG@k is:

$$\text{nDCG@}k(r) = \frac{\sum_{i=1}^{k} r_i \cdot w_{DCG}(i)}{\sum_{i=1}^{\min\{k,R\}} w_{DCG}(i)}, \tag{9}$$

where r_i indicates the relevance of the ith ranked result, and

$$w_{DCG}(i) = \frac{1}{\log_2(i+1)}$$

4.2.3 Average Rank@k

In our evaluation, Average Rank at cutoff k (AR@k) is used for measuring popularity of search results. AR@k is calculated as:

$$\text{AR@}k(r) = \frac{1}{k}\sum_{i=1}^{k} rank_i, \tag{10}$$

where $rank_i$ denotes the ranking score of the ith ranked result, and the value of $rank_i$ is obtained from the provider rank of each API. A lower AR@k indicates a higher rank.

4.3 Experimental Setup

4.3.1 Training Triplets Generation

For the evaluation of the proposed weight learning algorithm, we need to create a collection of triplets for training. First we randomly select a part of the predefined 52 queries mentioned in evaluation methodology as training queries. For each selected query q_i, we define a $matchset_{q_i}$ including APIs whose categories are the same as that of q_i. Similarly, a $unmatchset_{q_i}$ is defined for APIs which belong to categories different with q_i. Next, we randomly sample an API service $s_{q_i}^{+}$ from $matchset_{q_i}$. Then an API service $s_{q_i}^{-}$ is randomly chosen from set $unmatchset_{q_i}$. Do the same thing to all the training queries. Through such process, a collection of training triplets $\mathcal{Z}$ would be generated for combination weights learning in our experiment.

4.3.2 Compared Methods

We compare the following methods in our experiments:

- **PW**: Descriptions of APIs in ProgrammableWeb are used for semantic matching by LSI model. We set this method as baseline.
- **List Only**: We leverage Twitter list information of APIs to compute the semantic similarity using LSI model for ranking APIs.
- **+Pop**: We integrate semantic similarity mined from list information of APIs with popularity factor referred from APIs' follower number.
- **+Act**: We integrate semantic similarity mined from list information of APIs with activity factor reflected by APIs' tweet number.
- **+Decay**: We integrate semantic similarity mined from list information of APIs with decay factor generated from APIs' list number.
- **Uniform**: The semantic similarity and three nonfunctional social factors are uniformly combined with the same weight.
- **SMS**: We use the proposed weight learning algorithm for integrating the modeled four social factors. Specifically, we run SMS with $\beta^{(0)} = 10^{-4}$, $\lambda = 10^{-4}$ and $T = 20000$.

Note that the way of integration in +Pop, +Act and +Decay is searching and ranking APIs for query q based on $SF^k(s_i) \cdot SF^1(q, s_i)(2 \le k \le 4)$, where $SF^k(s_i)$ is the k-th nonfunctional social factors of API s_i.

TABLE 3
Comparison of PW and List Only based on 15 test queries

Query	P@10		nDCG@10	
	PW	List Only	PW	List Only
enterprise	0.20	**0.60**	0.46	**0.97**
social	**0.80**	0.30	**0.99**	0.46
investment	0.60	**0.70**	**0.84**	0.78
economics	0.00	**0.60**	0.00	**0.70**
eCommerce	**0.40**	0.30	**0.91**	0.59
sound	0.00	**0.30**	0.00	**0.43**
music	**1.00**	0.40	**1.00**	0.48
voyage	0.00	**1.00**	0.00	**1.00**
travel	**1.00**	**1.00**	**1.00**	**1.00**
traffic	**0.20**	0.10	**0.40**	0.39
athlete	0.10	**0.50**	**1.00**	0.72
photo	**1.00**	0.8	**1.00**	0.82
government	0.20	**0.40**	0.58	**0.92**
game	**0.70**	**0.70**	**0.97**	0.87
school	0.10	**0.90**	0.43	**0.98**
Average	0.42	**0.57**	0.64	**0.74**

4.4 Result Analysis

In the first two evaluation parts, we randomly select 70% of the 52 queries as the training queries and a collection of training triplets is generated. The rest 15 queries are used for testing.

