Knowl Inf Syst (2019) 58:669–700
https://doi.org/10.1007/s10115-018-1171-4

REGULAR PAPER

Mining and clustering service goals for RESTful service discovery

Neng Zhang[1] · Jian Wang[1] · Keqing He[1] · Zheng Li[2] · Yiwang Huang[3]

Received: 13 October 2016 / Revised: 28 October 2017 / Accepted: 30 January 2018 /
Published online: 20 February 2018

Abstract In recent years, RESTful services that are mainly described using short texts are becoming increasingly popular. The keyword-based discovery technology adopted by existing service registries usually suffers from low recall and is insufficient to retrieve accurate RESTful services according to users' functional goals. Moreover, it is often difficult for users to specify queries that can precisely represent their requirements due to the lack of knowledge on their desired service functionalities. Toward these issues, we propose a RESTful service discovery approach by leveraging service goal (i.e., service functionality) knowledge mined from services' textual descriptions. The approach first groups the available services into clusters using probabilistic topic models. Then, service goals are extracted from the textual descriptions of services and also clustered based on the topic modeling results of services. Based on service goal clusters, we design a mechanism to recommend semantically relevant service goals to help users refine their initial queries. Relevant services are retrieved by matching user selected service goals with those of candidate services. To improve the recall of the goal-based service discovery approach, we further propose a hybrid approach by integrating it with two existing service discovery approaches. A series of experiments conducted on

✉ Jian Wang
jianwang@whu.edu.cn

Neng Zhang
nengzhang@whu.edu.cn

Keqing He
hekeqing@whu.edu.cn

Zheng Li
zhengli_hope@whu.edu.cn

Yiwang Huang
huangyw@whu.edu.cn

1 State Key Lab of Software Engineering, Computer School, Wuhan University, Wuhan, China

2 School of Computer and Information Engineering, Henan University, Kaifeng, China

3 School of Data Science, Tongren University, Tongren, China

real-world services crawled from a publicly accessible registry, ProgrammableWeb, demonstrate the effectiveness of the proposed approaches.

Keywords Service discovery · RESTful service · Service goal · Topic model · Clustering · Recommendation

1 Introduction

Web services are basic constructs of service-oriented computing (SOC), which encapsulate application functionalities and can be accessed over a network [1]. The wide adoption of SOC leads to the rapid growth of services on the Internet. Most of the existing services can be divided into two mainstream types: Simple Object Access Protocol (SOAP)-based services and REpresentational State Transfer (REST)-based service [also referred to as RESTful services or Web application programming interfaces (APIs)]. The "classical" SOAP-based services must be described using the standard Web Service Description Language (WSDL) which packs a lot of power into describing various aspects of a service. Such verbosity is often undesired and leaves a high entry-barrier for developers [2]. In recent years, an increasing amount of attentions have been shifted to the RESTful services, a simpler service architecture designed for solving the complexity of developing and describing SOAP-based services. The RESTful architectural principles [3] are characterized by their natural suitability for the Web, relying almost entirely on the use of URIs for resource identification and interaction, and HTTP for message transmission. Many companies, e.g., Google, Facebook, and Twitter, have offered easy-to-use, public APIs that provide access to some of the resources they hold, thus enabling third-parties to combine heterogeneous data from diverse services [4]. According to recent statistics, more than 14,800 services have been registered at ProgrammableWeb[1] (PW) as of March 21, 2016, and nearly 62% of the services are RESTful services while only 16% are SOAP-based services. Although RESTful services can be described by several XML-based languages such as Web Application Description Language (WADL) and WSDL 2.0, providers often use simple natural language texts on webpages to explain their APIs [5]. As an example, Fig. 1 shows a part of the descriptive data of a PW API, Cleartrip Hotel.

The publicly registered services can greatly improve the efficiency and quality of software development [6]. A large number of service discovery approaches have been proposed to find reusable services according to user queries. However, it remains several challenging issues. On one hand, most of the approaches [7–18,20–23] are designed for SOAP-based services, while only limited attentions [2,6,24–28] have been given to RESTful services. To the best of our knowledge, existing RESTful service discovery approaches are mainly based on hRESTS [29], a machine-readable microformat for RESTful services. The service documents required for hRESTS construction are often difficult to collect for services registered at public platforms such as PW. As a matter of fact, current service registries still essentially adopt the keyword-based technology for service search, which usually suffers from low recall [19,30]. Moreover, many activities on the Web such as Web search are driven by users' high-level goals [31], e.g., "book hotels" and "search playlist," in order to obtain accurate search results. Because the keyword-based technology is mainly based on the bag-of-words model that cannot capture important relationships between the words [32] in service descriptions and user queries, it is insufficient to retrieve accurate services for users' functional goals.

[1] http://www.programmableweb.com/.

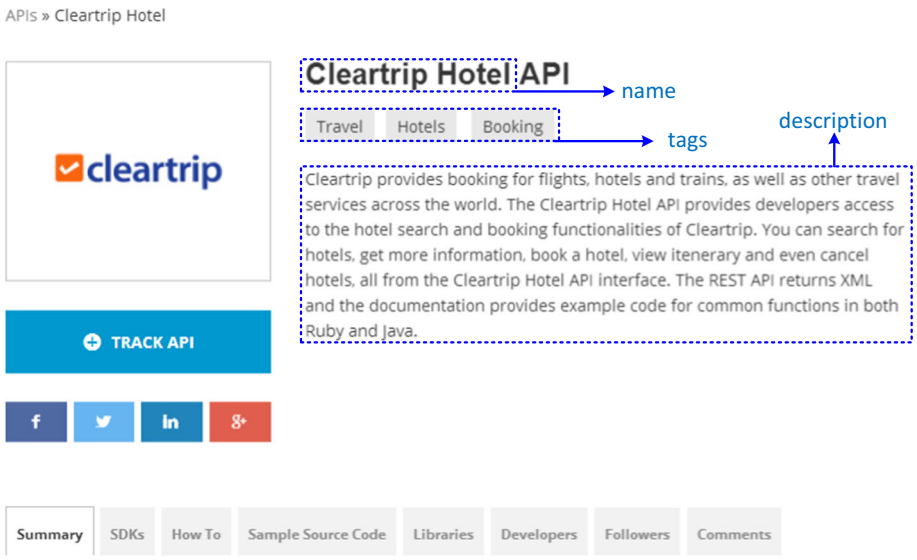


Fig. 1 Example Web API in PW

On the other hand, apart from the service search technology, the quality of user queries is another factor that will affect the results of service discovery. A query that precisely represents user requirements will help retrieve accurate services. Unfortunately, it is often difficult for users to specify high-quality queries due to the lack of knowledge on their desired service functionalities. For example, suppose someone wants to query the accommodation information in some place, there can be many semantically similar functionalities offered by various services, such as "get hotel," "find hostel," "retrieve accommodation," and "search apartment." It is non-trivial for the user to come up with all these similar functionalities, which will result in the missing of many relevant services. Several query expansion mechanisms [10–13] have been proposed to address the incompleteness issue of user queries. Their basic idea is to enhance the queries by using relevant concepts or terms retrieved from lexical databases (e.g., WordNet[2] [33]) or domain ontologies. Although these mechanisms can help retrieve more services that are relevant to user queries, they may also introduce many irrelevant services due to the fact that some expanded concepts or terms might not be desired by users.

To address the above issues, we propose an approach to support the discovery of RESTful services by leveraging service goal (i.e., service functionality) knowledge mined from textual descriptions of services. In our approach, the available services are grouped into clusters using probabilistic topic models, e.g., Latent Dirichlet Allocation (LDA) [34]. Service goals extracted from the services' textual descriptions are also clustered based on the topic modeling results of services. These clustering results are used to facilitate service discovery for user queries. Specifically, service clusters are used to reduce the service search space. Based on the service goal clusters, we design a mechanism to help users refine their initial queries by recommending semantically similar service goals based on the service goal clusters. Relevant services are finally retrieved by matching user selected service goals with those of candidate services in the reduced search space. To improve the recall of the goal-based service discovery approach (referred to as GoSD), we further propose a hybrid approach by

[2] http://wordnet.princeton.edu/.

integrating it with two existing service discovery approaches: the keyword-based approach and a probabilistic semantic approach based on LDA. We conducted a series of experiments on a real-world service dataset crawled from PW to evaluate the proposed approaches. The results demonstrate that GoSD can retrieve accurate services and that the limited recall of GoSD can be improved by the hybrid approach.

The remainder of this paper is organized as follows. Section 2 discusses some related works. Section 3 gives an overall framework of the proposed service discovery approach. The framework comprises two major modules: offline data process and online service discovery, which are described in Sects. 4 and 5, respectively. Section 6 presents the experiments and analysis. Section 7 concludes the paper.

2 Related works

2.1 Service discovery

As a fundamental issue of SOC, service discovery has attracted a lot of attentions. Many research works on service discovery perform profile-based service signature (e.g., inputs, outputs, preconditions, and effects) matching or exploit the structure of WSDL document to find similar services [6]. For example, Wang et al. [7] described a service discovery approach that combines traditional information retrieval techniques [e.g., vector space model (VSM) and term frequency—inverse document frequency (TF-IDF)] with a structure matching algorithm for WSDL. In [8], Plebani et al. evaluated the similarity between service interfaces in WSDL by considering the structures of service interfaces and the terms used inside them. Liu et al. [9] leveraged bipartite graphs to calculate similarity between services. Moreover, many classification techniques and clustering techniques have been applied in service discovery [14–19,30,35–37]. Service classification uses supervised learning techniques (e.g., decision tree and support vector machine) to classify services to predefined categories, while service clustering leverages unsupervised learning techniques (e.g., K-Means and topic models) to group similar services on the basis of textual features. Both these techniques can help reduce the search space during service discovery and thus boost the discovery performance. However, these syntactic approaches are insufficient to find semantically similar services for user queries. In addition, they mainly consider the discovery of WSDL-based SOAP services, while neglect the discovery of RESTful services described in short texts.

A number of semantic approaches [20–23] have been proposed to enable automatic service discovery. The key of most semantic approaches is to describe services and queries using ontology-based semantic description languages (e.g., OWL-S, WSMO, and SAWSDL) and develop logical reasoning algorithms to retrieve similar services. For example, Klusch et al. [20] proposed an approach for matching services described using OWL-S, which combined logical reasoning and syntactic concept similarity computation. In [21], a hybrid service matchmaker was designed for SAWSDL, which determined three types of similarities: logic-based similarity, text-based similarity, and structural similarity. These approaches have shown to be effective because of the accurate descriptions of services and queries. However, they are limited by requiring considerable efforts on correctly specifying and managing ontologies, and annotating services and queries [38].

To the best of our knowledge, only a few works [2,6,24–28] have reported the discovery of RESTful services. Most of these approaches are based on hRESTS [29] and its semantic extensions like SA-REST and MicroWSMO. They are limited by the fact that the HTML

service documents required for hRESTS construction that contain operation descriptions (e.g., addresses, HTTP methods, and input/output data formats) are usually difficult to collect for services at public registries. For example, the operation descriptions of RESTful services registered at PW are generally hidden in services' homepages. It requires considerably more efforts to collect.

To sum up, existing works have their respective limitations in dealing with RESTful service discovery. In this paper, we propose an approach for RESTful service discovery based on service goal knowledge mined from services' textual descriptions.

2.2 Service query expansion

In order to retrieve more services that are relevant to user queries, several query expansion mechanisms [10–13] have been proposed to enhance queries by using relevant concepts or terms extracted from domain ontologies or lexical databases. For example, Kokash et al. [10] addressed the inadequacy of VSM by expanding both WSDL descriptions and queries with synonyms in WordNet. Paliwal et al. [12] proposed to enhance queries by utilizing relevant concepts extracted from domain ontologies. Compared with these works, we propose a novel query expansion mechanism by leveraging service goal knowledge. To ensure that the expanded queries can precisely reflect user requirements, users are requested to participate in the query expansion process. Similar to our work, Jung et al. [28] proposed a natural language processing (NLP)-based approach to mine functional goals from services' textual descriptions and a goal suggestion mechanism for user queries. The mechanism was based on the measurement of Jaccard similarity between functional goals and user queries, and was thus unable to suggest functional goals that are semantically similar to queries. In [6], we proposed a NLP-based approach to extract service goals from textual descriptions of services, as well as a goal recommendation mechanism using the k-Nearest Neighbors algorithm (or k-NN for short). Compared with these studies, this work measures semantic similarities between service goals and user queries; therefore, our goal recommendation mechanism can recommend semantically similar service goals for queries.

