The Journal of Systems and Software 142 (2018) 73–91

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Web service discovery based on goal-oriented query expansion

Neng Zhang[a], Jian Wang[a,*], Yutao Ma[a], Keqing He[a], Zheng Li[b], Xiaoqing (Frank) Liu[c]

[a] School of Computer Science, Wuhan University, Wuhan, China
[b] School of Computer and Information Engineering, Henan University, Kaifeng, China
[c] Computer Science and Computer Engineering Department, University of Arkansas, Fayetteville, USA

ARTICLE INFO

Article history:
Received 13 July 2017
Revised 9 March 2018
Accepted 18 April 2018
Available online 21 April 2018

Keywords:
Web service
Service discovery
Service-based system (SBS)
Service goal knowledge
Query expansion

ABSTRACT

With the broad adoption of service-oriented architecture, many software systems have been developed by composing loosely-coupled Web services. Service discovery, a critical step of building service-based systems (SBSs), aims to find a set of candidate services for each functional task to be performed by an SBS. The keyword-based search technology adopted by existing service registries is insufficient to retrieve semantically similar services for queries. Although many semantics-aware service discovery approaches have been proposed, they are hard to apply in practice due to the difficulties in ontology construction and semantic annotation. This paper aims to help service requesters (e.g., SBS designers) obtain relevant services accurately with a keyword query by exploiting domain knowledge about service functionalities (i.e., service goals) mined from textual descriptions of services. We firstly extract service goals from services' textual descriptions using an NLP-based method and cluster service goals by measuring their semantic similarities. A query expansion approach is then proposed to help service requesters refine initial queries by recommending similar service goals. Finally, we develop a hybrid service discovery approach by integrating goal-based matching with two practical approaches: keyword-based and topic model-based. Experiments conducted on a real-world dataset show the effectiveness of our approach.

1. Introduction

Recent years have witnessed a software development paradigm transformation from component-based to service-based due to the broad adoption of service-oriented architecture (SOA) and its related technologies (He et al., 2014). The service-based paradigm can not only reduce the cost, time, and efforts required for software development, but also promote the reusability, agility, and quality of the resulting systems (Bano et al., 2014). A large number of software systems have been developed by discovering and composing loosely-coupled Web services provided by different organizations (He et al., 2016).

Many companies like Google, Facebook, and Amazon have encapsulated some of their capabilities as Web services and published them through Web service registries such as ProgrammableWeb[1] (PW) and Mashape.[2] For example, as of June 1, 2017, more than 17,600 and 6,000 services have been registered at PW and Mashape, respectively. These services can be used to create value-added services (referred to as composite services or mashups) or other service-based systems (SBSs). Generally speaking, the process of building an SBS consists of three stages (He et al., 2016): 1) *system planning*, which determines tasks needed to implement the functionalities of the SBS; 2) *service discovery*, which identifies a set of candidate services for each task; and 3) *service selection*, which selects one service from each set of candidate services to fulfil the constraints of system quality, e.g., reliability and throughput. In this research, we focus on the service discovery stage, i.e., *how to retrieve a set of candidate services that can perform a specific task from a service registry?*

There are many semantically similar functionalities provided by different services in a registry. The keyword-based search technology adopted by existing service registries is insufficient to discover services whose functionalities are similar to a query (e.g., the keyword description of a task) (Chen et al., 2013; Wang et al., 2017). For example, many services in PW provide similar functionalities like "book hotel," "book lodge," and "book accommodation." Using any one of these functional keywords as a query, services that contain similar functionalities are hard to be retrieved by PW. As an example, we submitted "book lodge" to the PW service search engine, but the search engine returned only six services that ex-

* Corresponding author.
E-mail addresses: nengzhang@whu.edu.cn (N. Zhang), jianwang@whu.edu.cn (J. Wang), ytma@whu.edu.cn (Y. Ma), hekeqing@whu.edu.cn (K. He), zhengli_hope@whu.edu.cn (Z. Li), frankliu@uark.edu (X. Liu).

[1] https://www.programmableweb.com/.
[2] https://www.mashape.com/.

https://doi.org/10.1016/j.jss.2018.04.046

plicitly contain words *book*/*booking* and *lodge*/*lodging* in their descriptive data.

There are two essential elements in service discovery: the queries specified by service requesters and the service discovery approaches adopted by service registries. Accordingly, to address the drawback of keyword-based technology, two research directions are dedicated to discovering similar services.

One research direction is to expand queries with relevant concepts extracted from lexical databases (e.g., WordNet[3] (Miller, 1995)) or domain ontologies. The primary limitation of existing query expansion approaches (Kokash et al., 2006; Aljoumaa et al., 2011; Paliwal et al., 2012; Ma et al., 2013) is that they purely rely on external knowledge bases and do not leverage the local knowledge about service registries. Therefore, the expanded queries may be ineffective for service discovery in a specific registry. For example, the highly relevant concepts (including the synonyms, hypernyms, and hyponyms) of *book* and *hotel* that can be extracted from WordNet are

book: reserve, record, enter, hold;

hotel: lodge, hostel, building, edifice.

Many of the extracted concepts are irrelevant, e.g., *record, enter, hold, building,* and *edifice*, while they do not include some desired concepts, e.g., *accommodation*, which will result in retrieving many unnecessary services and missing some relevant ones. Similarly, some concepts extracted from domain ontologies may also be irrelevant while some relevant ones are missing. Moreover, ontology-based query expansion approaches are limited by the fact that there is usually no suitable domain ontology at hand.

The other research direction is to develop semantics-aware service discovery approaches. The basic idea is to describe services and queries using ontology-based semantic Web service description languages, e.g., SAWSDL[4] (Semantic Annotations for WSDL and XML Schema), OWL-S[5] (Ontology Web Language for Services), and WSMO[6] (Web Service Modeling Ontology), and to design logic-based reasoning algorithms for retrieving similar services (Crasso et al., 2011; Klusch et al., 2016). This principle has developed a large number of methods and techniques (Klusch and Kaufer, 2009; Wei et al., 2011; García J et al., 2012; Roman et al., 2015; Rodriguez Mier et al., 2016; Chen et al., 2017a), which have been shown to be more useful than the keyword-based technology. However, these approaches are difficult to apply due to several factors (Crasso et al., 2011; Aznag et al., 2014; Cassar et al., 2014): 1) constructing and maintaining ontologies may be difficult, 2) it requires considerable efforts in manually annotating services and queries using ontology-based description languages, and 3) logic-based reasoning algorithms usually lead to high complexity. Alternatively, there are many non-logic-based semantics-aware service discovery approaches (Wang et al., 2017; Aznag et al., 2014; Cassar et al., 2014; Li et al., 2014; Naim et al., 2016) proposed by leveraging latent topic/factor models, e.g., Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999), Latent Dirichlet Allocation (LDA) (Blei et al., 2003), and Bi-term Topic Model (BTM) (Cheng et al., 2014). These approaches can help retrieve similar services; however latent topics learned by these models are too coarse to precisely match services to queries, which limits the performance.

In Zhang et al. (2017), we proposed an approach to extracting service goals (i.e., service functionalities), e.g., <book, hotel, null> and <retrieve, pricing information, {for booking hotel}>, from textual descriptions of services. In this paper, we propose a service discovery approach by utilizing the service goals, aiming to help service requesters obtain similar services accurately with a simple keyword query. The key part of our approach is a goal-oriented query expansion approach. Given a query, a set of semantically similar service goals is recommended. Informed by the recommendations, service requesters can gain a better understanding of service functionalities relevant to their functional requirements and select some desired goals as an expanded query. The contributions of this research are outlined below:

1. We propose a method for service goal clustering. Similarities between service goals are measured based on the semantic similarities of words in WordNet. Moreover, we distinguish the importance of three different types of words in service goals, which contributes to more accurate similarities.
2. We propose a goal-oriented query expansion approach based on service goal clusters. Faced with a query, service goals assigned to the clusters that are similar to the query are recommended, from which the service requester can expand the query by selecting some appropriate goals. Note that we do not adopt an automatic strategy that directly uses all recommended goals as an expanded query without the requester's involvement because some of the recommendations may not be appropriate for the requester, which will lead to returning irrelevant services.
3. We propose a hybrid service discovery approach based on the expanded query. The approach integrates a goal-based service matching mechanism with two widely adopted approaches: a keyword-based approach and an approach based on the LDA topic model.