4.4.1 *Effectiveness of Twitter Lists*

In this part, we compare the performance of PW and List Only, which have difference in the employed data sources, to show the effectiveness of Twitter lists. Since neither of the two methods incorporates factors influencing the popularity of APIs, we evaluate them in terms of P@10 and nDCG@10. As showed in Table 3, it can be observed that for most queries, List Only outperforms PW based on P@10. In terms nDCG@10, PW performs unstably in each query though it outperforms List Only on more than half of queries. There even exist three queries that are not matched with any relevant APIs. On average, the values of P@10 and nDCG@10 of List Only are higher than those of PW, which indicates that the crowdsourcing based semantic information provided by Twitter lists can facilitate the quality of API search. There is still room to improve the performance of Twitter lists based method, for which we intend to incorporate some other social factors and evaluate improved methods in the next part.

4.4.2 *Methods Comparison*

We evaluate and compare all the methods listed in Section *4.3.2* in terms of P@k, nDCG@k and AR@k ($k \in \{5, 10, 15\}$). Fig.5 shows the top-k API retrieval results based on the average metric values of 15 test queries, from which we can draw some observations.

First of all, in terms of P@k and nDCG@k, SMS always achieves the best performance among all the evaluated methods. Specifically, we find that (i) compared with Uniform, SMS improves P@k and nDCG@k by at least 11% and 3%, which illustrates the effectiveness of the weight learning algorithm for social information integration; (ii) SMS shows its superiority when compared with methods integrated with one nonfunctional social factor (+Pop, +Act and +Decay), which caused by the contributions made by multiple social factors.

Second, compared with the baseline, all other methods achieve better performances based on all the three metrics, which shows the superiority of the incorporation of social information collected from Twitter (list data, follower number, tweet number). Obviously, different social information and different combination strategies provide distinguished improvements for API search performance.

Third, Uniform performs not well as those methods integrated with one social factor (+Pop, +Act and +Decay). The possible reason is the inappropriate weight assignment for lists based semantic information and each nonfunctional social factor. It indicates that the general weight setting way does not necessarily lead satisfied result even though it combines all the social information.

Finally, +Pop, +Act and +Decay outperform List Only in terms of all the metrics to varying degrees, which indicates different influences are generated when incorporating different social factors. Among the three methods, +Pop and +Decay achieve similar performance. The possible reason is that the two methods respectively leverage API follower number and list number which have strong correction as discussed in Section 2. +Act shows the relatively worse performance and it illustrates that the positive affect generated by activity factor is weaker than those generated by popularity and decay factors.

4.4.3 *Influence of Social Factors and Combination Strategy*

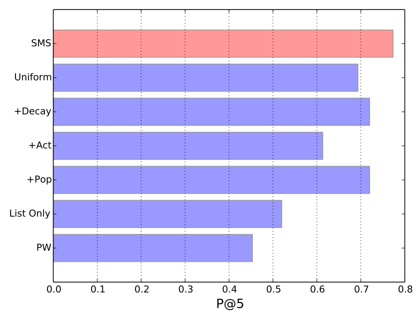
In this part, we specifically evaluate the influence of different social factors and different combination strategies with respect to different amount of training queries. We randomly select 20%, 30%, 40%, 50%, 60% and 70% of the predefined queries as training queries. In order to directly demonstrate the effectiveness of our proposed weight learning algorithm, we add the following methods in this evaluation:

- **WL+P**: We use the weight learning algorithm for integrating the semantic similarity with popularity factor.
- **WL+A**: We use the weight learning algorithm for integrating the semantic similarity with activity factor.
- **WL+D**: We use the weight learning algorithm for integrating the semantic similarity with decay factor.

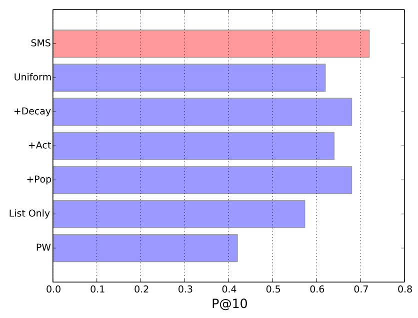
Fig.6 reports the API discovery performance when different social factors and different combination strategies for two factors are utilized in terms of P@10, nDCG@10 and AR@10. We can draw some observations from Fig.6.

In terms of P@10 and nDCG@10, WL+P (+A, +D) obviously achieves the best performance compared with List Only and +Pop (+Act, +Decay), which demonstrates that the weights learned by the proposed algorithm can more effectively combine lists based semantic information and one of the social factors. Though the methods simply multiply semantic similarity ($SF^1(q, s_i)$) with single nonfunctional social factor ($SF^k(s_i)(2 \leq k \leq 4)$) underperform in a bit, they still make great progress compared to List Only.