The main contributions of this work lie in three aspects. Firstly, we propose an approach to group services into clusters using topic models and cluster the extracted service goals based on the topic model trained for services. Secondly, a new mechanism is designed to recommend similar service goals for user queries by leveraging service goal clusters, whereas in [6] goals are recommended based on the retrieved similar services using k-NN. Thirdly, we propose an approach that can retrieve accurate services according to user selected service goals from the recommendations.

3 Overall service discovery framework

As illustrated in Fig. 2, the overall framework of the proposed RESTful service discovery approach consists of two major modules: offline data process and online service discovery. In the offline module, three tasks are performed over available services. Firstly, services are modeled using topic models, e.g., LDA, and grouped into clusters based on the generated service-topic distributions. Secondly, service goals are extracted from the services' textual descriptions using the service goal extraction approach proposed in our prior work [6]. A service-service goal assignment matrix is built to record the service goals of each service. Thirdly, the extracted service goals are folded into the topic model trained for services (using a technique called *Folding-in* [15,16,39]) and clustered based on their topic distributions.

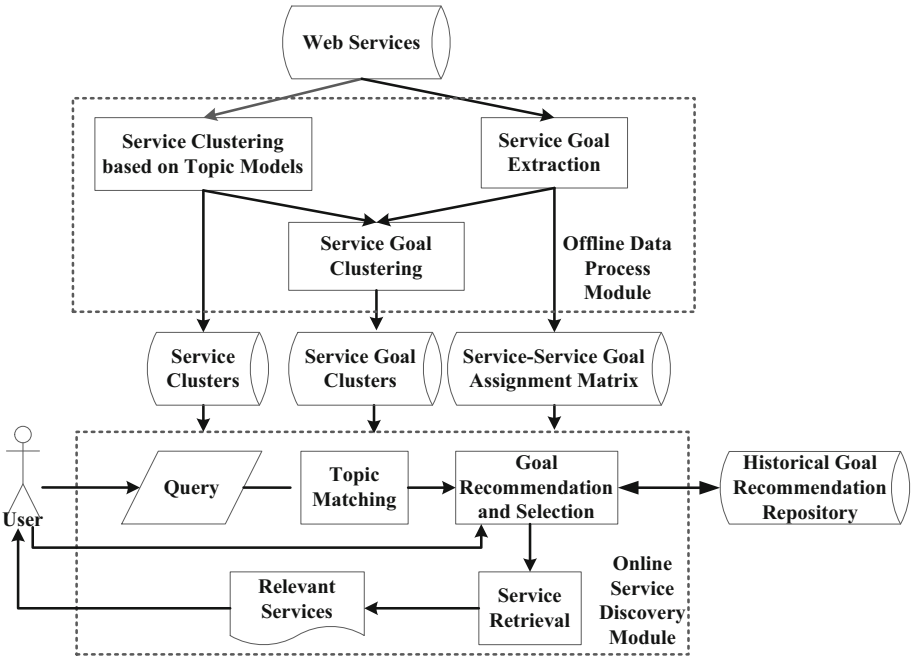


Fig. 2 Overall service discovery framework

In the online module, the service clusters, service-service goal assignment matrix, and service goal clusters produced in the offline module are leveraged to facilitate the service discovery for user queries. When a user query is sent to the service search engine, the first step is to determine a set of topics that are relevant to the query. The identified relevant topics are used in two subsequent steps, i.e., goal recommendation and service retrieval. As the initial query might be incomplete due to the user's lack of knowledge on his/her desired service functionalities. In the goal recommendation step, service goals assigned to the clusters that correspond to the relevant topics are recommended to help the user refine the initial query. To facilitate the user selection of appropriate service goals from the recommendations, we propose a method to sort the recommended goals by measuring their semantic similarities to the query. Furthermore, a historical goal recommendation repository is maintained to record service goals selected by users for their queries. The repository is leveraged to produce better service goal recommendations for future user queries. In the final step, relevant services are retrieved by matching the user selected service goals with those of services. To improve the efficiency, the scope of service search is restricted to the service clusters that correspond to the relevant topics of the query.

4 Offline data process

In this section, we describe the three offline steps of the proposed service discovery framework, namely the service clustering based on topic models, service goal extraction, and service goal clustering.

4.1 Service clustering based on topic models

Comparing a user query to all services in a large repository will be time-consuming and computationally expensive [16,18]. Many clustering techniques have been adopted to address this issue by grouping similar services together [14–19,37]. By organizing services into clusters in advance, the service search space can be restricted to the clusters that are relevant to a user query, which will largely reduce the amount of comparisons and improve the efficiency.

Recently, topic models, e.g., LDA, have been widely applied in service clustering [15–18,37]. They can learn a set of latent topics to capture the underlying semantics of services [18]. After the topic modeling process, each service can be represented as a vector in terms of the learned topics. This can reduce the data dimensionality as the size of the topic space, which is typically much smaller than the size of the word vocabulary [30]. Moreover, semantic similarities between services and user queries (after inferring their topic distributions) can be approximately measured based on their topic vector representations [39].

We use LDA to learn latent topics from the words describing services and then group the services into clusters based on the learned topics. We first take the following steps to preprocess the descriptive data of each service:

1. *Tokenization* We perform tokenization over the descriptive data to produce the original content vector by using the NLTK[3] [40] toolkit.
2. *Word form normalization* Stemming and lemmatization are two widely applied word form normalization techniques [41], which can transform the inflected words into their stems and lemmas, respectively. For a word, the stem is a part of the word to which affixes can be attached, and the lemma is the basic form of the word. For example, the stem and lemma of "manage" are "manag" and "manage," respectively. There are some differences between these two techniques. On the one hand, stemming can deal with more morphological variants of words, for example, the noun forms of verbs, such as "booking" and "management." On the other hand, the semantic similarity between lemmas can be measured using semantic networks or lexicons like WordNet, BabelNet, and VerbNet. WordNet is viewed as the most popular computational lexicon of the English language and many semantic similarity algorithms based on WordNet have been proposed; BabelNet is a multilingual lexicalized semantic network created by linking Wikipedia to WordNet; and VerbNet is a comprehensive verb lexicon that incorporates both semantic and syntactic information. Considering the popularity and coverage of WordNet, we choose it as the basic lexicon used in our approach. The algorithms of stemming and lemmatization have been included in NLTK. In this work, stemming is used to prepare data for LDA, and lemmatization is used to refine the service goals extracted from services' textual descriptions, such that in the goal recommendation step, the semantic similarity between service goals and user queries (after processed by lemmatization) can be measured using WordNet.
3. *Stop word removal* Stop words such as "the," "other," and "of" are removed.

Next, the preprocessed services are used as training data of LDA. LDA is a generative probabilistic model for a corpus (e.g., a collection of documents). The basic idea is that documents are random mixtures over a set of latent topics, where each topic is characterized by a distribution over words [34]. The generative process for a set of services S given T latent topics is as follows:

[3] http://www.nltk.org/.

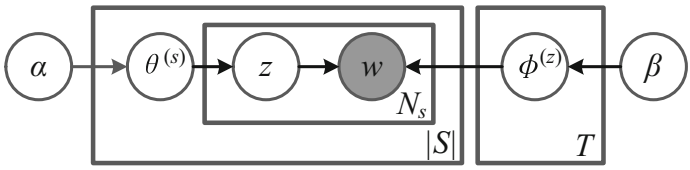


Fig. 3 Graphic model of LDA for a set of services S. $|S|$ is the number of services, N_s is the number of words in service s, and T is the number of latent topics

1. Pick a multinomial distribution $\phi^{(z)}$ for each topic $z \in \{1, \ldots, T\}$ from a Dirichlet distribution with parameter β;
2. Pick a multinomial distribution $\theta^{(s)}$ for each service $s \in S$ from a Dirichlet distribution with parameter α;
3. For each word w in service s:
   (a) Pick a topic z from the multinomial distribution $\theta^{(s)}$;
   (b) Pick w from the multinomial distribution $\phi^{(z)}$.

The generative process described above can be illustrated using a graphic model, as shown in Fig. 3. Fitting the LDA model involves estimating both the topic distribution of each service s, $\theta^{(s)}$, and the word distribution of each topic z, $\phi^{(z)}$ [42]. We implement LDA based on the JGibbLDA[4] toolkit that uses Gibbs sampling for parameter estimation. The Gibbs sampling algorithm starts with random topic assignments of all words in the services, $\mathbf{w} = \{w_1, w_2, \ldots, w_N\}$ (each word w_i belongs to some service s_i). In a Gibbs sample, the topic of each word w_i is resampled using

$$p\left(z_i = t | \mathbf{z}_{-i}, \mathbf{w}\right) \propto \frac{n_{-i,t}^{(w_i)} + \beta}{n_{-i,t}^{(\cdot)} + |V|\,\beta} \frac{n_{-i,t}^{(s_i)} + \alpha}{n_{-i.}^{(s_i)} + T\alpha}, \tag{1}$$

where $\mathbf{z}_{-i}$ denotes the topic assignments of all words except the current word w_i; $n_{-i,t}^{(w_i)}$ is the number of instances of word w_i assigned to topic t, not including the current word; V is the set of distinct words in $\mathbf{w}$; $n_{-i,t}^{(s_i)}$ is the number of words in service s_i assigned to topic t, excluding the current word; $n_{-i,t}^{(\cdot)}$ is the number of all words in $\mathbf{w}$ assigned to topic t, except the current word; and $n_{-i.}^{(s_i)}$ is the number of words in service s_i (assigned to all topics), excluding the current word. Intuitively, the first ratio expresses the probability of word w_i under topic t, and the second ratio expresses the probability of topic t in service s_i [43].

After the "burn-in" period (i.e., discarding an initial set of samples to avoid starting biases) [39,44], the topic-word distribution $\phi^{(z)}$ and the service-topic distribution $\theta^{(s)}$ can be estimated given a Gibbs sample as

$$\hat{\phi}_w^{(z)} = \frac{n_z^{(w)} + \beta}{n_z^{(\cdot)} + |V|\,\beta}, \tag{2}$$

$$\hat{\theta}_z^{(s)} = \frac{n_z^{(s)} + \alpha}{n_\cdot^{(s)} + T\alpha}, \tag{3}$$

where $n_z^{(w)}$ is the number of word $w \in V$ assigned to topic z; $n_z^{(\cdot)}$ is the number of all words assigned to topic z; $n_z^{(s)}$ is the number of words in service s assigned to topic z; and $n_\cdot^{(s)}$ is the number of words in service s (assigned to all topics).

[4] http://jgibblda.sourceforge.net/.

It is worth noting that the number of latent topics T must be determined before training. The study in [43] shows that as T increases, the LDA model can more accurately fit the data until an optimal point is reached and thereafter the model becomes more complex than necessary and results in overfitting which degrades the performance of the model. Several methods have been proposed to determine T that can best account for a corpus. For example, Griffiths and Styvers [43] discussed a method for estimating posterior probabilities $p(\mathbf{w}|\mathbf{z}, T)$ of the LDA models trained using a range of values for T, and the best T is chosen based on the model that leads to the maximum posterior probability. In [30,34], the best T was determined by measuring the likelihood on a held-out test set using the learned topic models.