We conducted experiments on a real-world service dataset crawled from PW. Fourteen subjects were recruited to build a set of 21 queries and evaluate the recommended service goals and retrieved services for each query. According to the evaluation results, the proposed goal-oriented query expansion approach can efficiently recommend semantically similar service goals for queries; and the proposed service discovery approach achieves better performance than several existing approaches.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 gives an overall framework of the proposed service discovery approach. In Section 4, we briefly introduce our previous service goal extraction approach and describe the method for service goal clustering. Section 5 describes the process of service discovery based on goal-oriented query expansion. Section 6 presents the experiments and evaluations. Section 7 discusses the contributions, limitations, and threats to validity. Section 8 concludes this work and introduces our future work.

2. Related work

2.1. Web service discovery

Web Service discovery refers to finding services that can satisfy functional requirements specified by a user query. As one of the core techniques in SOA, it has been studied extensively in the past two decades. Most existing works on service discovery can be categorized into two groups: syntactic approaches (Dong et al., 2004; Wang and Stroulia, 2003; Cong et al., 2015) and semantics-aware approaches (Wang et al., 2017; Aznag et al., 2014; Klusch and Kaufer, 2009; Wei et al., 2011; García J et al., 2012; Roman et al., 2015; Rodriguez Mier et al., 2016; Cassar et al., 2014; Li et al., 2014; Naim et al., 2016; Lu et al., 2016; Chen et al., 2017b).

Syntactic approaches mainly discover services through matching the keywords of services with those of queries using information retrieval (IR) techniques, e.g., vector space model (VSM) and term frequency-inverse document frequency (TF-IDF) (Manning et al., 2009). Although syntactic approaches can benefit from IR techniques and have incorporated several enhancement methods, e.g.,

[3] http://wordnet.princeton.edu/.
[4] https://www.w3.org/TR/sawsdl/.
[5] https://www.w3.org/Submission/OWL-S/.
[6] https://www.w3.org/Submission/WSMO/.

structural matching of WSDL (Wang and Stroulia, 2003) and clustering algorithms (Dong et al., 2004; Cong et al., 2015), they still suffer from low precision and recall (Plebani and Pernici, 2009).

Semantics-aware approaches attempt to overcome the drawback of syntactic approaches by searching for semantically similar services of queries. Existing studies can be divided into two subcategories: *logic-based* (Klusch and Kaufer, 2009; Wei et al., 2011; García J et al., 2012; Roman et al., 2015; Rodriguez Mier et al., 2016; Chen et al., 2017a) and *non-logic-based* (Wang et al., 2017; Aznag et al., 2014; Cassar et al., 2014; Li et al., 2014; Naim et al., 2016; Lu et al., 2016; Chen et al., 2017b). The key idea of logic-based approaches is to describe services using ontology-based semantic Web service description languages (e.g., SAWSDL, OWL-S, and WSMO) and design logic-based reasoning algorithms for service retrieval. For example, Klusch et al. proposed service matchmakers for different types of semantic services, e.g., SAWSDL-MX (Klusch et al., 2009a), OWLS-MX (Klusch et al., 2009b), and WSMO-MX (Klusch and Kaufer, 2009). Wei et al. (2011) proposed a customizable SAWSDL service matchmaker that can support several matching strategies according to different application requirements by extending XQuery with various similarity measures. In García J et al. (2012), an approach was proposed to filter services by adding a preprocessing stage based on SPARQL queries. It has been demonstrated that logic-based approaches can achieve good performance because of the accurate descriptions of services and queries. However, they are difficult to apply because considerable efforts are required for specifying and managing ontologies, and manually annotating services and queries using semantic description languages (Crasso et al., 2011).

Recently, some non-logic-based approaches (Wang et al., 2017; Aznag et al., 2014; Cassar et al., 2014; Li et al., 2014; Naim et al., 2016) have been proposed to retrieve similar services by utilizing latent topic models. For example, Cassar et al. (2014) used PLSA and LDA to extract latent topics from service descriptions and proposed a service matchmaker based on the topic distributions of services. Wang et al. (2017) adopted BTM to learn latent topics from services and introduced a concept "common topic group (CTG)" to organize the services that share multiple topics. A service discovery approach was then proposed based on CTG matching. Although these approaches can help obtain similar services, their accuracy is limited by the coarseness of the learned latent topics. Moreover, there are several non-logic-based approaches (Lu et al., 2016), Chen et al. (2017) proposed by leveraging the semantic relationships among words in WordNet. However, as explained in Section 1, some semantic relationships may be unexpected, which affects the performance of these approaches.

In summary, existing syntactic approaches and semantics-aware approaches (including both logic-based and non-logic-based) are challenging to achieve both high effectiveness and practicality. Also, as mentioned previously, it is difficult for users to specify high-quality queries, which may lead to returning poor services. Nevertheless, most of these approaches do not consider improving the quality of user queries. Compared with these works, we propose a novel non-logic-based service discovery approach, which expands user queries based on service goal knowledge extracted from textual service descriptions. In Zhang et al. (2017), we introduced a three-stage algorithm for service goal extraction based on in-depth analysis of linguistic information of various sentences. We investigate in this work how to leverage the extracted service goal knowledge to improve the performance of service discovery.

2.2. Query expansion

Query expansion is a technique that has been widely adopted to improve IR systems such as Web search (Ghali and Qadi, 2017) and image retrieval (Wang et al., 2017) by enhancing original queries with relevant terms extracted from vocabularies, ontologies, query logs, and so on.

To the best of our knowledge, only a limited number of works pay attention to query expansion for service discovery (Kokash et al., 2006; Aljoumaa et al., 2011; Paliwal et al., 2012; Ma et al., 2013). For example, Kokash et al. (Kokash et al., 2006) expanded keywords in WSDL descriptions and queries using their synonyms in WordNet. Aljoumaa et al. (2011) reformulated queries expressed regarding goals with verbs and terms extracted from domain ontologies. Paliwal et al. (2012) enhanced queries by leveraging relevant concepts in domain ontologies. Ma et al. (2013) proposed a query expansion approach that simultaneously uses WordNet and domain ontologies to find relevant terms of a query. The approach consists of two stages: a lexical expansion that uses WordNet to obtain general associated terms and a semantic expansion that uses domain ontologies to deal with domain-specific terms. To sum up, these approaches can enhance user queries to a certain extent by utilizing semantically relevant terms in WordNet or domain ontologies. However, as discussed in Section 1, some concepts extracted from WordNet or domain ontologies may be irrelevant while some relevant ones are missing, which still prevents us from obtaining desired services.

Here, we propose a novel goal-oriented query expansion approach, which differs from existing query expansion approaches in three aspects. Firstly, instead of expanding queries with relevant terms, we expand queries with service goals (in the form of $<Verb + Noun + Prepositional\ Phrases>$). Secondly, while existing approaches mainly depend on external knowledge bases (e.g., WordNet and domain ontologies), our approach leverages the service goal knowledge mined from services; thus the expanded queries are more useful for service discovery. Thirdly, in our approach, service requesters are directly involved in the query expansion process, to ensure that the expanded queries can accurately represent their requirements.

3. Approach overview

Fig. 1 gives an overall framework of the proposed service discovery approach, which consists of two major modules: offline service mining and online service discovery. Several types of knowledge will be mined from a given set of services during the offline module, which will be used to facilitate the online service discovery.