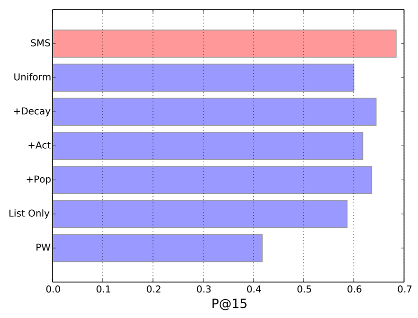
In terms of AR@10, we do not consider the performance of methods incorporating activity factor since they contain



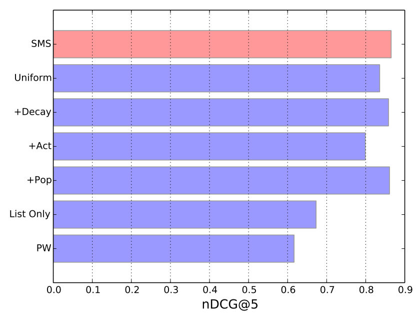
(a)



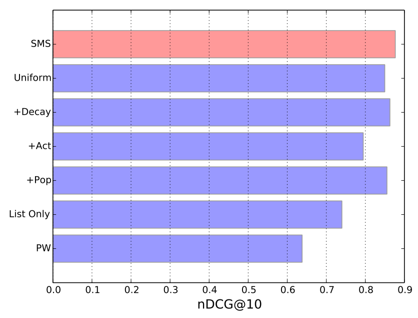
(b)



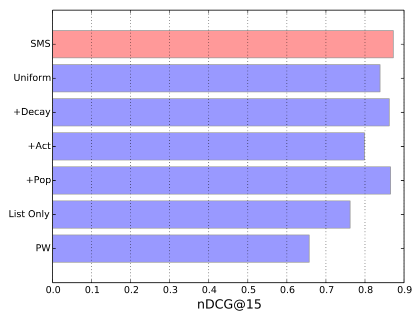
(c)



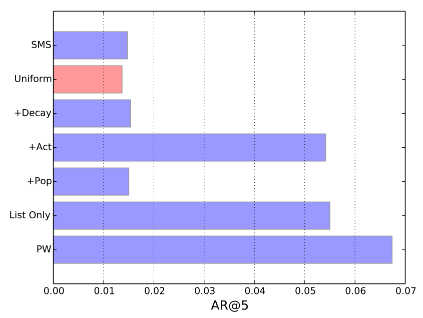
(d)



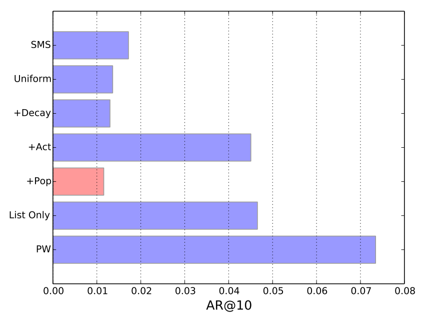
(e)



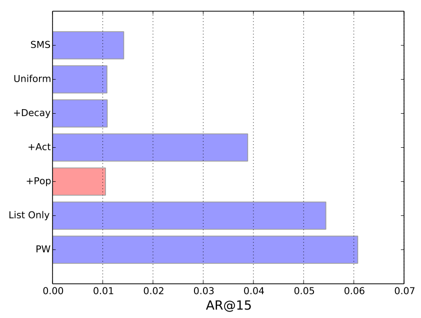
(f)



(g)



(h)



(i)

Fig. 5. Comparison of API Search Methods in terms of P@k, nDCG@k, and AR@k (the bar in red indicates method with the best performance)

hardly any popularity information. Both of WL+P (+D) and +Pop (+Decay) outperform List Only. However, +Pop and +Decay performs better than WL+P and WL+D, which is quite different with P@10 and nDCG@10 based results. It probably caused by the weakening of popularity factor as the weight training algorithm may assign larger weight for semantic information.

From Fig.6 (a)-(c), according to the blank spaces between broken lines of List Only and WL+P (+A, +D), it can be easily found that the popularity factor makes more significant influence to search result in terms of P@10 compared to the other social factors. The same conclusions could be drawn based on nDCG@10 and AR@10 by the rest figures.