We apply LDA to the preprocessed services. The best T is determined using the Bayesian model selection method discussed in [43]. After the topic modeling process, a set of latent topics are learned and two types of probability distributions are generated, i.e., the service-topic distributions and the topic-word distributions. The learned latent topics are used to cluster the services. We create T service clusters $\{SC_1, \ldots, SC_T\}$ (a cluster for each topic). Based on the service-topic distributions, all the services can be described in terms of a vector, e.g., $\overrightarrow{s} = (\pi_1, \pi_2, \ldots, \pi_T)$, where each dimension π_t reflects the probability of that service being generated by sampling from topic t. Each service is supposed to be assigned to the clusters corresponding to the topics that are related to it. As the topic distributions of services can be notably different, it is difficult to set a proper threshold for determining the relevant topics of all services. A simple solution is to assign each service to the cluster that corresponds to its most relevant topic [15,16], i.e., $\text{argmax}_{t\in\{1,\ldots,T\}}\pi_t$. This single cluster assignment may result in missing a number of services that are relevant to user queries in service discovery. Alternatively, each service could be assigned to multiple clusters, e.g., the clusters corresponding to the top k most relevant topics. Multiple cluster assignment will increase the service search space and help obtain more services that are relevant to user queries. However, it comes at the cost of an increased number of comparisons required for answering a query [16].

4.2 Service goal extraction

A service goal is used to exhibit the intentional functionality of a service [6,45]. According to the studies on goal modeling [31,46], we defined the service goal as a triple $sg =< sgv, sgn, sgp >$, *where sgv is a verb or verb phrase, which denotes the action of the service goal, sgn is a noun or noun phrase, which denotes the entities affected by the action, and sgp is an optional set of parameters, which denote the additional information such as how the action affects the entity, the initial or final state of the entity.*

As shown in Fig. 1, the textual description of a service generally contains several sentences. Many syntactic parsers, e.g., the Stanford Parser[5] and parsers in NLTK, have been developed to recognize sentences and determine their corresponding parse trees. According to the experimental evaluation conducted in [47], the Stanford Parser (which has also been integrated in NLTK) can achieve better performance than several existing parsers and can generate accurate analysis for most sentences encountered. In our approach, we used the Stanford Parser to parse the sentences extracted from services' textual descriptions. Grammatical information (i.e., Part-of-Speech (POS) tags [48]) and syntactic information (i.e., typed dependencies [49]) are generated for each sentence. Figure 4a depicts the POS tags and typed dependencies of an example sentence extracted from the textual description of

[5] http://nlp.stanford.edu/software/lex-parser.shtml.

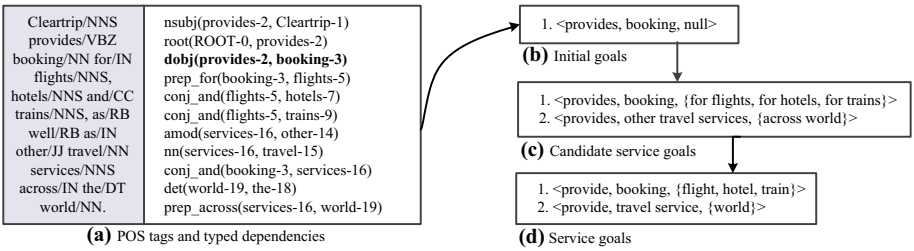


Fig. 4 Illustration of service goal extraction from an example sentence

Cleartrip Hotel: *Cleartrip provides booking for flights, hotels and trains, as well as other travel services across the world.*

It is non-trivial to mine service goals from sentences due to the diversity and complexity of sentence structures. By analyzing the typed dependencies of various sentences, we determined the function of different typed dependencies (e.g., amod, conj, dobj, and prep) for service goal extraction: some of them are used for generating the skeletons of service goals (named as **initial goals**), and some are used for enriching the initial goals and discovering potential service goals. Based on this insight, we proposed an automatic approach to extract service goals given the grammatical and syntactic information of a sentence. The approach contains three steps, which are briefly introduced as follows. Firstly, a set of initial goals are generated by employing three manually crafted typed dependency patterns: (1) dobj$(a, b) \rightarrow <a, b,$ null>, (2) nsubjpass$(a, b) \rightarrow <a, b,$ null>, and (3) prep(a, b) && $V(a) \rightarrow <a, b,$ null>. In the third pattern, $V(a)$ is to check whether a is a verb according to the POS tag of a, $POS(a)$:

$$V(a) = \begin{cases} 1, & \text{if } POS(a) \in POS_V \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

where POS_V is the set of POS tags associated with verbs, e.g., {VB, VBD, VBG, VBN, VBP, VBZ}. Moreover, a should not appear in any nsubjpass(a, b) or dobj(a, b), and if there are several prep(a, b) dependencies related to verb a, and only the first prep(a, b) behind a can be used for initial goal generation. For example, initial goal $ig =$ <provides, booking, null> is generated from the typed dependency dobj(provides-2, booking-3) in Fig. 4a. Secondly, the initial goals are extended using typed dependencies to obtain useful information of them and discover service goals that have not been generated. In our approach, six kinds of typed dependencies, including prt, prep, amod, nn, conj, and appos, are leveraged in initial goal extension. Note that prep dependencies (except those used for initial goal generation) are used to extract prepositional phrases related to *sgv* and *sgn* parts of initial goals. For example, the *sgp* part of *ig*, i.e., {for flights, for hotels, for trains}, is obtained from prep_for(booking-3, flights-5), conj_and(flights-5, hotels-7), and conj_and(flights-5, trains-9). A coordinate noun of "booking," i.e., "services," is obtained from conj_and(booking-3, services-16), which contributes to generating a new service goal, as shown in Fig. 4c. By extending the initial goals, a set of candidate service goals are obtained. Thirdly, candidate service goals are refined by performing lemmatization using the NLTK WordNet Lemmatizer and stop word removal using the built-in stop word list in NLTK.

The service goals of a service are obtained by collecting service goals extracted from all the sentences in its textual description. After performing the service goal extraction approach over all the services, a service-service goal assignment matrix $SSGAM \subseteq S \times SG$ is built to record the service goals of each service, where $S = \{s_1, s_2, \ldots, s_m\}$ is the set of services and $SG = \{sg_1, sg_2, \ldots, sg_n\}$ is the set of service goals extracted from S. Each entry $SSGAM_{ij} \in \{0, 1\}$, and $SSGAM_{ij} = 1$ means that service $s_i \in S$ contains service goal $sg_j \in SG$.

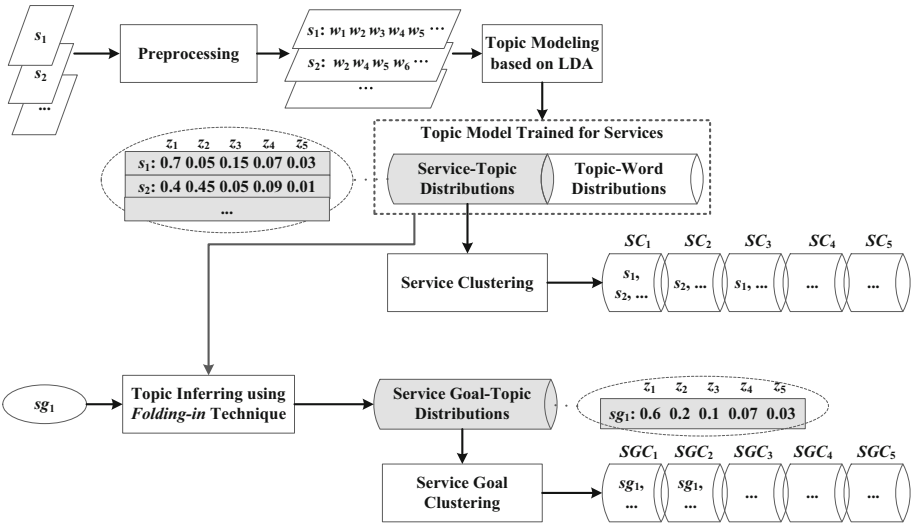


Fig. 5 Illustration of the offline data process

4.3 Service goal clustering

Through our observation, many semantically similar service goals exist in the set of extracted service goals, such as <get, real-time hotel pricing data, null>, <find, low-cost hostel, null>, and <retrieve, accommodation detail, null>. These similar service goals can be grouped together to provide comprehensive views of service functionalities.

Folding-in [15,16,39] is a technique used for fitting new documents into topic models. The key of *Folding-in* is to infer distributions over latent topics for the new documents, which can be realized by a similar procedure as in the training phase. In this work, the extracted service goals are folded into the LDA model trained for services. The words of service goals are stemmed first. Afterward, the "fold-in" process begins with randomly assigning topics to the words of service goals and continues sampling topic assignments for the service goals (with all of the topic assignments for the services fixed) [42]. Given enough iterations, the topic distribution of each service goal is estimated based on a Gibbs sample as in Eq. (3).

Based on the service goal-topic distributions, we group the service goals into clusters using a similar way as described in the service clustering. We create T service goal clusters $\{SGC_1, \ldots, SGC_T\}$ (a cluster for each topic) and cluster the service goals using top k cluster assignment (i.e., each service goal is assigned to each of the clusters that correspond to its top k most relevant topics). The service goal clusters will be used to recommend similar service goals for user queries, as described in Sect. 5.2. The top k cluster assignment with a larger k value will produce larger service goal clusters, which will increase the number of service goals to be recommended for user queries and thus may contribute to obtaining more service goals that are similar to the queries. However, it requires more time to sort the recommended goals by measuring their semantic similarities to the corresponding query.

In summary, the whole offline data process is illustrated in Fig. 5. Firstly, services in a given registry are preprocessed. Afterward, we leverage LDA to perform topic modeling, and group the services into clusters based on the learned service-topic distributions. For each service goal extracted from these services, we estimate its topic distribution using the

Folding-in technique and then assign it to service goal clusters that correspond to its highly relevant topics.

5 Online service discovery

As shown in Fig. 2, the online service discovery for a user query comprises three steps: topic matching, goal recommendation, and service retrieval, which are described in this section.

5.1 Topic matching

When a user query q is sent to the service search engine, this first step is to determine a set of topics that are relevant to q. q is first preprocessed by tokenization, stemming, and stop word removal, and then folded into the LDA model trained for services. After sampling the assignments of topics to words in q for a number of iterations, the topic distribution of q is estimated based on a Gibbs sample. The top k topics with maximum probabilities is selected as the relevant topics of q, referred to as $RelT_k(q)$. $RelT_k(q)$ will be used to recommend similar service goals for q, as described in the next section. Moreover, $RelT_k(q)$ will be used to reduce the service search space for q. Specifically, the candidate services for q, referred to as $CanS(q)$, are the services assigned to the clusters corresponding to $RelT_k(q)$, i.e., $CanS(q) = \bigcup_{t \in RelT_k(q)} SC_t$.

5.2 Goal recommendation and selection

Due to the lack of knowledge on the desired service functionalities, it is often difficult for users to specify queries that can precisely represent their requirements. If no satisfactory service is found, users have to further refine their queries and launch a next round of discovery. To address this issue, we propose a mechanism to derive user's immediate goals implied by their initial queries, which will help retrieve accurate services. For a given user query q, service goals assigned to the clusters that correspond to the relevant topics of q will be recommended, which are referred to as $RSG_T(q) = \bigcup_{t \in RelT_k(q)} SGC_t$. From the recommendations, the user can get a better understanding of the service functionalities related to q and select appropriate service goals as a refined query.