The input of offline service mining is categorized services, which can be provided by most existing service registries. For example, services registered at PW and Mashape are organized by predefined categories. If the services have no predefined category, they need to be categorized using classification techniques, e.g., Support Vector Machine (SVM) (Zhang et al., 2012; Chang and Lin, 2011). Moreover, there may be overlaps among the predefined categories, e.g., *Travel, Hotels*, and *Booking* in PW, which will lead to the missing of some relevant services if the scope of service search is restricted to the category that is most similar to a query. To address this issue, the category assignments of services can be adjusted by performing service categorization.

Once the services are well categorized, two steps will be executed on the set of services in each domain. (Note that we use the two terms "category" and "domain" interchangeably in this paper.) Firstly, service goals are extracted from the textual description of each service using a natural language processing (NLP)-based method proposed in our previous work (Zhang et al., 2017). Secondly, the set of service goals extracted from all services is grouped into clusters using clustering techniques, e.g., the overlapping *K*-Means algorithm (Khanmohammadi et al., 2017). Similarities between service goals are measured based on the word similarities in WordNet. In particular, to measure the similarities more accurately,

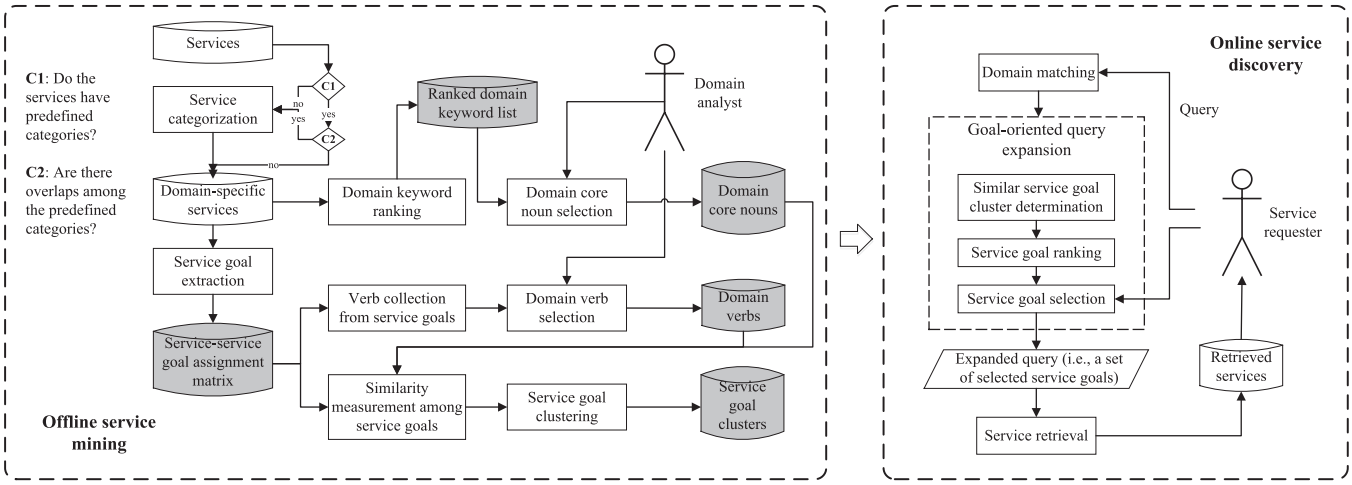


Fig. 1. Overview of the proposed service discovery approach.

we distinguish the importance of three different types of words in service goals. For this task, a set of core nouns and a set of verbs are built for each domain. As for the set of core nouns, domain analysts select them from a ranked domain keyword list generated using some well-known techniques, e.g., TF-IDF. Similarly, domain analysts select the set of verbs from a verb list collected from service goals. Section 4 presents more details of offline service mining.

The online service discovery process will be initialized when a service requester submits a query to the Web service search engine. In the first step, the similarity between each domain and the query is computed, and the domain with the maximum similarity is chosen as the target domain of the query. Afterward, a query expansion process is conducted by employing service goal clusters of the target domain as follows. A set of service goal clusters that are similar to the query is determined first; then service goals assigned to the similar clusters are ranked and recommended by measuring their similarities to the query; afterward, the requester can select some desired goals from the recommendations as an expanded query. Finally, a set of services is retrieved by matching each service in the target domain with the expanded query. Section 5 presents more details of online service discovery.

4. Offline service mining

In this section, we focus on describing two main steps in the offline service mining module, namely service goal extraction and service goal clustering for services in each domain.

In our prior work (Wang et al., 2013), according to the studies on goal modeling (Aljoumaa et al., 2011; Rolland et al., 1998), a concept "service goal" is introduced to represent the functionality of a service. Formally, a service goal is defined as a triple *sg* = < *sgv, sgn, sgp* >, *where sgv is a verb* (*phrase*) *that denotes the action of sg; sgn is a noun* (*phrase*) *that denotes the entity affected by the action; and sgp is an optional set of parameters that denote additional information such as the initial or final state of the entity, and how the action affects the entity.* The parameters in *sgp* are typically represented in the form of prepositional phrases. For example, for a sentence (referred to as *esen*) contained in the textual description of API *DealAngel* registered at PW: "*DealAngel allows users to search for hotels by location, retrieving pricing and reservation information for booking hotels,*" it contains three service goals: <search for, hotel, {by location}>, <retrieve, pricing information, {for booking hotel}>, and <retrieve, reservation information, {for booking hotel}>.

4.1. Service goal extraction

The textual description of services usually contains several sentences. To extract service goals from textual service descriptions, it is necessary to obtain the grammatical structure of sentences. In Zhang et al. (2017), we parsed the sentences using the Stanford Parser[7] (Marneffe et al., 2006), an NLP tool that can generate accurate linguistic analysis for most sentences we may encounter (Stevenson and Greenwood, 2006). Two kinds of linguistic information are generated for each sentence: POS tags and Stanford typed dependencies, e.g., the results of sentence *esen* depicted in Fig. 2(a). The Stanford typed dependency relations such as dobj, conj_and, and prep_for can refer to Marneffe et al. (2006); and the POS tags such as NN, VB, and VBZ are defined in Marcus et al. (1993). For example, the dependency dobj(*retrieving*-11, *information*-15) in Fig. 2(a) indicates that the word *retrieving* is the accusative object of *information*; 11 and 15 are the position indexes of *retrieving* and *information*, respectively, in *esen*.

We proposed in Zhang et al. (2017) an approach to extracting service goals from a sentence based on its linguistic information. The approach contains three stages according to the insights gained through in-depth analysis of various sentences: 1) there can be several service goals contained in a sentence, e.g., *esen*; 2) the skeletons of some service goals (referred to as "initial goals") can be extracted directly from a few particular dependencies; and 3) the final service goals can be obtained by extending initial goals using other dependencies. The three stages are briefly introduced as follows.

Initial goal generation: A set of initial goals is generated from three dependencies using the following rules: 1) nsubjpass(*a, b*) → <*a, b*, null>, 2) dobj(*a, b*) → <*a, b*, null>, and 3) prep(*a, b*) → <*a, b*, null>. For example, two initial goals <retrieving, information, null> and <search, hotels, null> are generated from the dependencies in Fig. 2(a) using rules 2) and 3), respectively.

Initial goal extension: Initial goals are extended using dependencies such as amod, nn, prep, and conj to obtain some meaningful information related to them. For example, the prefix modifier *pricing* of *information* (i.e., the *sgn* part of an initial goal <retrieving, information, null>) is obtained from nn(*information*-15, *pricing*-12). Some potential service goals are also obtained from the extension results of initial goals. For example, the coordinate noun *reservation* of *pricing* identified from conj_and(*pricing*-12, *reservation*-14)

[7] https://nlp.stanford.edu/software/lex-parser.shtml.