There exist some apparent drops of performance on P@10 and nDCG@10 with the increase of percentage of training queries , especially when the training queries changes from 60% to 70%. The possible reason is that the number of queries for testing are reduced and the proportion of queries which is inherently hard to find relevant APIs increases.

5 Case Study

This section illustrates retrieval cases for two queries: *athlete* and *school*, in order to better understand the distinction of the baseline method PW and the proposed SMS that leverages multiple social media information. For the two queries, Table 4 reveals the comparison between the top 5 results respectively generated by PW and SMS, along with the description, category and provider rank.

As showed in Table 4, the top 5 APIs returned for query *athlete* by PW method come from various categories, of which only the first API called Athlinks matches with the topic of *athlete*. Since PW method extracts description information offered by API providers, the semantics can be discovered is limited. In contrast, 80% of the top 5 APIs returned by SMS are consistent with the category of *athlete* and the extracts from description, such as "*running events*",

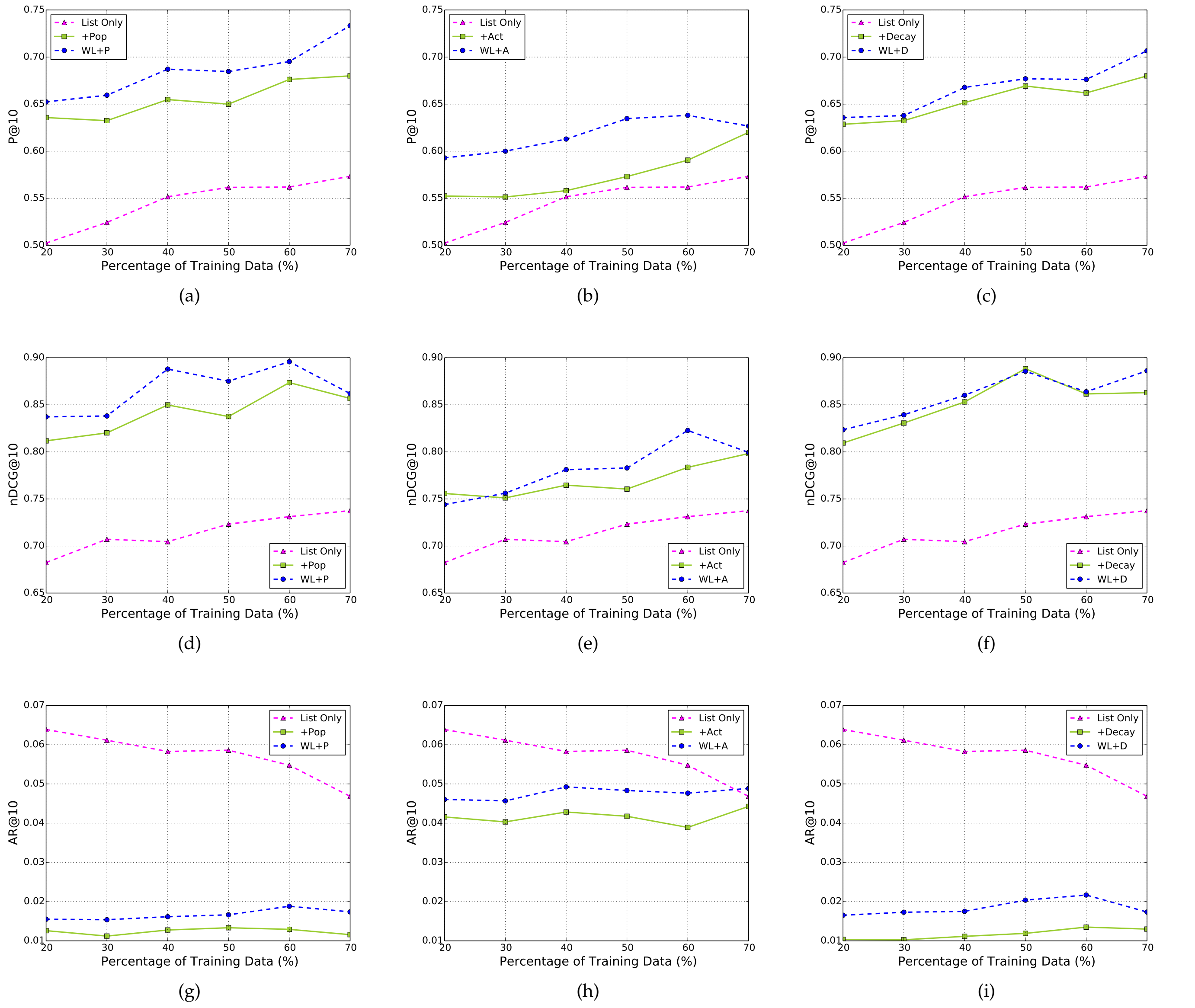


Fig. 6. Influence of Different Social Factors and Different Combination Strategies