As there can be a considerable number of (e.g., more than 100) service goals recommended for user queries, as shown in Fig. 8b, it might be a time-consuming task for users to select appropriate goals from the recommendations. To facilitate the user selection, the recommended service goals can be sorted according to their similarities to the corresponding query. Several similarity measures can be used for this purpose, such as Jaccard similarity and Cosine similarity [50]. For example, Fig. 6a, b show the top ten service goals recommended for query "search playlist" sorted using Jaccard similarity and Cosine similarity, respectively. The bold ones are the relevant goals identified by a user. As can be seen from the figures, the two recommendation lists are very similar: nine recommended goals are identical and only one is different. Both of the two similarity measures fail to capture service goals that are semantically similar to the query, e.g., <access, music playlist, null>. To overcome this limitation, we propose a method to sort the recommended service goals by measuring their semantic similarities to the corresponding query. Specifically, the semantic similarity of each service goal $sg \in RSG_T(q)$ to query q is calculated as

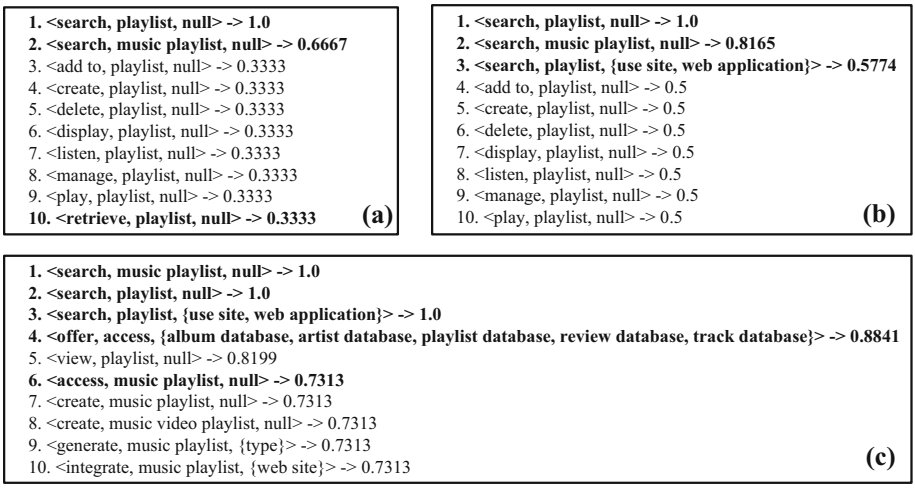


Fig. 6 Top ten service goals recommended for "search playlist" sorted using three different similarity measures: **a** Jaccard similarity, **b** Cosine similarity, and **c** the semantic similarity based on WordNet

$$GQSim(sg, q) = \frac{\sum_{w_i \in W^l(q)} \max_{w_j \in W^l(sg)} wsim(w_i, w_j)}{\left|W^l(q)\right|}, \tag{5}$$

$$wsim(w_i, w_j) = \begin{cases} 1, & \text{if } w_i \text{ equals to } w_j \\ WNSim(w_i, w_j), & \text{otherwise} \end{cases}, \tag{6}$$

where $W^l(sg)$ and $W^l(q)$ are the sets of words contained in sg and q, respectively, which are obtained using the lemmatization algorithm; $WNSim(w_i, w_j)$ is the semantic similarity between the two words w_i and w_j in WordNet. The first branch of Eq. (6) should be performed over the stems of w_i and w_j, so as to measure the similarity between some similar words, e.g., "manage" and "management," that cannot be measured using WordNet, e.g., *WNSim*("manage," "management") = 0.

Figure 6c shows the top ten service goals recommended for "search playlist" sorted according to the semantic similarities calculated using Eq. (5). We can see that some semantically similar service goals of the query are included in the top ten of the recommendation list, which is better than the recommendation lists shown in Fig. 6a, b. However, it can also be seen that some service goals that are not similar to the query, e.g., <view, playlist, null>, are sorted high in the list, while some similar goals, e.g., <retrieve, playlist, null>, are ranked behind. Based on our analysis, this issue is caused by the inappropriate similarities between some words in WordNet. The user selected service goals from the recommendations can be leveraged to improve the recommendations for future user queries. To achieve this, we design a historical goal recommendation repository to record service goals that have been selected by users for queries. The repository is defined as $HGRR = < U, Q, HSG >$, where $U = \{u_1, u_2, \ldots, u_r\}$ is a set of users, $Q = \{q_1, q_2, \ldots, q_l\}$ is a set of queries, and HSG is a $r \times l$ matrix that stores user selected service goals for queries. Generally speaking, a user may propose the same query at different time and the service goals selected at each time can be different. As illustrated in Fig. 7, each entry of the matrix HSG is such a set of couples $HSG_{ij} = \{(sg_k, n_k)\}$, where sg_k is a service goal that has been selected by user $u_i \in U$ for query $q_j \in Q$ and $n_k > 0$ is the total selection number of sg_k. In Fig. 7, "$t_1 : SG(u_i, q_j) = \{sg_1, sg_2, sg_3\}$"

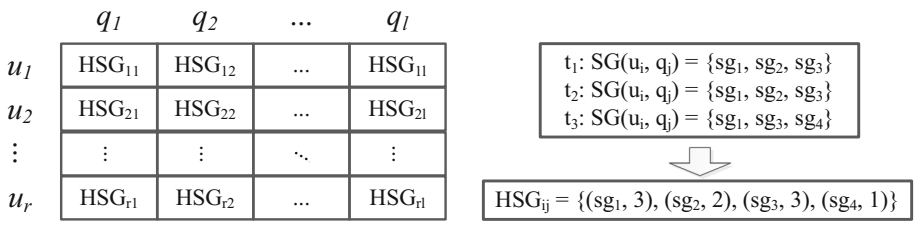


Fig. 7 Illustration of *HGRR*

means that at time t_1, user u_i launched a service discovery task with query q_j and selected three service goals $\{sg_1, sg_2, sg_3\}$ from the recommendations.

HGRR are used to produce better service goal recommendations for user queries. When a user u sends a query q to the service search engine, at the stage of goal recommendation, a set of queries in Q that are similar to q are obtained as

$$SimQ(q) = \left\{q_j | q_j \in Q \wedge QSim(q_j, q) \geq \theta\right\}, \tag{7}$$

where $QSim(q_j, q)$ is the semantic similarity of query q_j to q computed using Eq. (8), and $\theta \in [0, 1]$ is a threshold used to determine the queries that are similar to q.

$$QSim(q_j, q) = \frac{\sum_{w_i \in W^l(q)} \max_{w_j \in W^l(q_j)} wsim\left(w_i, w_j\right)}{\left|W^l(q)\right|}, \tag{8}$$

where $W^l(q_j)$ is the set of words contained in q_j, which are obtained using the lemmatization algorithm. The service goals that have been selected by users for the similar queries $SimQ(q)$ will be recommended for q, which are referred to as $RSG_H(q) = \bigcup_{u_i \in U, q_j \in SimQ(q)} \left\{sg_k | (sg_k, n_k) \in HGR_{ij}\right\}$.

It is worth noting that each query $q_j \in SimQ(q)$ may have been proposed by various users and the service goals that have been selected by the current user u will be more preferred by himself. Moreover, the service goals that have been frequently selected could be more preferred by u. Based on the analysis, the importance of each service goal $sg_k \in RSG_H(q)$ can be measured according to three factors: (1) the semantic similarity of the query for which sg_k was selected to q, (2) the user who selected sg_k, and (3) the selection number of sg_k.

For each query $q_j \in SimQ(q)$, the selection number of sg_k is divided into two parts: (1) the number of sg_k selected by u, denoted as $n_{u,q_j}(sg_k)$, and (2) the number of sg_k selected by other users except u, denoted as $n_{-u,q_j}(sg_k)$, as follows

$$n_{u,q_j}(sg_k) = n_k, (sg_k, n_k) \in HGR_{ij} \wedge u_i = u, \tag{9}$$

$$n_{-u,q_j}(sg_k) = \sum n_k, \ (sg_k, n_k) \in HGR_{ij} \wedge u_i \in U - \{u\}. \tag{10}$$

The importance of sg_k derived from the historical goal recommendation results of q_j is calculated as

$$I_H\left(q_j \rightarrow sg_k\right) = QSim(q_j, q) \times \left[\lambda \times \log_2\left(1 + n_{u,q_j}(sg_k)\right) + (1-\lambda) \times \log_2\left(1 + n_{-u,q_j}(sg_k)\right)\right], \tag{11}$$

where $\lambda \in [0, 1]$ is a parameter to balance the weights of $n_{u,q_j}(sg_k)$ and $n_{-u,q_j}(sg_k)$.

The importance of sg_k is obtained by accumulating the importance derived from the historical goal recommendation results of all queries in $SimQ(q)$:

$$I_H(sg_k) = \sum\nolimits_{q_j \in SimQ(q)} I_H\left(q_j \rightarrow sg_k\right). \tag{12}$$

$I_H(sg_k)$ is then normalized to (0, 1]:

$$I_H(sg_k) = \frac{I_H(sg_k)}{\max_{sg_i \in RSG_H(q)} I_H(sg_i)}. \tag{13}$$

The service goal list recommended for q is finally generated by incorporating the semantic similarities of services goals to q and the importance of service goals measured based on *HGRR*. The set of service goals to be recommended for q are

$$RSG(q) = RSG_T(q) \cup RSG_H(q). \tag{14}$$

The recommendation priority of each service goal $sg \in RSG(q)$ is calculated as

$$RP(sg) = \zeta \times GQSim(sg, q) + (1-\zeta) \times I_H(sg), \tag{15}$$

where $\zeta \in [0, 1]$ is a parameter to balance between the semantic similarity of sg to q and the importance of sg computed based on *HGRR*.

5.3 Service retrieval

In this step, a set of relevant services are retrieved for query q by matching the user selected service goals for q, referred to as $SG(q)$, with service goals of candidate services in $CanS(q)$. For each service $s_i \in CanS(q)$, the service goals that can satisfy any service goal in $SG(q)$ are obtained as

$$SG\left(s_i \triangleright q\right) = \left\{sg_i \in SG(s_i) | \exists sg_j \in SG(q),\, W^s(sg_j) \subseteq W^s(sg_i)\right\}, \tag{16}$$

where $SG(s_i)$ is the set of service goals of s_i, $SG(s_i) = \{sg_k | SSGAM_{ik} = 1\}$; $W^s(sg_i)$ and $W^s(sg_j)$ are the sets of words contained in service goals sg_i and sg_j, respectively, which are obtained using the stemming algorithm. The reason for stemming the words of service goals is that stems can contribute to matching morphologically similar goals, e.g., <manage, playlist, null> and <provide, playlist management, null>.

A set of relevant services are then retrieved for q as

$$RelS(q) = \{s_i | SG\left(s_i \triangleright q\right) \neq \emptyset\}. \tag{17}$$

The goal-based service discovery approach (referred to as GoSD) described above can retrieve accurate services. However, some relevant services will be missed due to two main reasons. On one hand, some useful service goals cannot be extracted from the textual descriptions of services, in particular in the following cases: (1) service goals that are expressed in noun phrases, e.g., "photo uploading," and (2) informal and complex sentences that are incorrectly parsed using the Stanford Parser. On the other hand, some similar service goals of user queries are not included or sorted high in the recommendation lists and in turn are not selected by users for service retrieval. To overcome the shortcoming of GoSD, we propose a hybrid service discovery approach (referred to as HybridSD) by integrating GoSD with two popular service

discovery approaches: the keyword-based approach and a probabilistic semantic approach based on LDA, which are described as follows.

– *Keyword-based service discovery* (referred to as KWSD): In this approach, the available services were preprocessed by tokenization, stemming, and stop word removal, and then represented as vectors using the TF-IDF technique [51]. For a given user query q, it was preprocessed using the same steps above and also represented as a TF-IDF vector (using the IDF statistical information of services). The words of q that were not contained in the services were left out. We then calculated the Cosine similarity between each service vector $\overrightarrow{s}$ and the query vector $\overrightarrow{q}$ using

$$CSim(\overrightarrow{s}, \overrightarrow{q}) = \frac{\overrightarrow{s} \cdot \overrightarrow{q}}{\| \overrightarrow{s} \| \| \overrightarrow{q} \|} = \frac{\sum_{i=1}^{v} \overrightarrow{s_i} \cdot \overrightarrow{q_i}}{\sqrt{\sum_{i=1}^{v} \overrightarrow{s_i}^2 \sum_{i=1}^{v} \overrightarrow{q_i}^2}} \tag{18}$$

where v is the dimension number of $\overrightarrow{s}$ and $\overrightarrow{q}$.

– *LDA-based service discovery* (referred to as LDASD): Topic models, e.g., LDA, have been widely used for service modeling and discovery [15–18]. We realized a service discovery approach based on LDA as follows. After modeling the available services using LDA, the topic distribution of each service was represented as a vector $\overrightarrow{s} = (\pi_1, \pi_2, \ldots, \pi_T)$, where each dimension π_t is the probability of the service being generated by sampling from topic t. For a given user query q, it was preprocessed by tokenization, stemming, and stop word removal, and folded into the LDA model trained for services. The topic distribution of q was also represented as a vector in the topic space. We then calculated the Cosine similarity between each service vector $\overrightarrow{s}$ and the query vector $\overrightarrow{q}$ using Eq. (18).