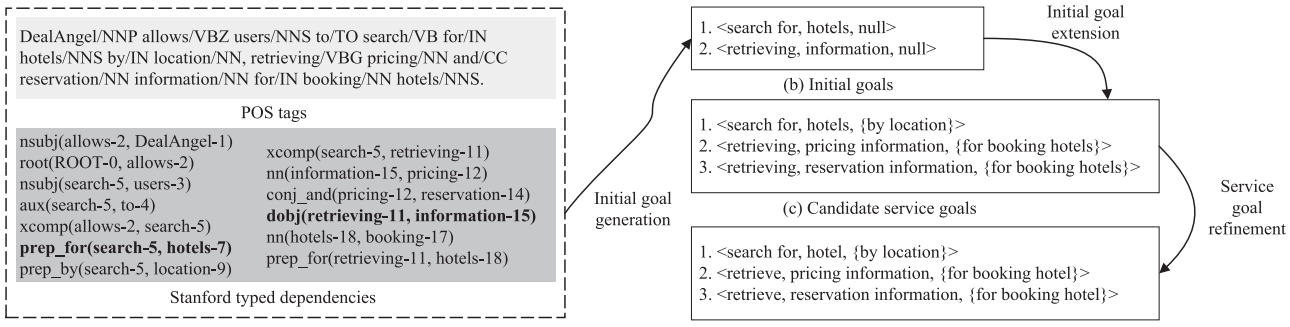


Fig. 2. Service goal extraction from sentence *esen*.

Table 1
Five service goals in the *travel* domain.

sg_1	<book, hotel, null>
sg_2	<book, lodging, null>
sg_3	<retrieve, pricing information, {for booking hotel}>
sg_4	<retrieve, hotel price, null>
sg_5	<retrieve, hotel information, null>

contributes to a new candidate service goal, as shown in Fig. 2(c). The output of this stage is a set of candidate service goals.

Service goal refinement: Candidate service goals are refined by performing lemmatization and stop word removal using the NLTK[8] toolkit (Bird et al., 2009).

The service goals of a Web service are obtained by collecting the service goals extracted from all sentences in its textual description. After extracting service goals for each service in a domain d, a service-service goal assignment matrix is built as $SSGAM_d \subseteq S_d \times SG_d$, where S_d is the set of services in d and SG_d is the entire set of service goals extracted from all services in S_d. Each entry $SSGAM_{d|s,\ sg} \in \{0, 1\}$, where $s \in S_d$ and $sg \in SG_d$. $SSGAM_{d|s,\ sg} = 1$ indicates that service s contains service goal sg.

4.2. Service goal clustering

Observing the service goals extracted for domain-specific services in PW, we find that many similar service goals share a basic functionality in a domain. The basic functionality of a service goal can be represented by the verb(s) and the nouns that are highly relevant to the corresponding domain, while the other words (including adjectives and the nouns with lower domain representation) can be viewed as the refinement of the basic functionality. Table 1 presents five example service goals in the *travel* domain. Apparently, they can be classified into two groups: $\{sg_1, sg_2, sg_3\}$ that share basic functionality "book hotel" ("book lodging" is semantically equivalent to "book hotel") and $\{sg_3, sg_4, sg_5\}$ that share basic functionality "retrieve hotel." Similar service goals can be grouped to provide comprehensive views of domain-specific service functionalities. Because a service goal, e.g., sg_3, may belong to more than one group, it is better to cluster service goals using overlapping clustering algorithms (where an instance can be assigned to multiple clusters).

The key of service goal clustering is to measure the similarities between service goals. We devise a method for this task based on the semantic similarities of words in WordNet. As claimed in Garg et al. (2015), based on a survey of the literature, asymmetric similarity measures can obtain better performance than symmetric similarity measures in many applications such as information retrieval and text clustering. This finding applies to the similarity measurement between service goals. Based on our observation, given two service goals sg_i and sg_j, the similarity of sg_i to sg_j is usually not equal to that of sg_j to sg_i. For example, the similarity of sg_5 to sg_3 (in Table 1) should be lower than that of sg_3 to sg_5 because sg_3 contains all words of sg_5 while sg_5 lacks some crucial words of sg_3, i.e., *booking* and *pricing*. Therefore, we adopt an asymmetric way to calculate the similarities between service goals. More specifically, the similarity of sg_i to sg_j is calculated using

$$SGSim^{asy}(sg_i,\ sg_j) = \frac{\sum_{w_j \in W(sg_j)} \max_{w_i \in W(sg_i)} \{wsim(w_i,\ w_j)\}}{|W(sg_j)|}, \quad (1)$$

$$wsim(w_i,\ w_j) = \begin{cases} 1, & \text{if } stem(w_i) \text{ equals } stem(w_j) \\ WNSim(w_i,\ w_j), & \text{otherwise} \end{cases}, \quad (2)$$

where $W(sg)$ denotes the set of words (except prepositions) in service goal sg; $|W(sg)|$ represents the number of words in $W(sg)$; $stem(w)$ obtains the stem of word w using the Porter stemming algorithm (Porter, 2006), e.g., $stem(booking) = book$; and $WNSim(w_i, w_j)$ denotes the semantic similarity between words w_i and w_j in WordNet. The first branch of Eq. (2) is used to compute the similarity between some words with the same stem, which may not be correctly measured by WordNet, e.g., $WNSim(book, booking) = 0.0522$.

Fig. 3(a) shows the asymmetric similarities among the five service goals in Table 1. The cell in row "sg_i" and column "sg_j" represents $SGSim^{asy}(sg_i, sg_j)$ (the similarity of sg_i to sg_j). There are some unexpected similarities. For example, the similarity of sg_4 to sg_3 and the similarity of sg_5 to sg_3 are almost the same, i.e., $SGSim^{asy}(sg_4, sg_3) = 0.6797$ and $SGSim^{asy}(sg_5, sg_3) = 0.6823$. This case is unexpected because the word *price* in sg_4 is more meaningful than the general word *information* in sg_5; and thus $SGSim^{asy}(sg_4, sg_3)$ should be notably higher than $SGSim^{asy}(sg_5, sg_3)$.

To measure the similarities between service goals more accurately, it is desirable to distinguish the importance of different words in service goals. As stated previously, the basic functionality of a service goal can be represented by its verb(s) and domain-specific core nouns (i.e., essential nouns in a domain), while the other words (e.g., adjectives and non-core nouns) are additional modifiers. That is, the other words are less important in identifying the functionality of a service goal. For example, the functionality of sg_5 is mainly dependent on the verb *retrieve* and the core noun *hotel* (in the *Travel* domain), while *information* is a modifier of *hotel*, which contributes little to the service goal. Moreover, according to the analysis in Aljoumaa et al. (2011), domain-specific core nouns are usually more critical than verbs in reflecting the point of concerns for users. For example, when we look for services using query "book hotel," the most significant issue for us

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

	sg_1	sg_2	sg_3	sg_4	sg_5
sg_1	1.0	0.7164	0.5206	0.4263	0.5211
sg_2	0.7164	1.0	0.4072	0.2373	0.332
sg_3	1.0	0.7164	1.0	1.0	1.0
sg_4	0.6395	0.3559	0.6797	1.0	0.7831
sg_5	0.7816	0.498	0.6823	0.7831	1.0

(a) Similarities calculated using Eq. (1)

	sg_1	sg_2	sg_3	sg_4	sg_5
sg_1	1.0	0.7996	0.45	0.3333	0.6
sg_2	0.7996	1.0	0.3758	0.2509	0.4197
sg_3	1.0	0.7996	1.0	1.0	1.0
sg_4	0.6667	0.4663	0.75	1.0	0.9
sg_5	0.6667	0.4663	0.55	0.6667	1.0

(b) Similarities calculated using Eq. (6) with λ_1=0.3, λ_2=0.6, and λ_3=0.1

Fig. 3. Two types of similarities among five service goals.

Table 2
Three different types of words in five service goals.

sg_i	$V_{Travel}(sg_i)$	$CN_{Travel}(sg_i)$	$Oth_{Travel}(sg_i)$
sg_1	{book}	{hotel}	{}
sg_2	{book}	{lodge}	{}
sg_3	{book, retrieve}	{hotel, price}	{information}
sg_4	{retrieve}	{hotel, price}	{}
sg_5	{retrieve}	{hotel}	{information}

is not *book*, but *hotel*. Services with functionalities like "list hotel" and "compare hotel," are more relevant than those with "book flight ticket" and "book car." Based on the above analysis, we identify three different types of words in service goals of a domain: verbs, core nouns, and the other words; and the important priority among them is suggested to be: core nouns $\succ$ verbs $\succ$ the other words, where $\succ$ means "more important than."