"*learn from route*" and "*fitness*", show the reasonableness of results. It is lead by the collective knowledge of crowds inferred from Twitter lists and its optimal integration with nonfunctional social factors. For query *school*, the search result is similar with the case of query *athlete*. All of the top 5 APIs of SMS are related to the given query while only one of top 5 APIs returned by PW method belongs to Education category.

Furthermore, the provider rank in SMS is overall higher than that in PW method, which indicates that the incorporation of social factors improves the popularity of returned APIs.

6 Related Work

Research in the fields of service discovery has been widely published in recent years, specially including service recommendation, ranking and searching, which is the main work in this paper. In this section, we summarize them into several categories based on the methods they employ.

Semantic-based approaches: This kind of methods generally consider the semantic relevance of services. Wu et al. [1] [17] proposed a clustering approach utilizing both semantics mined from WSDL documents and tags to improve service search. Lee et al. [2] showed how to syntactically define and semantically describe the characteristics of APIs and how to use these descriptions to easily search and composite APIs. Chan et al. [18] proposed a practical approach to help users find usages of APIs given only simple text phrases, starting with a greedy subgraph search algorithm which returned a connected subgraph containing nodes with high semantic similarity to the query phrases. In [19], Dojchinovski et al. presented a semantics based model of a Web API directory graph that captured relationships among APIs, mashups, developers, and categories, based on which, a novel API selection method were proposed to allow users to get more

TABLE 4
Top 5 results by PW and SMS for two queries, along with API description, category and provider rank

PW				SMS			
API Name	**Extracts from Description**	**Category**	**Rank**	**API Name**	**Extracts from Description**	**Category**	**Rank**
Query: athlete							
Athlinks	results of endurance races, athlete profiles	Sports	76106	**Active.com**	running events, tennis tournaments high school sports ranking data	Sports	4690
Online Courier Quotes	Australian postage, courier and shipping rate	Shipping	7928056	**TrainingPeaks**	training and nutrition software, athletes and coaches	Sports	28401
TrueSample	validation of sample population, survey data	Social	3427166	**MapMyRun**	Plan stride, learn from route, share your journey	Sports	12113
CoPub	search and text mining, MedLine database	Science	1111535	HealthTap	online health network, medical advice, answers to health questions	Medical	6372
Qualtrics	integrate survey functionality, questions to ask, response	Q&A	1447	**Life Fitness**	exercise equipment, fitness, gyms	Sports	114977
Query: school							
JamBase	live music and concert information	Music	23514	**Education.com**	school data, academic performance, student demographics	Education	6916
A State of Trance	radio show, plays trance and progressive rock music	Music	4421509	**GreatSchools**	find nearby U.S. schools, see information about school	Education	10608
SecondHandSongs	reference database, artists, songs	Music	282017	**Schoology**	classroom and education management	Education	5410
Education.com	school data, academic performance, student demographics	Education	6916	**Inside Higher Ed**	online news, job listing provider for higher education	Education	17161
SpeakerRate	event organizers, attendees, speakers	Events	1125256	**Edmodo**	social network for educators and students	Education	2957

control over their preferences and recommended APIs while the information about their links with other objects and preferences can be exploited.

CF-based approaches: Approaches in this category employ collaborative filtering technique which has been widely adopted in recommender systems, for API or mashup mining. Zhong et al. [20] proposed a time-aware API recommendation approach for mashup creation with the combination of service evolution, collaborative filtering and content matching. A user-group item-based collaborative filtering method was proposed for personalized Open API recommendation in clouds by [21]. Zheng et al. [22] provided a QoS value prediction approach for service by combining the traditional user-based and item-based collaborative filtering methods. Users can discover suitable services according QoS information from similar users without service invocations. [23] et al. proposed a hybrid collaborative filtering for service recommendation and an inverse consumer frequency based user collaborative filtering for consumer recommendation. Sun et al. [24] designed a new similarity measure for similarity computation between services and proposed a normal recovery collaborative filtering approach to recommend services for consumers.