In HybridSD, for a given user query q, the user selected service goals $SG(q)$ at the stage of goal recommendation are employed to improve KWSD and LDASD. Specifically, a new query is built by collecting the words of all service goals in $SG(q)$ and then used as the input of KWSD and LDASD. The improved approaches are referred to as KWSD + GR and LDASD + GR, respectively. The detail of HybridSD is given in Algorithm 1. For each candidate service $s_i \in CanS(q)$, the similarity between s_i and q is computed using GoSD, KWSD + GR, and LDASD + GR, respectively. The overall similarity between s_i and q is then obtained by aggregating the three similarities using

$$OSim(s_i, q) = \mu_1 \times GSim(s_i, q) + \mu_2 \times KSim(s_i, q) + \mu_3 \times LSim(s_i, q), \tag{19}$$

$$GSim(s_i, q) = \begin{cases} 1, & \text{if } s_i \in RelS(q) \\ 0, & \text{otherwise} \end{cases}, \tag{20}$$

where $GSim(s_i, q)$, $KSim(s_i, q)$, and $LSim(s_i, q)$ are the similarities between s_i and q computed using GoSD, KWSD + GR, and LDASD + GR, respectively; μ_1, μ_2, and μ_3 are three coefficients for aggregating the three similarities.

Algorithm 1. HybridSD

Input: user query q; parameters for topic matching and goal recommendation: k, θ, λ, and ζ; coefficients for aggregating three partial similarities: μ_1, μ_2, and μ_3.

Output: relevant service list for q, *RelSList*.

1. $RelSList \leftarrow \emptyset$;
2. $RelT_k(q) \leftarrow$ **TopicMatching**(q, k); // perform topic match ing for q and select the top k most relevant topics of q
3. $CanS(q) \leftarrow \bigcup_{t \in RelT_k(q)} SC_t$; // the candidate services for q
4. $SG(q) \leftarrow$ **GoalRecommendation**$(q, \theta, \lambda, \zeta)$; // perform goal recommendation for q and the user select a set of service goals from the recommendations
5. Build a new query nq by collecting the words of all service goals in $SG(q)$;
6. $SSMap \leftarrow \emptyset$; // a map for holding the similarities of candidate services to q
7. **for** $\forall s_i \in CanS(q)$ **do**
8. $gsim \leftarrow$ compute the similarity between s_i and q using GoSD based on $SG(q)$;
9. $ksim \leftarrow$ compute the similarity between s_i and nq using KWSD;
10. $lsim \leftarrow$ compute the similarity between s_i and nq using LDASD;
11. $osim \leftarrow \mu_1 \times gsim + \mu_2 \times ksim + \mu_3 \times lsim$; // compute the overall similarity of s_i to q
12. $SSMap$.put(s_i, $osim$);
13. $RelSList \leftarrow$ Sort the services in $SSMap$ according to their similarities to q;
14. **return** *RelSList*;

6 Experiments and analysis

We conducted a series of experiments to evaluate the proposed goal recommendation mechanism and service discovery approaches. All experiments were developed in Java and carried out on a PC with Intel(R) Core(TM)2 T7300 2GHz CPU, 2GB RAM, and running Microsoft Windows 7 x86.

6.1 Experimental dataset

We have crawled the descriptive data (including name, tags, and textual description) of 10,168 services belonging to 66 categories from PW on April 14, 2014. An experimental dataset of 1249 services was constructed from seven categories, as depicted in Table 1. The total numbers of services, sentences, words, and typed dependencies (including nsubjpass, dobj, prep, prt, amod, nn, conj, and appos) in each category are listed. Please note that two kinds of prep dependencies are presented separately and their differences have been introduced in Sect. 4.2. As preparation, we performed the three offline data process steps described in Sect. 4 over the experimental dataset as follows.

– *Service clustering based on topic models* We preprocessed the descriptive data of each service by tokenization, stemming, and stop word removal, and applied LDA to the services using different numbers of latent topics (i.e., T), ranging from 20 to 50. The Dirichlet prior parameters were set as $\alpha = 50/T$ and $\beta = 0.1$ like [32,43]. For each T, we discarded the first 2000 Gibbs samples to make sure that the "burn-in" period was passed. By computing the posterior probabilities of trained LDA models using the method discussed in [43], we chose the best T as 36, which led to the maximum posterior

Table 1 Statistics of the experimental dataset

	Media management	Music	Photos	Transportation	Travel	Video	Weather
#services	64	209	254	177	227	232	86
#sentences	264	854	1058	734	971	1011	348
#words	4760	15,889	19,231	13,913	18,219	18,027	6799
#nsubjpass	43	108	133	86	131	138	51
#dobj	284	936	1257	796	1067	1188	368
#prep (used for initial goal generation)	24	107	101	82	155	93	33
#prt	2	18	25	18	20	17	4
#amod	280	958	1147	861	1096	1194	500
#nn	403	1395	1649	1576	1857	1620	760
#conj	381	1225	1285	945	1316	1308	487
#appos	35	86	110	104	109	108	58
#prep (used for initial goal extension)	408	1302	1645	1247	1610	1442	646

probability. The LDA model trained using $T = 36$ was selected for our experiments. Based on the service-topic distributions, we clustered the services using top k cluster assignment. Figure 8a depicts the distribution of the size of service clusters generated using different k values, ranging from 1 to 5. As can be seen, a larger k leads to larger service clusters.

– *Service goal extraction* Service goals contained in the textual description of each services were extracted using our service goal extraction approach proposed in [6], resulting in a service-service goal assignment matrix. In total, 4857 service goals were extracted from the services.
– *Service goal clustering* The extracted service goals were folded into the LDA model trained for services. We ran 2000 iterations to do LDA inference. Based on the inferred service goal-topic distributions, we clustered the service goals using top k cluster assignment. Figure 8b depicts the distribution of the size of service goal clusters generated using different $k$values, ranging from 1 to 5. A larger k also leads to larger service goal clusters.

To evaluate the proposed goal recommendation mechanism and service discovery approaches, we conducted a series of empirical experiments on a set of queries by recruiting three master students. As the manual creation of the ground truth of relevant service goals and relevant services for a query is an expensive process, we just randomly selected ten queries, as presented in Table 2, for evaluation. For each query, the three students first determined the relevance of services and service goals independently by labeling each of them as 0 (irrelevant) or 1 (relevant); afterward they discussed the debatable ones together to reach an agreement on the final ground truth of relevant services and relevant service goals. The statistics of the ground truth on ten queries is presented in Table 2.

Based on the ground truth of each query q, we evaluated the ranking list of recommended service goals and retrieved services produced for q using three popular metrics: P@N (precision), R@N (recall), and MAP@N (mean average precision).

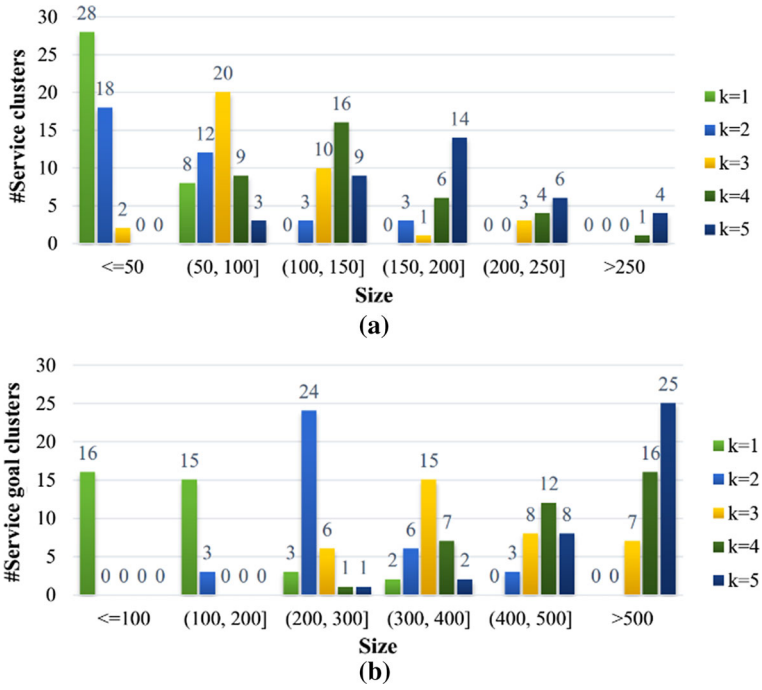


Fig. 8 Distribution of the size of service clusters and service goal clusters generated using different top k cluster assignment; k ranges from 1 to 5. **a** Distribution of the size of service clusters. **b** Distribution of the size of service goal clusters

$$\text{P@N} = \frac{|S_r \cap S_g|}{|S_r|}, \quad \text{R@N} = \frac{|S_r \cap S_g|}{|S_g|}, \quad \text{MAP@N} = \frac{1}{|S_g|} \sum_{i=1}^{N} \left(\frac{n_i}{i} \cdot I(i) \right) \tag{21}$$

where S_r is the set of the top N recommended service goals (or retrieved services), S_g is the set of relevant service goals (or services) in the ground truth of q, n_i represents the number of relevant service goals (or services) in S_g that exist in the first i of the recommended list, and $I(i)$ indicates whether the service goal (or service) at the ranking position i is in S_g.

6.2 Evaluation of goal recommendation

For each query q in Table 2, we preprocessed it by tokenization and stemming, and folded it into the LDA model trained for services. We ran 100 iterations to perform LDA inference for q. The top k topics with maximum probabilities were selected as relevant topics of q, $RelT_k(q)$. Afterward, service goals assigned to the clusters (generated using top k cluster assignment) that correspond to $RelT_k(q)$ were sorted according to their semantic similarities to q computed using Eq. (5) and recommended. Based on the relevant service goals of q in the ground truth, we measured the performance of the recommended service goal list using P@N, R@N, and MAP@N. Finally, we measured the overall P@N, R@N, and MAP@N performance of goal recommendation mechanism with respect to the average of ten queries' performance results.

We conducted the goal recommendation mechanism based on the service goal clusters generated using different k settings (varied from 1 to 5) and the corresponding $RelT_k(q)$.

Table 2 Experimental queries and statistics of the ground truth

Query ID	Description	#Relevant service goals	#Relevant services
1	Book hotel	24	44
2	Create video	38	39
3	Edit video	14	14
4	Find airport	22	22
5	Forecast weather	19	39
6	Get hotel	57	57
7	Publish video	6	14
8	Search playlist	12	16
9	Share music	14	29
10	Upload photo	27	64

Table 3 Performance of goal recommendation using different k settings

k	Metric	N							
		5	10	15	20	25	30	35	40
1	P@N	0.94	0.81	0.7	0.61	0.532	0.4667	0.4057	0.365
	R@N	0.281	0.4421	0.538	0.6027	0.6449	0.665	0.6685	0.6878
	MAP@N	0.4577	0.7846	0.9887	1.1315	1.2204	1.2741	1.286	1.3048
2	P@N	0.92	0.84	0.72	0.665	0.6	0.54	0.4829	0.4325
	R@N	0.2867	0.4668	0.566	0.6657	0.7273	0.7745	0.7978	0.81
	MAP@N	0.4348	0.7948	1.0105	1.2104	1.3475	1.4423	1.4925	1.5138
3	P@N	0.9	0.82	0.7	0.65	0.588	0.5333	0.4743	0.42
	R@N	0.2841	0.457	0.5544	0.6594	0.7154	0.7643	0.7793	0.7852
	MAP@N	0.4268	0.7693	0.9743	1.1721	1.3112	1.4113	1.4645	1.4748
4	P@N	0.86	0.82	0.7	0.65	0.584	0.5367	0.4771	0.4225
	R@N	0.2603	0.4665	0.5694	0.6743	0.7257	0.781	0.7959	0.8018
	MAP@N	0.394	0.747	0.9529	1.1479	1.2808	1.3913	1.4442	1.4543
5	P@N	0.86	0.82	0.7	0.65	0.584	0.53	0.4771	0.42
	R@N	0.2603	0.4665	0.5694	0.6743	0.7257	0.7719	0.7959	0.8001
	MAP@N	0.394	0.747	0.9529	1.1479	1.2808	1.3842	1.4432	1.447

Table 3 presents the overall P@N, R@N, and MAP@N performance achieved with different k values. P@N is higher than 0.6 when $N \leq 20$, meaning that most of the top 20 recommended service goals are relevant to queries. R@N is higher than 0.68 when $N = 40$, indicating that most of the relevant service goals of queries are recommended by top 40. In most cases, P@N, R@N, and MAP@N achieved using $k = 2$ are better than those achieved using $k = 1$. This can be explained by the fact that some relevant service goals outside the most relevant clusters of queries are recommended by increasing the search scope of service goals. Moreover, as k becomes larger than 2, all the three metrics decrease, which is due to the fact that many irrelevant service goals of queries are recommended by exploring three or more clusters and are sorted highly in the recommendation lists on account of inappropriate semantic similarities computed using Eq. (5).