To determine the three different types of words in service goals within a particular domain d, a set of verbs and a set of core nouns (denoted by V_d and CN_d, respectively) need to be predefined. As for V_d, the *sgv* parts of service goals in SG_d can be counted to obtain a verb list; and then V_d can be selected by domain analysts from the verb list. As for CN_d, a ranked list of domain keywords can be generated by applying TF-IDF on the set of services S_d. Afterward, domain analysts define CN_d based on the ranked domain keyword list.

Once V_d and CN_d have been built, the three types of words in each service goal $sg \in SG_d$ are obtained as

$$V_d(sg) = \left\{ v \in V_d \middle| \begin{array}{c} \exists w \in W(sg), \\ stem(w) \text{ equals } stem(v) \end{array} \right\}, \tag{3}$$

$$CN_d(sg) = \left\{ cn \in CN_d \middle| \begin{array}{c} \exists w \in W(sg), \\ stem(w) \text{ equals } stem(cn) \end{array} \right\}, \tag{4}$$

$$Oth_d(sg) = W(sg) - \left\{ w \in W(sg) \middle| \begin{array}{c} \forall w_k \in V_d(sg) \cup CN_d(sg), \\ stem(w) \text{ does not equal } stem(w_k) \end{array} \right\}, \tag{5}$$

where $V_d(sg)$, $CN_d(sg)$, and $Oth_d(sg)$ denote the sets of verbs, core nouns, and the other words contained in sg, respectively. The reason why $V_d(sg)$ and $CN_d(sg)$ are obtained according to the stems of words is that there are morphological variants of words in service goals, e.g., *book* and *booking*. For example, Table 2 presents the three different types of words contained in each service goal in Table 1, given V_{Travel} = {*provide, offer, book, reserve, retrieve, find, get*} and CN_{Travel} = {*travel, trip, hotel, lodge, accommodation, flight, price*}.

Next, the similarity of service goals sg_i to sg_j in domain d is computed using

$$SGSim_d^{asy}(sg_i, sg_j) = \lambda_1 \cdot WSim^{asy}(V_d(sg_i), V_d(sg_j)) + \lambda_2 \cdot WSim^{asy}(CN_d(sg_i), CN_d(sg_j)) + \lambda_3 \cdot WSim^{asy}(Oth_d(sg_i), Oth_d(sg_j)), \tag{6}$$

where λ_1, λ_2, and λ_3 are weight coefficients, $\sum_{i=1}^{3} \lambda_i = 1$. According to the importance priority among the three types of words, it is suggested to satisfy $\lambda_2 > \lambda_1 > \lambda_3$. More analysis of various settings of these three parameters is given in Section 6. $WSim^{asy}(W_1, W_2)$ calculates the similarity of word sets W_1 to W_2:

$$WSim^{asy}(W_1, W_2) = \frac{1}{|W_2|} \cdot \sum_{w_i \in W_2} \max_{w_j \in W_1} \{wsim(w_i, w_j)\}. \tag{7}$$

Fig. 3(b) shows the similarities between the five service goals in Table 1 calculated using Eq. (6) under the setting of λ_1= 0.3, λ_2= 0.6, and λ_3= 0.1. As can be seen, the similarities are more ideal than those depicted in Fig. 3(a), e.g., $SGSim^{asy}(sg_4, sg_3) = 0.75$ is notably higher than $SGSim^{asy}(sg_5, sg_3) = 0.55$.

A service goal similarity matrix is built for domain d by measuring the similarities among all service goals in SG_d, i.e., $SGSim_d^{asy}(sg_i, sg_j)$, $\forall sg_i \in SG_d$, $\forall sg_j \in SG_d$. We then use the overlapping *K*-Means algorithm (Khanmohammadi et al., 2017) to cluster the service goals by taking as input the service goal similarity matrix. Particularly, to obtain better service goal clusters, the best number of clusters and the initial cluster centroids are determined using the *X*-Means algorithm (Dan and Moore, 2000) before applying the overlapping K-Means algorithm.

5. Online service discovery based on goal-oriented query expansion

In this section, we describe the online service discovery module illustrated in Fig. 1, which contains three main steps: domain matching, goal-oriented query expansion, and service retrieval.

5.1. Domain matching

Comparing a query to each service in a large registry is a time-consuming task (Wang et al., 2017; Cassar et al., 2014). An efficient solution is to restrict the scope of service search to a subset of services that are potentially similar to a query. Recall that in our approach services are categorized, we can directly reduce service search space to the domain that is most similar to a given query. We refer to the most similar domain of query q as the target domain of q, denoted by $td(q)$. Since the possible overlaps among domains are processed using service categorization (that is, cross-domain services are assigned to each of their relevant domains), most of the similar services of q are probably assigned to $td(q)$ and can be discovered by just exploring the services in $td(q)$.

Given a query q, the first step of service discovery, i.e., domain matching, is to determine the target domain $td(q)$. q is firstly preprocessed by performing word segmentation, lemmatization, stop word removal, and word frequency count, resulting in a vector representation of q, denoted by $\vec{q}$. Afterwards, the Cosine similarity (Manning et al., 2009) between $\vec{q}$ and the TF-IDF vector of domain

d, denoted by $\vec{TFIDF}_d$, is calculated using

$$DQSim\left(\vec{TFIDF}_d, \vec{q}\right) = \frac{\sum_{w \in W(q) \cap W(d)} f_{q,\ w} \cdot tfidf_{d,\ w}}{\sqrt{\sum_{w_i \in W(q)} f_{q,\ w_i}^2} \cdot \sqrt{\sum_{w_j \in W(d)} tfidf_{d,\ w_j}^2}}, \tag{8}$$

where $W(q)$ denotes the set of words in q; $W(d)$ denotes the set of words contained in all services in d; $f_{q,\ w}$ is the frequency of word w in q; and $tfidf_{d,\ w}$ is the TF-IDF score of w in d.

Before applying Eq. (8), the weights of words in $\vec{q}$ and $\vec{TFIDF}_d$ can be amplified by leveraging domain-specific keywords, i.e., V_d and CN_d, so as to obtain a better similarity measure. More specifically, the weight of word w in $\vec{q}$ and $\vec{TFIDF}_d$ (i.e., $f_{q,\ w}$ or $tfidf_{d,\ w}$), denoted by y, is recomputed as

$$y = y \cdot \lambda_{d,\ w}, \tag{9}$$

$$\lambda_{d,\ w} = \max\left\{\lambda_1 \cdot I_{V_d}(w),\ \lambda_2 \cdot I_{CN_d}(w),\ \lambda_3\right\}, \tag{10}$$

where λ_1, λ_2, and λ_3 are weight coefficients referring to Eq. (6). $I_W(w)$ is the indicator function of word set W, i.e., $I_W(w) = 1$ if word w is in W and 0 otherwise. The amplification factor of w, i.e., $\lambda_{d,w}$, is determined by the highest importance of w in d. Similar to Eqs. (3)-(5), we implement $I_W(w)$ based on the stems of words as

$$I_W(w) = 1,\ \text{if } \exists w_k \in W,\ stem(w) \text{ equals } stem(w_k). \tag{11}$$

After calculating the similarities of all domains to q, the domain with the maximum similarity is chosen as $td(q)$.

5.2. Goal-Oriented query expansion

As stated previously, there can be many services in a registry that provide similar functionalities for a query. The keyword-based technology adopted by existing registries is insufficient to find similar services for queries. Several query expansion approaches (Kokash et al., 2006; Aljoumaa et al., 2011; Paliwal et al., 2012; Ma et al., 2013) have been proposed to solve this issue by automatically enhancing original queries with relevant concepts extracted from knowledge bases like WordNet and domain ontologies. However, a significant limitation of these approaches is that they do not leverage any knowledge about services in a registry; and thus the expanded queries may be ineffective for service search.