QoS-based approaches: QoS for Web services refers to various nonfunctional features such as availability, response time, throughout, and reliability. The goal of QoS-based approach is discovering services that meet the nonfunctional requirements of users. The service selection model proposed by Yu et al. [5] defined multiple QoS criteria and took global constraints into consideration. Zhang et al. [6] presented WSExpress considering QoS characteristics of Web services for service discovery. In [25], Xu et al. proposed reputation-enhanced QoS-based model in which a discovery agent facilitated QoS-based service discovery using the reputation scores in a service matching, ranking and selection algorithm. Kritikos et al. [26] analyzed all necessary requirements for an accurate, effective, and user-assisting QoS-based service matchmaking and selection process.

Social Network-based approaches: Torres et al. [7] constructed collaboration network of APIs and proposed a API discovery approach based on social information. In [8], Cao et al. presented an approach to recommend mashup services, which considered users' interests mined from their mashup usage history and the social network based on relationships of mashup services, Web APIs and tags. An iterative approach to find APIs for building mashups was presented in [27], with the combination of semantic and social information obtained from the built collaborative social network of APIs. Xu [28] et al. proposed a social-aware service recommendation approach that utilized multi-dimensional social relationships among potential users, topics, mashups, and services.

Others: We also list some work about service discovery not involved by those types above. Different with these service (API and mashup) recommendation approach that implemented in the same library, Zheng et al. [29] tried to recommend related APIs of different libraries through mining search results of Web search engines. Bianchini et al. [30] provided a multi-perspective based API search framework for enterprise mashup design, in which a new perspective of the web designers' experience is used together with other Web API search techniques, relying on classification features, and technical features, such as the API protocols and data formats. In [31], Gomadam et al. proposed a method for searching and ranking Web APIs that adopts document classification and faceted search, and the serviut score is used to rank APIs based on their utilization and popularity.

Most of these service discovery approaches exploit information that services originally contain or offered by service providers: functional similarity (semantic information), nonfunctional feature (QoS), or social network generated by the collaboration or relationship among services, their compositions and consumers. With the generally increasing functional and nonfunctional requirements of consumers, the performance of service discovery might be limited by the lack of richness and diversity in existing available information. Thus, we try to introduce social media information to facilitate service discovery process. In [14], we proposed a crowdsourcing based approach that leverages Twitter lists to infer semantic information for service search. Based on the previous work, this paper incorporates more factors of social media and presents a weight learning algorithm to get an optimal combination of all the social factors.

7 Conclusion

With the explosive growth of services, including Web services, cloud services, APIs and mashups, discovering the appropriate services for consumers is becoming an imperative issue. It is feasible to incorporate social media information to facilitate the quality of service discovery, since the presence of services in social network is rapidly increasing and direct feedback information of crowds from social network is more valuable for the research of service discovery.

In this paper, we propose a novel framework for service discovery called SMS, which exploits social media information collected from Twitter. We propose different methods to measure four social factors. For the functional social factor, LSI model is applied to mine semantic information of services from meta-data of lists that contains them. For the rest three nonfunctional social factors, we use normalization method and decay function over list number to measure them. We present a weight learning algorithm to learn an optimal combination of the measured social factors. The target query-service matching function is assumed as a linear combination of the multiple social factors. Then we learn the optimal weight assignment based on the generated collection of training triplets. Finally we use the learned weights to retrieve services for a coming query.

In the experiment part, we report the evaluation results on a real-world dataset crawled from Twitter. For evaluation, 52 sample queries of 12 categories are selected. We design a service category based binary judgment to measure the accuracy of our proposed SMS. In addition, we adopt an external data source about rank of service providers to evaluate the influence of popularity mined from social media. The evaluation results respectively demonstrate the effectiveness of social factors and the proposed weight learning algorithm from different aspects.

In the future, we plan to (i) explore more factors of services from social media to further improve the performance of service discovery; (ii) apply our framework to address other challenging issues in service discovery, such as similar service retrieval, service recommendation for composition.