Table 4 Time cost of goal recommendation on ten queries using different k settings

k	1	2	3	4	5
Time (ms)	62	119	204	398	470

Table 4 presents the total time cost of goal recommendation on ten queries under different k settings. It can be seen that the time cost increases with the increase of k. This is because that a larger k leads to a larger set of service goals recommended for queries; and more time is required for sorting the recommended goals by measuring their semantic similarities to the corresponding query. Although the time cost produced with $k = 2$ is higher than that produced with $k = 1$, it is much lower than those produced with $k = 3, 4, 5$.

Based on the above analysis, the goal recommendation mechanism achieves the best performance with relatively low time cost under the setting of $k = 2$; therefore, we chose $k = 2$ for our experiments. For the sake of simplicity, service clusters generated using top two cluster assignment were subsequently used for reducing the service search space.

More specifically, Table 5 presents the performance of goal recommendation on each of the ten queries using $k = 2$. For most queries, P@N achieved at top 20 is higher than 0.6 and R@N achieved at top 40 is higher than 0.8, which are consistent with the overall performance presented in Table 3. Moreover, there are some low P@N and low R@N cases. Based on our analysis, the low P@N on some queries, e.g., "publish video," is caused by either (1) only a few relevant service goals are identified by students for the queries, or (2) many irrelevant service goals of the queries are sorted high in the recommendation list. The low R@N on query "get hotel" is mainly caused by the fact that a considerable number of (more than 40) relevant service goals are identified by students for the query.

As discussed in Sect. 5.2, service goals recommended for user queries can be sorted using different similarity measures such as Jaccard similarity (JSim), Cosine similarity (CSim), and the semantic similarity based on WordNet (WNSim). Moreover, we build a historical goal recommendation repository (*HGRR*) to store the service goals selected by users for queries and design a hybrid goal recommendation mechanism by incorporating the recommendation list generated using WNSim and the recommendation list generated based on *HGRR*. We conducted the goal recommendation mechanisms using three different similarity measures (i.e., JSim, CSim, and WNSim) and the hybrid mechanism for each of the queries in Table 2. These mechanisms are referred to as JSimGR, CSimGR, WNSimGR, and HybridGR, respectively. All the mechanisms were conducted based on the service goal clusters generated using top two cluster assignment. For HybridGR, *HGRR* was built by collecting relevant service goals contained in the top 40 of the recommendation lists generated using WNSimGR with $k = 2$. Note that the ground truth of relevant service goals of each query is shared by the three students in *HGRR*. Some parameters were set as follows.

The parameter θ used for determining the queries in *HGRR*. Q that are similar to a current user query was set as 0.7 according to our manual analysis of all queries over the dataset. The parameter ζ is to balance the weights of the semantic similarity computed using WNSim and the importance measured based on *HGRR*. We set $\zeta = 0.5$ because we only want to check the performance of HybridGR under a naïve setting of these two factors. The parameter λ controls the weights between the number of service goals selected by the current user and the number of service goals selected by other users in the important measurement of service goals recommended based on *HGRR*. Since the service goals that have been selected by the current user would be more preferred by himself, λ should be (0.5, 1]. We ran a grid search on (0.5, 1] with a step size to be 0.1, and selected $\lambda = 0.8$ that maximizes the performance of HybridGR.

Table 5 Performance of goal recommendation on ten queries using $k = 2$

Query ID	N = 10			N = 20			N = 30			N = 40		
	P@N	R@N	MAP@N	P@N	R@N	MAP@N	P@N	R@N	MAP@N	P@N	R@N	MAP@N
1	1.0	0.4167	1.0	0.7	0.5833	1.3824	0.4667	0.5833	1.3824	0.375	0.625	1.424
2	0.9	0.2368	0.7071	0.95	0.5	1.6402	0.9333	0.7368	2.4828	0.775	0.8158	2.764
3	1.0	0.7143	1.0	0.55	0.7857	1.1	0.3667	0.7857	1.1	0.275	0.7857	1.1
4	0.7	0.3182	0.4675	0.55	0.5	0.7352	0.5	0.6818	0.9493	0.475	0.8636	1.1528
5	1.0	0.5263	1.0	0.85	0.8947	1.647	0.6	0.9474	1.7288	0.45	0.9474	1.7288
6	0.9	0.1579	0.9	0.85	0.2982	1.5516	0.7	0.3684	1.8878	0.6	0.4211	2.0764
7	0.5	0.8333	0.5	0.25	0.8333	0.5	0.1667	0.8333	0.5	0.125	0.8333	0.5
8	0.5	0.4167	0.4833	0.4	0.6667	0.5935	0.3667	0.9167	0.7032	0.275	0.9167	0.7032
9	1.0	0.7143	1.0	0.65	0.9286	1.2724	0.4333	0.9286	1.2724	0.325	0.9286	1.2724
10	0.9	0.3333	0.89	0.9	0.6667	1.682	0.8667	0.963	2.4161	0.65	0.963	2.4161

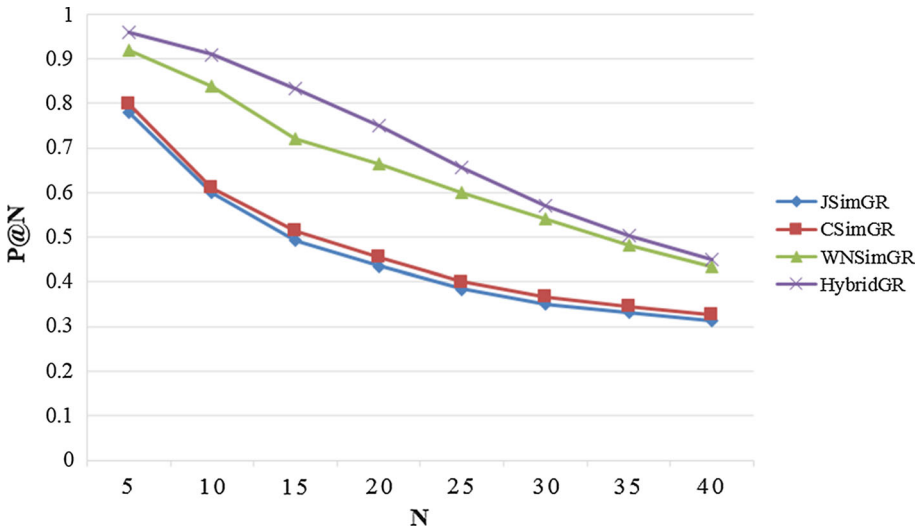


Fig. 9 P@N of goal recommendation mechanisms

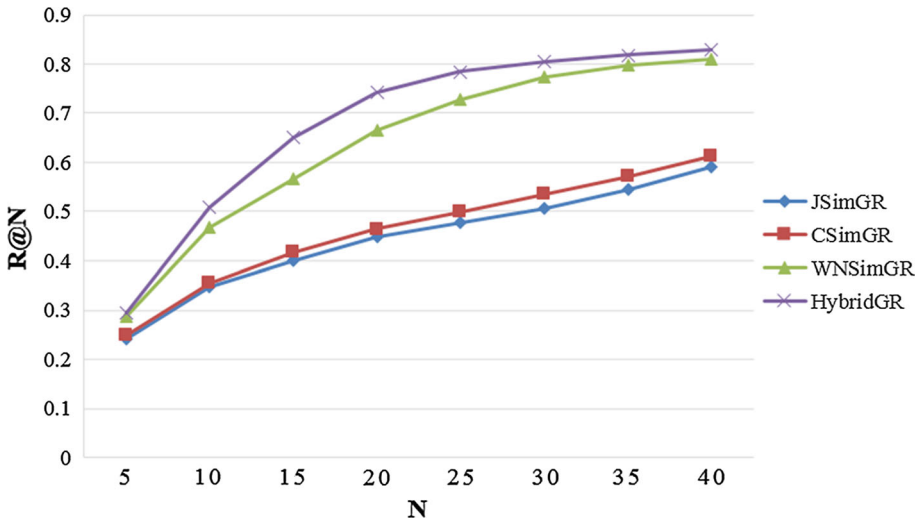


Fig. 10 R@N of goal recommendation mechanisms

Figures 9, 10, and 11 show the overall P@N, R@N, and MAP@N performance of four goal recommendation mechanisms, respectively. In terms of all the three metrics, JSimGR and CSimGR are quite close, and WNSimGR is much better than those two mechanisms, which is caused by the fact that WNSimGR can sort semantically relevant service goals of queries highly in the recommendation lists while JSimGR and CSimGR cannot. Moreover, HybridGR is better than WNSimGR, which demonstrates that the service goal recommendation lists generated using WNSimGR can be improved by leveraging *HGRR*, namely the service goals that have been selected by users for queries.

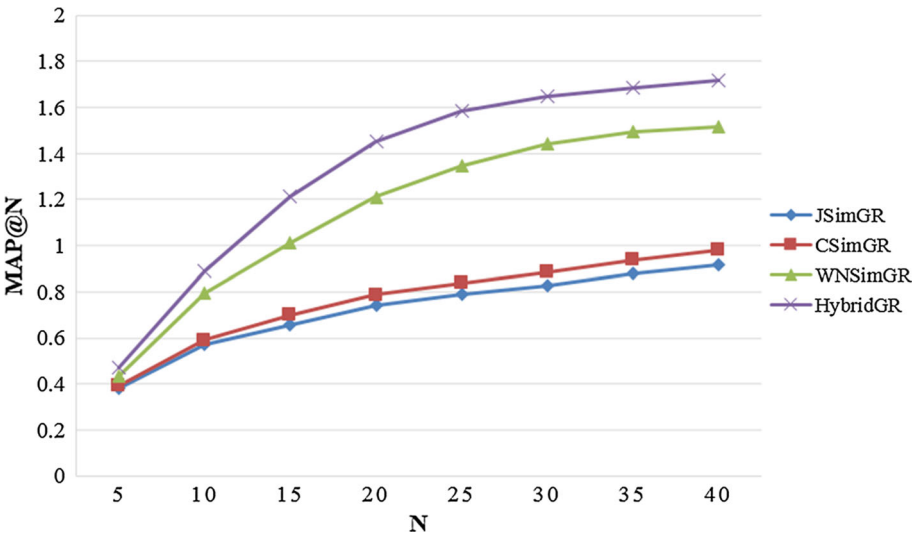


Fig. 11 MAP@N of goal recommendation mechanisms

6.3 Evaluation of service discovery

As stated previously, existing RESTful service discovery approaches are mainly based on hRESTS; and it is difficult to construct hRESTS for publicly registered services. Moreover, it is also difficult to find suitable ontologies to semantically annotate services in PW, which makes ontology-based service discovery hard to be an approach for comparison. Therefore, we compared the proposed goal-based service discovery approach (GoSD) and hybrid service discovery approach (HybridSD) with two well-known approaches, i.e., keyword-based service discovery (KWSD) and LDA-based service discovery (LDASD). The details of these approaches are given in Sect. 5.3. Service clusters generated using the top two cluster assignment were used for reducing the service search space. The relevant service goals contained in the top 40 recommendations of WNSimGR were used for GoSD and HybridSD. For HybridSD, we set the parameters in Eq. (19) as $\mu_1 = \mu_2 = \mu_3 = 1$, such that the three components were equally treated.