We propose a novel goal-oriented query expansion approach based on the mined service goal knowledge. The basic idea is that given a query q, service goals assigned to the clusters of its target domain $td(q)$ that are similar to q can be recommended to help the service requester expand q. For this purpose, we devise a method for measuring the similarity of each service goal cluster SGC_i in $td(q)$ to q. At first, three different types of words contained in SGC_i are obtained by collecting the corresponding types of words in the service goals belonging to SGC_i:

$$V_{td(q)}(SGC_i) = \bigcup_{sg_k \in SGC_i} V_{td(q)}(sg_k), \tag{12}$$

$$CN_{td(q)}(SGC_i) = \bigcup_{sg_k \in SGC_i} CN_{td(q)}(sg_k), \tag{13}$$

$$Oth_{td(q)}(SGC_i) = \bigcup_{sg_k \in SGC_i} Oth_{td(q)}(sg_k), \tag{14}$$

where $V_{td(q)}(SGC_i)$, $CN_{td(q)}(SGC_i)$, and $Oth_{td(q)}(SGC_i)$ denote the sets of verbs, core nouns, and the other words in SGC_i, respectively. $V_{td(q)}(sg_k)$, $CN_{td(q)}(sg_k)$, and $Oth_{td(q)}(sg_k)$ denote the sets of verbs, core nouns, and the other words in service goal sg_k, respectively, which can be obtained using Eqs. (3)–(5) based on $V_{td(q)}$ and $CN_{td(q)}$.

Next, for each set of the three types of words in SGC_i, we set a threshold $\theta \in [0, 1]$ to determine the words that are semantically similar to any word in q:

$$SimV(SGC_i \to q) = \left\{ w_i \in V_{td(q)}(SGC_i) \middle| \begin{matrix} \exists w_j \in W(q), \\ wsim(w_i,\ w_j) \geq \theta \end{matrix} \right\}, \tag{15}$$

$$SimCN(SGC_i \to q) = \left\{ w_i \in CN_{td(q)}(SGC_i) \middle| \begin{matrix} \exists w_j \in W(q), \\ wsim(w_i,\ w_j) \geq \theta \end{matrix} \right\}, \tag{16}$$

$$SimOth(SGC_i \to q) = \left\{ w_i \in Oth_{td(q)}(SGC_i) \middle| \begin{matrix} \exists w_j \in W(q), \\ wsim(w_i,\ w_j) \geq \theta \end{matrix} \right\}. \tag{17}$$

Finally, the similarity of SGC_i to q is calculated as

$$SGCQSim(SGC_i,\ q) = \lambda_1 \cdot \frac{\sum_{w_j \in SimV(SGC_i \to q)} f_{SGC_i,\ w_j}}{\sum_{w_i \in V_{td(q)}(SGC_i)} f_{SGC_i,\ w_i}} + \lambda_2 \cdot \frac{\sum_{w_j \in SimCN(SGC_i \to q)} f_{SGC_i,\ w_j}}{\sum_{w_i \in CN_{td(q)}(SGC_i)} f_{SGC_i,\ w_i}} + \lambda_3 \cdot \frac{\sum_{w_j \in SimOth(SGC_i \to q)} f_{SGC_i,\ w_j}}{\sum_{w_i \in Oth_{td(q)}(SGC_i)} f_{SGC_i,\ w_i}}, \tag{18}$$

where λ_1, λ_2, and λ_3 are weight coefficients referring to Eq. (6) and $f_{SGC_i,\ w}$ is the frequency of word w in SGC_i. The more the words in SGC_i that are semantically similar to q are, the higher $SGCQSim(SGC_i, q)$ is.

After computing the similarities of all service goal clusters to q, the top k most similar clusters are recommended for q. From the recommendations, the requester can better understand the service functionalities related to his/her requirements and select some desired service goals as an expanded query. However, there can be a considerable number of recommended goals, and it will be a time-consuming task for the requester to select appropriate goals. To facilitate the selection, recommended goals can be sorted by their semantic similarities to q. For this purpose, three different types of words in q are separated by

$$V_{td(q)}(q) = \left\{ v \in V_{td(q)} \middle| \begin{matrix} \exists w \in W(q), \\ stem(w) \text{ equals } stem(v) \end{matrix} \right\}, \tag{19}$$

$$CN_{td(q)}(q) = \left\{ cn \in CN_{td(q)} \middle| \begin{matrix} \exists w \in W(q), \\ stem(w) \text{ equals } stem(cn) \end{matrix} \right\}, \tag{20}$$

$$Oth_{td(q)}(q) = W(q) - \left\{ w \in W(q) \middle| \begin{matrix} \forall w_k \in V_{td(q)}(q) \cup CN_{td(q)}(q), \\ stem(w) \text{ does not equal } stem(w_k) \end{matrix} \right\}, \tag{21}$$

where $V_{td(q)}(q)$, $CN_{td(q)}(q)$, and $Oth_{td(q)}(q)$ denote the sets of verbs, core nouns, and the other words in q, respectively.

The similarity of a recommended service goal sg to q is then calculated using

$$SGQSim_{td(q)}(sg,\ q) = \lambda_1 \cdot WSim^{asy}\left(V_{td(q)}(sg),\ V_{td(q)}(q)\right) + \lambda_2 \cdot WSim^{asy}\left(CN_{td(q)}(sg),\ CN_{td(q)}(q)\right) + \lambda_3 \cdot WSim^{asy}\left(Oth_{td(q)}(sg),\ Oth_{td(q)}(q)\right), \tag{22}$$

where λ_1, λ_2, and λ_3 are weight coefficients referring to Eq. (6).

In reality, some of the service goals recommended for q can be more relevant than others. For example, all the five service goals in Table 1 could be recommended for "book hotel," and obviously sg_1,

sg_2, and sg_3 are more relevant than sg_4 and sg_5. Referring to this point, service requesters are allowed to assign preference ratings to their selected service goals, such that the final expanded result of q can be represented as $SG(q)=(sg_i, r_i)$, where r_i is the requester's rating on service goal sg_i. A higher r_i means that *the requester more prefers* sg_i.

5.3. Service retrieval

In this step, a set of services is retrieved from the target domain $td(q)$ according to the expanded query, i.e., $SG(q)$.

A direct way of service retrieval is to match the service goals of each service s with $SG(q)$. Service goals in $SG(q)$ that can be satisfied by s are firstly obtained by

$$SG(s \models q) = \left\{ (sg_i,\ r_i) \in SG(q) \middle|_{W(sg_i) \subseteq W\left(sg_j\right)}^{\exists sg_j \in SG(s),} \right\}, \tag{23}$$

where $SG(s)$ denotes the set of service goals of s, which can be obtained from the service-service goal assignment matrix of $td(q)$, i.e., $SG(s)=\{sg \in SG_{td(q)} | SSGAM_{td(q)|s,\ sg}=1\}$.

The service requester's preference on s is then measured as the requester's maximum rating assigned to service goals in $SG(s \vDash q)$, i.e.,

$$Pref(s \vDash q) = \begin{cases} 0, & \text{if } SG(s \vDash q) = \varnothing \\ \max\limits_{(sg_i,\ r_i) \in SG(s \vDash q)} r_i, & \text{otherwise} \end{cases}. \tag{24}$$

By measuring the requester's preferences on all services in $td(q)$, a ranking list of the services can be generated for q. However, the preferences on some relevant services may not be correctly measured for two reasons. On the one hand, some meaningful service goals cannot be extracted from the textual descriptions of services using our service goal extraction approach (Zhang et al., 2017) due to several factors: 1) service goals expressed as noun phrases, e.g., "*Escapio is a German booking portal for hand-picked, unique hotels*." (excerpted from API *Escapio* in PW), are not handled by the approach; 2) some useful Stanford typed dependency relations, e.g., xcomp, are not covered by the approach; and 3) some complex and informal sentences cannot be correctly parsed using the Stanford Parser, and thus the service goals therein cannot be extracted. On the other hand, not all word similarities in WordNet are suitable to the current context, which will affect the similarities between service goals and the similarities of recommended goals to a query. Therefore, it is difficult to group all similar service goals. Some similar goals of a query may not be included in the top k most similar service goal clusters and fail to be recommended. Moreover, some similar goals may not rank highly in the recommendation list and may thus be missed by the requester.