Acknowledgments

This research was partially supported by the Natural Science Foundation of China under grant of No. 61379119, National Science and Technology Supporting Program of China under grant of No. 2015BAH18F02, the Fundamental Research Funds for the Central Universities under grant of No. ZH2016007.

References

[1] J. Wu, L. Chen, Z. Zheng, M. R. Lyu, and Z. Wu, "Clustering web services to facilitate service discovery," *Knowledge and information systems*, vol. 38, no. 1, pp. 207–229, 2014.

[2] Y.-J. Lee and J.-S. Kim, "Automatic web api composition for semantic data mashups," in *Proceedings of the 4th IEEE International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2012, pp. 953–957.

[3] S.-Y. Lin, C.-H. Lai, C.-H. Wu, and C.-C. Lo, "A trustworthy qos-based collaborative filtering approach for web service discovery," *Journal of Systems and Software*, vol. 93, pp. 217–228, 2014.

[4] N. N. Chan, W. Gaaloul, and S. Tata, "A recommender system based on historical usage data for web service discovery," *Service Oriented Computing and Applications*, vol. 6, no. 1, pp. 51–63, 2012.

[5] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, p. 6, 2007.

[6] Y. Zhang, Z. Zheng, and M. R. Lyu, "Wsexpress: A qos-aware search engine for web services," in *Proceedings of the 17th IEEE International Conference on Web Services (ICWS)*. IEEE, 2010, pp. 91–98.

[7] R. Torres, B. Tapia, and H. Astudillo, "Improving web api discovery by leveraging social information," in *Proceedings of the 18th IEEE International Conference on Web Services (ICWS)*. IEEE, 2011, pp. 744–745.

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

[9] L. Chen, Q. Yu, S. Y. Philip, and J. Wu, "Ws-hfs: A heterogeneous feature selection framework for web services mining," in *2015 IEEE International Conference on Web Services (ICWS)*. IEEE, 2015, pp. 193–200.

[10] G. Liu, A. Liu, Y. Wang, and L. Li, "An efficient multiple trust paths finding algorithm for trustworthy service provider selection in real-time online social network environments," in *2014 IEEE International Conference on Web Services (ICWS)*. IEEE, 2014, pp. 121–128.

[11] S. Ghosh, N. Sharma, F. Benevenuto, N. Ganguly, and K. Gummadi, "Cognos: crowdsourcing search for topic experts in microblogs," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2012, pp. 575–590.

[12] P. Lamere, "Social tagging and music information retrieval," *Journal of New Music Research*, vol. 37, no. 2, pp. 101–114, 2008.

[13] N. Kallen, "Twitter blog: Soon to launch: Lists," 2009.

[14] T. Liang, L. Chen, H. Ying, Z. Zheng, and J. Wu, "Crowdsourcing based api search via leveraging twitter lists information," in *Proceedings of IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015, pp. 1540–1547.

[15] R. Pochampally and V. Varma, "User context as a source of topic retrieval in twitter," in *Workshop on Enriching Information Retrieval (with ACM SIGIR)*, 2011, pp. 1–3.

[16] B. Rosario, "Latent semantic indexing: An overview," *Techn. rep. INFOSYS*, vol. 240, 2000.

[17] L. Chen, L. Hu, Z. Zheng, J. Wu, J. Yin, Y. Li, and S. Deng, "Wtcluster: Utilizing tags for web services clustering," in *Service-Oriented Computing*. Springer, 2011, pp. 204–218.

[18] W.-K. Chan, H. Cheng, and D. Lo, "Searching connected api subgraph via text phrases," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. ACM, 2012, p. 10.

[19] M. Dojchinovski, J. Kuchar, T. Vitvar, and M. Zaremba, "Personalised graph-based selection of web apis," in *The Semantic Web–ISWC 2012*. Springer, 2012, pp. 34–48.

[20] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation in an evolving service ecosystem," in *Proceedings of the 21st IEEE International Conference on Web Services (ICWS)*. IEEE, 2014, pp. 25–32.

[21] H. Sun, Z. Zheng, J. Chen, W. Pan, C. Liu, and W. Ma, "Personalized open api recommendation in clouds via item-based collaborative filtering," in *Proceedings of the Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2011, pp. 237–244.

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

[23] J. Cao, Z. Wu, Y. Wang, and Y. Zhuang, "Hybrid collaborative filtering algorithm for bidirectional web service recommendation," *Knowledge and information systems*, vol. 36, no. 3, pp. 607–627, 2013.