We conducted the four service discovery approaches for each query in Table 2. For each service discovery approach, the performance of the service list retrieved for each query q was measured using P@N, R@N, and MAP@N based on the ground truth of relevant services of q; and the overall P@N, R@N, and MAP@N performance of the approach was then measured with respect to the average of ten queries' performance results.

Figures 12, 13, and 14 show the overall P@N, R@N, and MAP@N performance of four service discovery approaches, respectively. In terms of MAP@N, the order is HybridSD $>$ GoSD $>$ KWSD $>$ LDASD. In terms of P@N, GoSD is higher than 0.9 and better than the other three approaches in most cases, indicating that GoSD can retrieve more accurate services for queries. Because users often expect to quickly find services of interest, P@N should be as high as possible. From this perspective, GoSD will be more preferred by users. In terms of R@N, GoSD outperforms KWSD and LDASD when $N \leq 30$. As N becomes larger than 30, R@N of GoSD stabilizes, which indicates that no more services relevant to queries are retrieved using this approach. Through analysis, this issue is mainly caused by

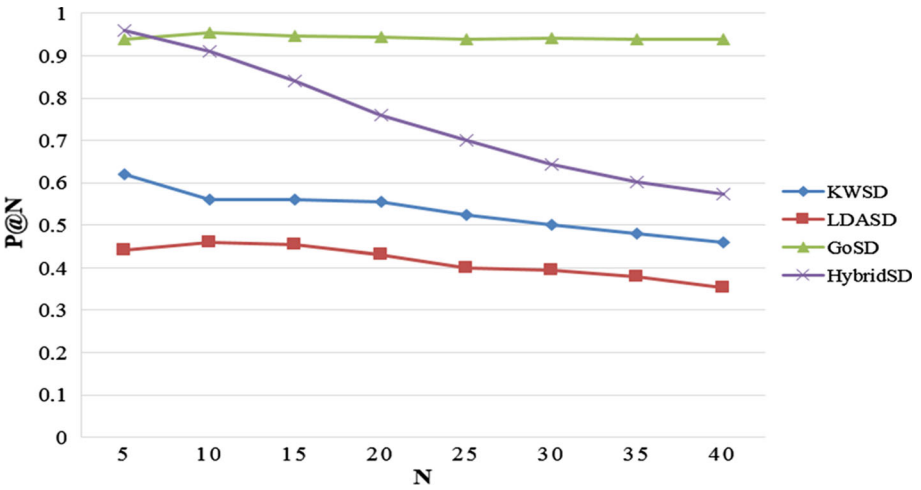


Fig. 12 P@N of service discovery approaches

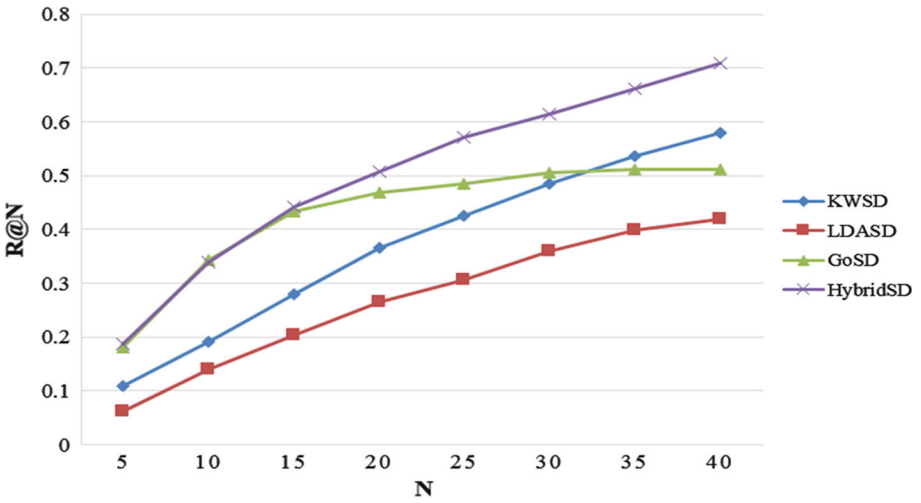


Fig. 13 R@N of service discovery approaches

two reasons: (1) some useful service goals fail to be extracted from the textual descriptions of services, as explained in Sect. 5.3, and (2) some relevant service goals of queries are not recommended by top 40 and thus are not used for GoSD. R@N of HybridSD is higher than that of GoSD when $N > 15$, because some relevant services missed by GoSD are retrieved by the other two complementary approaches, i.e., KWSD + GR and LDASD + GR. Meanwhile, P@N of HybridSD is lower than GoSD due to the irrelevant services introduced by KWSD + GR and LDASD + GR.

In the above experiments, the service search space was restricted to the service clusters that correspond to the top two relevant topics of each query. We also conducted service discovery approaches for the queries in Table 2 with a search over all services. Tables 6, 7, and 8 present the overall P@N, R@N, and MAP@N performance of four service discovery

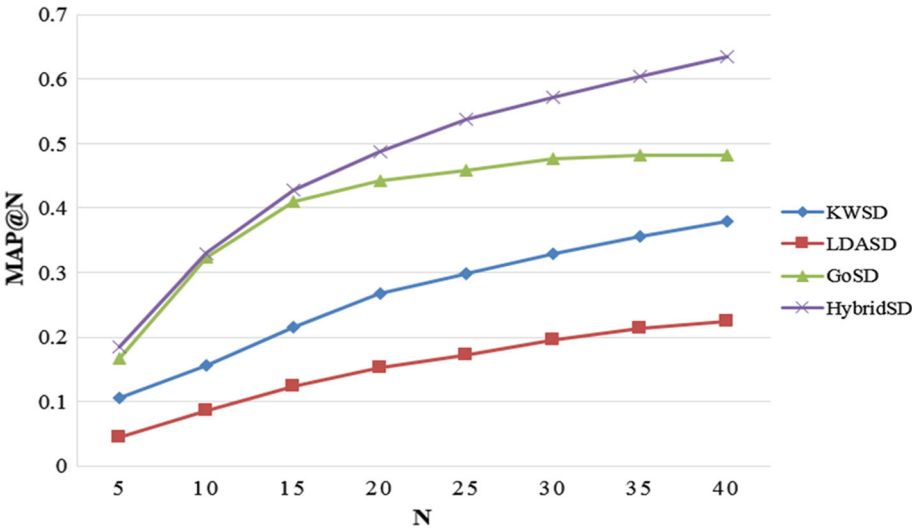


Fig. 14 MAP@N of service discovery approaches

Table 6 P@N of service discovery based on reduced and full service search spaces

Approach	Service searching space	N							
		5	10	15	20	25	30	35	40
GoSD	Reduced	0.94	0.9533	0.9456	0.944	0.939	0.9423	0.9402	0.9402
	Full	0.98	0.9333	0.9474	0.9458	0.9488	0.9441	0.9408	0.9423
KWSD	Reduced	0.62	0.56	0.56	0.555	0.524	0.5	0.48	0.46
	Full	0.62	0.56	0.5533	0.545	0.52	0.5067	0.48	0.4675
LDASD	Reduced	0.44	0.46	0.4533	0.43	0.4	0.3933	0.38	0.3525
	Full	0.44	0.46	0.4533	0.43	0.4	0.3933	0.38	0.3525
HybridSD	Reduced	0.96	0.91	0.84	0.76	0.7	0.6433	0.6029	0.5725
	Full	0.96	0.91	0.8533	0.77	0.712	0.6667	0.6229	0.595

Table 7 R@N of service discovery based on reduced and full service search spaces

Approach	Service search space	N							
		5	10	15	20	25	30	35	40
GoSD	Reduced	0.182	0.3434	0.4342	0.4688	0.4853	0.5059	0.511	0.511
	Full	0.1891	0.3346	0.4449	0.4864	0.507	0.5235	0.54	0.553
KWSD	Reduced	0.1094	0.1908	0.2795	0.3669	0.4246	0.4848	0.5367	0.5794
	Full	0.1094	0.1908	0.2772	0.3601	0.4206	0.4875	0.5299	0.5929
LDASD	Reduced	0.0618	0.1392	0.2041	0.2648	0.3065	0.3598	0.3994	0.4192
	Full	0.0618	0.1392	0.2041	0.2648	0.3065	0.3598	0.3994	0.4192
HybridSD	Reduced	0.1874	0.3391	0.441	0.508	0.5704	0.6145	0.6613	0.7091
	Full	0.1874	0.3391	0.4518	0.5191	0.5857	0.6396	0.6815	0.7346

Table 8 MAP@N of service discovery based on reduced and full service search spaces

Approach	Service search space	N							
		5	10	15	20	25	30	35	40
GoSD	Reduced	0.1673	0.3229	0.4099	0.443	0.4582	0.4769	0.4813	0.4813
	Full	0.1877	0.3281	0.4319	0.472	0.491	0.5063	0.5212	0.533
KWSD	Reduced	0.1056	0.1556	0.2149	0.267	0.2987	0.3292	0.356	0.3796
	Full	0.1056	0.1553	0.213	0.2614	0.2976	0.3344	0.3578	0.3861
LDASD	Reduced	0.0454	0.0861	0.1236	0.1525	0.1727	0.1961	0.2135	0.2237
	Full	0.0454	0.0861	0.1236	0.1525	0.1727	0.1961	0.2135	0.2237
HybridSD	Reduced	0.1841	0.3298	0.427	0.488	0.5382	0.572	0.6047	0.6343
	Full	0.1841	0.3298	0.4366	0.4985	0.552	0.5945	0.6271	0.6614

Table 9 Time cost of service discovery on ten queries based on reduced and full service search spaces

	Service search space	GoSD	KWSD	LDASD	HybridSD
Time (ms)	Reduced	321	329	584	1234
	Full	714	548	1051	2313

approaches based on the reduced and full service search spaces, respectively. For GoSD, KWSD, and HybridSD, the performance achieved at top 40 based on the full service search space is a little better than that achieved based on the reduced service search space. This is because that some relevant services of queries outside the top two service clusters are retrieved over all services. However, as presented in Table 9, it requires more time to compare each of the ten queries to all services.

It has been demonstrated that the proposed goal recommendation mechanism can effectively recommend relevant service goals for user queries and that the user selected service goals can help retrieve relevant services accurately. Moreover, we validated the contribution of the goal recommendation mechanism to existing service discovery approaches. Specifically, we conducted the original KWSD and LDASD and their improved versions (i.e., KWSD + GR and LDASD + GR) for ten queries in Table 2. For each query, a new query was built by collecting the words of relevant service goals contained in the top 40 of service goal recommendation list produced using WNSimGR, and then used as the input of KWSD + GR and LDASD + GR, as described in Sect. 5.3. Table 10 presents the overall P@N, R@N, and MAP@N performance of these approaches. In terms of all metrics, the improved approaches perform better than their original versions in most cases. Specifically, P@N, R@N, and MAP@N of KWSD are improved, on average, by 13.35, 15.93, and 25.18%, respectively. P@N, R@N, and MAP@N of LDASD are improved, on average, by 4.49, 7.27, and 17.32%, respectively. These results show that the proposed goal recommendation mechanism can help improve the search results of KWSD and LDASD.