To address the above issue, we propose to complement the goal-based service discovery approach with some widely used service discovery approaches, e.g., keyword-based and topic model-based. The expanded result of query q is used as the input of the complementary approaches. More specifically, the words of all service goals in $SG(q)$ are collected to obtain a vector $\vec{eq}=(w_i, y_i)$, $w_i \in \cup_{(sg_j,\ r_j) \in SG(q)} W(sg_j)$ and y_i is the weight of word w_i. y_i is computed by integrating three factors: the frequency of w_i in $SG(q)$, the requester's preference rating on service goals where w_i exists, and the importance of w_i in domain $td(q)$, using

$$y_i = \left(\sum_{(sg_j,\ r_j) \in SG(q)} f_{sg_j,\ w_i} \cdot r_j \right) \cdot \lambda_{td(q),\ w_i}, \tag{25}$$

where $f_{sg_j,\ w_i}$ is the frequency of w_i in service goal sg_j; r_j is the requester's rating on sg_j; and $\lambda_{td(q),\ w_i}$ denotes the importance of w_i in $td(q)$, which can be set using Eq. (10) based on $V_{td(q)}$ and $CN_{td(q)}$.

Two kinds of similarities between each service s in $td(q)$ and $\vec{eq}$ are then computed using a keyword-based approach and an approach based on LDA, respectively. As for the keyword-based approach, we calculate the Cosine similarity between $\vec{eq}$ and the word frequency vector of s, denoted by $\vec{s}$. Note that the weights of words in $\vec{s}$ are amplified using Eq. (9) based on the importance of words in $td(q)$. As for the LDA-based approach, we perform the LDA algorithm on the services contained in all domains, denoted by S, to obtain two probability distributions: the *service-topic distribution* $P(t|s_i)$ ($t \in \{1, \ldots, T\}$, where T is the number of latent topics) that captures the probability distribution of each service $s_i \in S$ over topics and the *topic-word distribution* $P(w|t)$ ($w \in W$, where W is the set of words contained in services S) that captures the probability distribution of each topic t over words. Based on the learned LDA model, the similarity between s and $\vec{eq}$ is calculated using Eq. (26) (Naim et al., 2016).

$$SQLSim\left(\vec{eq}|s\right) = \prod_{w_i \in W\left(\vec{eq}\right)} \left(\sum_{t=1}^{T} P(w_i|t) P(t|s) \right)^{y_i}, \tag{26}$$

where $W(\vec{eq})$ denotes the set of words contained in $\vec{eq}$.

We finally obtain the overall similarity between s and q by aggregating the three kinds of similarities:

$$\begin{aligned} Sim(s,\ q) &= \gamma_1 \cdot Pref(s \models q) \\ &\quad + \gamma_2 \cdot SQCSim\left(\vec{s}, \vec{eq}\right) + \gamma_3 \cdot SQLSim\left(\vec{eq}|s\right), \end{aligned} \tag{27}$$

where $SQCSim(\vec{s}, \vec{eq})$ denotes the Cosine similarity between $\vec{s}$ and $\vec{eq}$; and γ_1, γ_2, and γ_3 are weight coefficients associated with the three similarities. Before applying Eq. (27), $Pref(s \vDash q)$, $SQCSim(\vec{s}, \vec{eq})$ and $SQLSim(\vec{eq}|s)$ are normalized to [0, 1], respectively, with respect to their corresponding maximum similarities.

6. Experiments

We conducted a series of experiments to evaluate the proposed approach. All experiments were developed in Java, and conducted on a PC with 3.6 GHz Intel(R) Core(TM) i7 CPU and 8GB RAM.

6.1. Dataset and preprocessing

The publically accessible service registry PW was used as our testbed. We collected the descriptive data including the name, tags (i.e., primary category and secondary categories), and textual descriptions of 13,520 Web APIs from PW on July 24, 2016.[9] The APIs were divided into different domains according to their tags. We selected seven domains: *Mapping* (747), *Music* (256), *Photos* (363), *Transportation* (363), *Travel* (357), *Video* (441), and *Weather* (154), for experimentation according to their scale and popularity. The numbers, e.g., 747 and 256, indicate the scale (i.e., the number of APIs) of the corresponding domains. The popularity of a domain was measured according to the proportion of APIs in the domain that have been used by mashups in PW.

Apparently, there are overlaps among the selected domains. For example, the mapping service is vital for active and intelligent transportation; and a good travel plan requires a guide map and the transportation information about places. There can be APIs relevant to more than one of the *Mapping, Transportation*, and *Travel* domains. Since users assign the tags of APIs (e.g., PW managers and API providers), it is difficult to find out

[9] The PW API dataset and the extracted services goals used in this work are published at: http://software.whu.edu.cn/webapi/.

all relevant tags for an API from hundreds of candidate tags (467 tags are used for categorizing APIs in PW). As an example, API *The Airport Guide* contains a piece of description: "*Current weather forecast information is also provided, along with remarks particular to the airport such as local hazards to navigation. Enhanced membership provides information about local hotels* ." The API is only tagged by "Transportation" and "Air Travel," while some relevant tags are not assigned to it, e.g., "Travel" (*hotel* is a core noun in the *Travel* domain) and "Weather." This issue will lead to the missing of some relevant APIs when looking up services in the most similar domain of a query. To address this issue, we firstly preprocessed the descriptive data of services by performing word segmentation, lemmatization, and stop word removal; afterward, for each domain d, we categorized all APIs in the other six domains using SVM (Zhang et al., 2012; Chang and Lin, 2011). More specifically, we constructed a training set, which consists of two parts: the original set of APIs in d as domain relevant set and the same number of APIs randomly selected from other domains as domain irrelevant set. Next, an SVM classifier was learned for d from the training set and applied to classify the APIs in the other six domains.

We then performed the following two steps on the set of APIs in each domain d:

Domain keyword ranking: A ranked domain keyword list *RDKL* was generated by applying TF-IDF on the APIs.

Service goal extraction: Service goals were extracted from the textual description of each API using our service goal extraction approach (Zhang et al., 2017), resulting in a service-service goal assignment matrix. A verb list *VL* was then generated by counting the *sgv* parts of all extracted service goals.

6.2. Experiment design

We recruited 14 subjects (including six undergraduates and eight MScs) to evaluate the proposed goal-oriented query expansion and service discovery approach. Before conducting the experiments, the subjects went through a training process. Firstly, we held a meeting with the subjects to introduce the proposed approach and describe the experiment tasks assigned to them: 1) select a set of verbs from the *VL* of each domain; 2) select a set of core nouns from the *RDKL* of each domain; 3) construct a set of experimental queries; 4) evaluate the relevance of service goals for each query; and 5) evaluate the relevance of APIs for each query. Each domain was randomly assigned to two subjects. Afterward, the subjects were given one week to familiarize themselves with the background knowledge of their assigned domains using resources on the Internet, e.g., Wikipedia pages. The APIs and their service goals, *RDKLs*, and *VLs* of all domains were also given to the subjects. After the familiarization process, we held a meeting with the subjects again to perform the experiment tasks described above.

Domain verb selection: For each domain d, the two subjects assigned to d were asked to select a set of verbs independently from the *VL* of d. The debatable ones, if any, were discussed together to build a final set of verbs V_d.

Domain core noun selection: The two subjects assigned to d were asked to independently select no less than 50 core nouns from the *RDKL* of d. A final set of core nouns CN_d was then built by discussing the debatable ones.

Query construction: As the manual creation of the benchmarks of relevant service goals and relevant APIs for a query is an expensive process, we only asked the two subjects assigned to each domain to construct three representative queries. Table 3 presents the 21 queries constructed for the seven selected domains.