[24] H. Sun, Z. Zheng, J. Chen, and M. R. Lyu, "Personalized web service recommendation via normal recovery collaborative filtering," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 573–579, 2013.

[25] Z. Xu, P. Martin, W. Powley, and F. Zulkernine, "Reputation-enhanced qos-based web services discovery," in *Proceedings of the 14th IEEE International Conference on Web Services (ICWS)*. IEEE, 2007, pp. 249–256.

[26] K. Kritikos and D. Plexousakis, "Requirements for qos-based web service description and discovery," *IEEE Transactions on Services Computing*, vol. 2, no. 4, pp. 320–337, 2009.

[27] B. Tapia, R. Torres, and H. Astudillo, "Simplifying mashup component selection with a combined similarity-and social-based technique," in *Proceedings of the 5th International Workshop on Web APIs and Service Mashups*. ACM, 2011, p. 8.

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

[29] W. Zheng, Q. Zhang, and M. Lyu, "Cross-library api recommendation using web search engines," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. ACM, 2011, pp. 480–483.

[30] D. Bianchini, V. De Antonellis, and M. Melchiori, "A multi-perspective framework for web api search in enterprise mashup design," in *Advanced Information Systems Engineering*. Springer, 2013, pp. 353–368.

[31] K. Gomadam, A. Ranabahu, M. Nagarajan, A. P. Sheth, and K. Verma, "A faceted classification based approach to search and rank web apis," in *Proceedings of the 15th IEEE International Conference on Web Services (ICWS)*. IEEE, 2008, pp. 177–184.

Tingting Liang received the B.S degree in the school of science, Jiangnan University, Wuxi, China, in 2013. She is currently working toward the Ph.D degree in the College of Computer Science, Zhejiang University. Her publications have appeared in some well-known conference proceedings, e.g, ICSOC, ICWS, ICDM, etc. Her research interests include service computing, data mining and recommender system.

Liang Chen received the Ph.D. and B.S. degree in computer science from Zhejiang University, Hangzhou, China in 2009 and 2015, respectively. He is currently a Postdoc Research Fellow in Advanced Analytical Institute, University of Sydney Technology, Australia. Dr. Chen has served as the workshop co-chair in many conferences, such as CIKM, BigData, PAKDD, etc. His publications have appeared in many well-known conference proceedings and international journals, e.g, WWW, ICSE, ICDM, CIKM, ICSOC, ICWS, TSC, TSMC, WWWJ, etc. His research interests include service computing, recommender system, and data mining.

Jian Wu received his B.S. and Ph.D. Degrees in computer science from Zhejiang University, Hangzhou, China, in 1998 and 2004, respectively. He is currently a full professor at the College of Computer Science, Zhejiang University, and visiting professor at University of Illinois at Urbana-Champaign. His research interests include service computing and data mining. He is the recipient of the second grade prize of the National Science Progress Award. He is currently leading some research projects supported by National Natural Science Foundation of China and National High-tech R&D Program of China (863 Program).

Guandong Xu received the Ph.D. degree in Computer Science from Victoria University, Australia. He is currently a Senior Lecturer of School of Software and the Advanced Analytics Institute at University of Technology Sydney. He has authored three monographs with the Springer and the CRC Press, and 100+ journal and conference papers. His current research interests include data science and data analytics, Web data mining, behaviour analytics, recommender systems, predictive analytics, social network analysis. Dr. Xu has served in the Editorial Board or as Guest Editor for several international journals. He is the Assistant Editor-in-Chief of the World Wide Web Journal.

Zhaohui Wu received the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 1993. From 1991 to 1993, he was a joint Ph.D. student in the area of knowledge representation and expert system with the German Research Center for Artificial Intelligence, Kaiserslautern, Germany. He is currently a Professor at the College of Computer Science, the Director of the Institute of Computer System and Architecture, Zhejiang University, and the leader of Modern Service Industry expert group in the Ministry of Science and Technology, China. His research interests include intelligent transportation systems, distributed artificial intelligence, semantic grid, and ubiquitous embedded systems. He is the author of four books and more than 100 referred papers. Dr. Wu is a Standing Council Member of China Computer Federation (CCF), Beijing. Since June 2005, he has been the Vice Chairman of the CCF Pervasive Computing Committee. He is also on the editorial boards of several journals and has served as a Program Chair for various international conferences.