7 Conclusions

This paper reports our continuous efforts on RESTful service discovery. In our prior work [6], we proposed an approach to extract service goals from the textual descriptions of RESTful

Table 10 Performance of service discovery improved by goal recommendation

Approach	Metric	N							
		5	10	15	20	25	30	35	40
KWSD	P@N	0.62	0.56	0.56	0.555	0.524	0.5	0.48	0.46
	R@N	0.1094	0.1908	0.2795	0.3669	0.4246	0.4848	0.5367	0.5794
	MAP@N	0.1056	0.1556	0.2149	0.267	0.2987	0.3292	0.356	0.3796
KWSD + GR	P@N	0.74	0.68	0.6667	0.615	0.556	0.54	0.5257	0.5175
	R@N	0.1392	0.2526	0.3421	0.4147	0.4559	0.5186	0.5799	0.6375
	MAP@N	0.1305	0.2158	0.2839	0.3308	0.3575	0.3949	0.4303	0.4656
LDASD	P@N	0.44	0.46	0.4533	0.43	0.4	0.3933	0.38	0.3525
	R@N	0.0618	0.1392	0.2041	0.2648	0.3065	0.3598	0.3994	0.4192
	MAP@N	0.0454	0.0861	0.1236	0.1525	0.1727	0.1961	0.2135	0.2237
LDASD + GR	P@N	0.52	0.52	0.4533	0.42	0.4	0.39	0.3857	0.375
	R@N	0.0769	0.1563	0.2115	0.2609	0.3201	0.3682	0.4098	0.4606
	MAP@N	0.0611	0.1126	0.1441	0.1696	0.1932	0.2154	0.2364	0.2527

services. In this work, we propose an approach for RESTful service discovery by leveraging the mined service goal knowledge. We use topic models, e.g., LDA, to group the available services into clusters. The service goals extracted from the services' textual descriptions are also clustered by folding them into the topic model trained for services. In service discovery, service clusters are used for reducing the service search space, in order to improve the efficiency. We particularly design a mechanism to address the incompleteness of user queries by recommending semantically similar service goals based on the service goal clusters. Finally, relevant services are retrieved by matching user selected service goals with those of candidate services. Moreover, we propose a hybrid service discovery approach by integrating the goal-based service discovery approach with two popular service discovery approaches: the keyword-based approach and a probabilistic semantic approach based on LDA. Experiment results conducted on a real-world service dataset demonstrate the effectiveness of the proposed approaches.

There are still some limitations in our proposed approach. Firstly, our approach cannot cope with the case that all sentences express a goal. This issue might be solved by analyzing more typed dependencies and identifying templates within sentences. Secondly, our approach cannot deal with the service goals represented as noun phrases, e.g., "hotel booking" and "order management." Thirdly, some useful typed dependencies, e.g., pobj, mark, and xcomp, have not been considered by our approach, which may result in the missing of some meaningful service goals.

In the future, we will improve the proposed approach by solving these limitations In addition, we will attempt to establish semantic relationships among service goals by exploring the structures of mashups composed by APIs, and external ontologies or knowledge bases (e.g., Wikipedia), to promote goal recommendation. We also plan to apply the proposed approaches in our service supermarket CloudCRM [52].

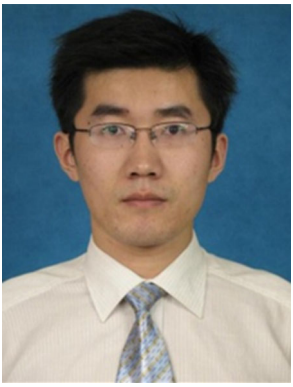
Acknowledgements This research was supported by the National Basic Research Program of China (No. 2014CB340404), the National Key Research and Development Program of China (No. 2017YFB1400602), and the National Natural Science Foundation of China (Nos. 61672387, 61702378, 61402150, and 61562073), the Strategic Team-Building of Scientific and Technological Innovation in Hubei Province, and the Natural Science Foundation of Hubei Province of China (No. 2017CKB894).

References

1. Hu Y, Peng Q, Hu X et al (2015) Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering. IEEE Trans Serv Comput 8(5):782–794
2. John D, Rajasree MS (2013) RESTDoc: describe, discover and compose RESTful semantic web services using annotated documentations. Int J Web Semant Technol 4(1):37–49
3. Fielding RT (2000) Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine
4. Maleshkova M, Pedrinaci C, Domingue J (2010) Investigating web APIs on the World Wide Web. In: Proceedings of the IEEE European conference on web services, pp 107–114
5. Jiang W, Lee D, Hu S (2012) Large-scale longitudinal analysis of SOAP-based and RESTful web services. In: Proceedings of the IEEE international conference on web services, pp 218–225
6. Wang J, Zhang N, Zeng C et al (2013) Towards services discovery based on service goal extraction and recommendation. In: Proceedings of the IEEE international conference on services computing, pp 65–72
7. Wang Y, Stroulia E (2003) Flexible interface matching for web service discovery. In: Proceedings of the international conference on web information systems engineering, pp 147–156
8. Plebani P, Pernici B (2009) URBE: web service retrieval based on similarity evaluation. IEEE Trans Knowl Data Eng 21(11):1629–1642
9. Liu F, Shi Y, Yu J et al (2010) Measuring similarity of web services based on WSDL. In: Proceedings of the IEEE international conference on web services, pp 155–162
10. Kokash N, Heuvel WJVD, D'Andrea V (2006) Leveraging web services discovery with customizable hybrid matching. In: Proceedings of the international conference on service-oriented computing, pp 522–528
11. Paulraj D, Swamynathan S (2011) Content based service discovery in semantic web services using WordNet. In: Proceedings of the international conference on advanced computing, network and security, pp 48–56
12. Paliwal AV, Shafiq B, Vaidya J et al (2012) Semantics-based automated service discovery. IEEE Trans Serv Comput 5(2):260–275
13. Ma SP, Li CH, Tsai YY et al (2013) Web service discovery using lexical and semantic query expansion. In: Proceedings of the IEEE international conference on e-business engineering, pp 423–428
14. Cong Z, Fernandez A, Billhardt H et al (2015) Service discovery acceleration with hierarchical clustering. Inf Syst Front 17(4):799–808
15. Ma J, Zhang Y, He J (2008) Efficiently finding web services using a clustering semantic approach. In: Proceedings of the international workshop on context enabled source and service selection, integration and adaptation, pp 1–8
16. Cassar G, Barnaghi P, Moessner K (2013) Probabilistic matchmaking methods for automated service discovery. IEEE Trans Serv Comput 7(4):654–666
17. Li Z, He K, Wang J et al (2014) An on-demand services discovery approach based on topic clustering. J Internet Technol 15(4):543–555
18. Wang J, Gao P, Ma Y et al (2017) A web service discovery approach based on common topic groups extraction. IEEE Access 5:10193–10208
19. Chen L, Hu L, Zheng Z, et al (2011) WTCluster: utilizing tags for web services clustering. In: Proceedings of the international conference on service-oriented computing, pp 204–218
20. Klusch M, Fries B, Sycara K (2009) OWLS-MX: a hybrid semantic web service matchmaker for OWL-S services. Web Semant Sci Serv Agents World Wide Web 7(2):121–133
21. Klusch M, Kapahnke P, Zinnikus I (2009) Hybrid adaptive web service selection with SAWSDL-MX and WSDL-analyzer. In: Proceedings of the European semantic web conference on the semantic web: research and applications, pp 550–564
22. Klusch M, Kaufer F (2009) WSMO-MX: a hybrid semantic web service matchmaker. Web Intell Agent Syst 7(1):23–42
23. García JM, Ruiz D, Ruiz-Cortés A (2012) Improving semantic web services discovery using SPARQL-based repository filtering. Web Semant Sci Serv Agents World Wide Web 17(4):12–24
24. Lampe U, Schulte S, Siebenhaar M, et al (2010) Adaptive matchmaking for RESTful services based on hRESTS and MicroWSMO. In: Proceedings of the workshop on emerging web services technology, pp 10–17
25. Sellami S, Slaimi F, Boucelma O et al (2013) Flexible matchmaking for RESTful web services. In: Proceedings of the OTM conference on the move to meaningful internet systems, pp 542–554
26. Roman D, Kopecký J, Vitvar T et al (2015) WSMO-Lite and hRESTS: lightweight semantic annotations for web services and RESTful APIs. Web Semant Sci Serv Agents World Wide Web 31:39–58

27. Zhang N, He K, Wang J et al (2016) WSGM-SD: an approach to RESTful service discovery based on weighted service goal model. Chin J Electron 25(2):256–263
28. Jung Y, Cho Y, Park YM et al (2013) Automatic tagging of functional-goals for goal-driven semantic service discovery. In: Proceedings of the IEEE international conference on semantic computing, pp 212–219
29. Kopecky J, Gomadam K, Vitvar T (2008) hRESTS: an HTML microformat for describing RESTful web services. In: Proceedings of the IEEE international conference on web intelligence and intelligent agent technology, pp 619–625
30. Liu X, Agarwal S, Ding C et al (2016) An LDA-SVM active learning framework for web service classification. In: Proceedings of the IEEE international conference on web services, pp 49–56
31. Strohmaier M, Lux M, Granitzer M et al (2007) How do users express goals on the web? An exploration of intentional structures in web search. In: Proceedings of the international conference on web information systems engineering, pp 67–78
32. Zhou TC, Lyu RT, King I et al (2014) Learning to suggest questions in social media. Knowl Inf Syst 43(2):389–416
33. Miller GA (1995) WordNet: a lexical database for English. ACM Commun 38(11):39–41
34. Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. J Mach Learn Res 3:993–1022
35. Wang H, Shi Y, Zhou X, et al (2010) Web service classification using support vector machine. In: Proceedings of the IEEE international conference on tools with artificial intelligence, pp 3–6
36. Zhang J, Wang J, Hung P et al (2012) Leveraging incrementally enriched domain knowledge to enhance service categorization. Int J Web Serv Res 9(3):43–66
37. Chen L, Wang Y, Yu Q et al (2013) WT-LDA: user tagging augmented LDA for web service clustering. In: Proceedings of the international conference on service-oriented computing, pp 162–176
38. Crasso M, Zunino A, Campo M (2011) A survey of approaches to web service discovery in service-oriented architectures. J Database Manag 22(1):102–132
39. Wang W, Barnaghi P, Bargiela A (2010) Probabilistic topic models for learning terminological ontologies. IEEE Trans Knowl Data Eng 22(7):1028–1040
40. Bird S, Loper E, Klein E (2009) Natural language processing with Python. O'Reilly Media Inc., Sebastopol
41. Korenius T, Laurikkala J, Järvelin K, et al (2004) Stemming and lemmatization in the clustering of Finnish text documents. In: Proceedings of the ACM international conference on information and knowledge management, pp 625–633
42. Yao L, Mimno D, Mccallum A (2009) Efficient methods for topic model inference on streaming document collections. In: Proceedings of the ACM international conference on knowledge discovery and data mining, pp 937–946
43. Griffiths TL, Steyvers M (2004) Finding scientific topics. Proc Nat Acad Sci USA 101(Suppl 1):5228–5235
44. Andrieu C, Freitas ND, Doucet A et al (2002) An introduction to MCMC for machine learning. Mach Learn 50(1):5–43
45. Zhang N, Wang J, Ma Y (2017) Mining domain knowledge on service goals from textual service descriptions. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2017.2693147
46. Rolland C, Souveyet C, Achour CB (1998) Guiding goal modeling using scenarios. IEEE Trans Softw Eng 24(12):1055–1071
47. Stevenson M, Greenwood MA (2006) Comparing information extraction pattern models. In: Proceedings of the workshop on information extraction beyond the document. Association for Computational Linguistics, pp 12–19
48. Santorini B (1990) Part-of-speech tagging guidelines for the Penn Treebank Project. Technical Reports (CIS), Paper 570
49. Marnee MCD, Manning CD (2008) Stanford typed dependencies manual
50. Manning CD, Raghavan P, Schütze H (2009) An introduction to information retrieval. Cambridge University Press, Cambridge
51. Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Int J Inf Process Manag 24(5):513–523
52. Wang J, Feng Z, Zhang J et al (2014) A unified RGPS-based approach supporting service-oriented process customization. Web Serv Foundations, pp 657–682

Neng Zhang is a Ph.D. student in the State Key Laboratory of Software Engineering, Wuhan University, China. He received the B.S. degree in 2012 from Wuhan University, China. His research interests include services computing and knowledge mining.



Jian Wang is a lecturer of the State Key Lab of Software Engineering, Computer School, Wuhan University, China. He received the Ph.D. degree in 2008 from Wuhan University, China. His current research interests include services computing and software engineering.

Keqing He is a professor of the State Key Lab of Software Engineering, Computer School, Wuhan University, China. He received his doctoral degree from Hokkaido University of Japan in 1995. His research interests include services computing and software engineering.

Zheng Li is an associate professor of the School of Computer and Information Engineering, Henan University, Kaifeng, China. She received the Ph.D. degree in 2013 from Wuhan University, China. Her current research interests include services computing and software engineering.

Yiwang Huang is an associate professor of the School of Data Science, Tongren University, Tongren, China. He received the Ph.D. degree in 2015 from Wuhan University, China. His current research interests include services computing and software engineering.