Relevance evaluation of service goals for queries: For each of the 21 queries, the two subjects assigned to the domain d of the query were asked to determine the relevance of all service goals in d on a scale of 0–3 ("3"–highly relevant, "2"–relevant, "1"–potentially relevant, and "0"–irrelevant). They evaluated the goals independently and then discussed debatable ones to reach an agreement.

Relevance evaluation of APIs for queries: For each of the 21 queries, the two subjects were also asked to evaluate the relevance of all APIs in the domain of the query on a scale of 0–3 (as defined above) independently. They then discussed the debatable ones to reach an agreement.

Next, for each domain d, the similarities between service goals were computed using Eq. (6) based on V_d and CN_d. According to the analysis in Section 4.2, parameters λ_1, λ_2, and λ_3 used to reflect the importance of three different types of words should be set with respect to $\lambda_2 > \lambda_1 > \lambda_3$. To validate this priority, we tested various settings of these three parameters. Specifically, λ_3 was set from 0.1 to 0.5 with a step size 0.1 (Note that $\lambda_3 \in [0.1, 0.5]$ was sufficient for our validation purpose since the setting of $\lambda_3 > 0.5$ makes no sense according to the above analysis). For each value of λ_3, we constructed all possible combinations of λ_1, λ_2, and λ_3: both λ_1 and λ_2 took values from [0.1, 0.9 - λ_3] with a step size 0.1, while keeping the restriction $\sum_{i=1}^{3} \lambda_i = 1$. Given the service goal similarity matrix produced under each of the settings, we first sought the optimal number of clusters from the range [2, 150] (2 is the possible minimum number of clusters and 150 is the possible maximum number of clusters) using X-Means (Dan and Moore, 2000). Afterwards, we performed the overlapping K-Means algorithm (Khanmohammadi et al., 2017) on the service goal similarity matrix by using the optimal number of clusters and the produced cluster centroids for initialization.

Until now, we finished the offline service mining process illustrated in Fig. 1. Based on the mined knowledge, we performed the online service discovery process for each of the 21 experimental queries, e.g., q, as follows.

Domain matching: q was preprocessed using the same steps adopted in the step of *domain keyword ranking* and represented as a vector by counting the word frequencies. We then calculated the Cosine similarity between the query vector and the TF-IDF vector of each domain (i.e., *RDKL*). The domain with the maximum similarity was chosen as the target domain of q, i.e., $td(q)$. Table 3 presents the top three most similar domains of each query. For example, the target domain of "book hotel" is *Travel* (with similarity 0.2134). As can be seen, the target domains of all the 21 queries match with their source domains.

Goal-oriented query expansion: In the step of *service goal clustering*, we obtained different results of service goal clusters under different settings of λ_1, λ_2, and λ_3. Here, we performed the proposed goal-oriented query expansion approach for query q using each set of service goal clusters produced for $td(q)$. The similarity of each service goal cluster to q was first computed using Eq. (18). λ_1, λ_2, and λ_3 in Eq. (18) were set as the same values used for producing the corresponding set of service goal clusters. To evaluate parameter θ (in Eqs. (15)-(17)) used for determining words that are semantically similar to q, we varied it from 0.1 to 1.0 with a step size 0.1. Based on the similarities of service goal clusters to q obtained under each value of θ, we recommended the service goals assigned to the top k most similar clusters after measuring their similarities to q using Eq. (22). λ_1, λ_2, and λ_3 in Eq. (22) were set as the same values as in Eq. (18). k was varied from 1 to the total number of clusters in $td(q)$.

Service retrieval: Because users are usually only interested in the top entries returned by search algorithms, we collected the relevant service goals (i.e., the goals that were evaluated as 1, 2, or 3) in the top 50 of the goal recommendation list generated for q (with a suitable setting: $\lambda_1 = 0.3$, $\lambda_2 = 0.6$, $\lambda_3 = 0.1$, $\theta = 0.9$, and

Table 3
Twenty-One Experimental Queries.

	Query	Source domain	Top three most similar domains
1	create map	Mapping	Mapping (0.6506), Transportation (0.0038), Video (0.0019)
2	provide GPS		Mapping (0.0042), Transportation (0.0027), Travel (0.0017)
3	identify place		Mapping (0.0282), Photos (0.0007), Music (0.0003)
4	search playlist	Music	Music (0.044), Video (0.0083), Travel (0.0059)
5	share music		Music (0.7746), Video (0.0099), Photos (0.0042)
6	get artist		Music (0.0766), Photos (0.0022), Video (0.0022)
7	create image	Photos	Photos (0.2121), Mapping (0.0066), Video (0.0019)
8	print photo		Photos (0.5923), Music (0.0004), Video (< 0.0001)
9	upload photo		Photos (0.5599), Video (0.0178), Music (0.0016)
10	search parking	Transportation	Transportation (0.0239), Travel (0.0069), Mapping (0.0033)
11	find bus stop		Transportation (0.1435), Travel (0.0067), Mapping (0.0019)
12	get traffic information		Transportation (0.0652), Mapping (0.0075), Travel (0.0063)
13	find airport	Travel	Travel (0.0413), Transportation (0.0245), Mapping (0.0019)
14	search flight		Travel (0.1018), Mapping (0.0033), Music (0.0022)
15	book hotel		Travel (0.2134), Transportation (0.0157), Photos (0.0001)
16	publish video	Video	Video (0.887), Music (0.0023), Photos (0.0001)
17	create video		Video (0.8859), Mapping (0.0021), Photos (0.0018)
18	edit video		Video (0.8859), Photos (0.0086), Mapping (0.0004)
19	provide weather warning	Weather	Weather (0.7317), Transportation (0.0056), Travel (0.0043)
20	forecast weather		Weather (0.8627), Transportation (0.0049), Travel (0.0039)
21	get weather report		Weather (0.7322), Transportation (0.0063), Travel (0.0039)

Table 4
Optimal MAP@N and NDCG@N of query expansion achieved with different settings of λ_1, λ_2, λ_3.

λ_1	λ_2	λ_3	MAP@N					NDCG@N				
			N = 10	N = 20	N = 30	N = 40	N = 50	N = 10	N = 20	N = 30	N = 40	N = 50
0.1	0.8	0.1	0.1718	0.2874	0.3629	0.4146	0.4512	0.7752	0.7688	0.7728	0.7858	0.784
0.2	0.7	0.1	0.1725	0.2914	0.3697	0.4203	0.4624	0.7725	0.767	0.7856	**0.7885**	0.785
0.3	0.6	0.1	0.1722	0.2929	**0.373**	**0.4221**	0.4574	0.7527	0.7693	**0.7882**	0.7873	0.7824
0.4	0.5	0.1	0.1722	0.2771	0.3451	0.3883	0.4271	0.7488	0.749	0.7576	0.7523	0.757
0.5	0.4	0.1	0.1668	0.2633	0.3182	0.3577	0.3834	0.7319	0.7126	0.7092	0.7075	0.7012
0.6	0.3	0.1	0.1595	0.2533	0.3041	0.3375	0.3609	0.7042	0.6889	0.6739	0.6662	0.6586
0.7	0.2	0.1	0.155	0.2437	0.2872	0.3159	0.3404	0.678	0.6587	0.6416	0.6312	0.6301
0.8	0.1	0.1	0.1403	0.23	0.2751	0.2952	0.3192	0.6241	0.6312	0.6146	0.596	0.5953
0.1	0.7	0.2	0.1718	0.2875	0.3637	0.4148	0.4521	**0.7797**	0.769	0.7778	0.7865	0.783
0.2	0.6	0.2	0.172	0.2913	0.3703	0.4205	**0.4675**	0.7605	0.7699	0.7837	0.7881	**0.7913**
0.3	0.5	0.2	0.1721	0.2885	0.3618	0.4103	0.4503	0.7539	0.7654	0.7722	0.776	0.7771
0.4	0.4	0.2	0.1721	0.2678	0.336	0.3784	0.4136	0.7486	0.7278	0.7448	0.7406	0.7397
0.5	0.3	0.2	0.1631	0.2585	0.3101	0.344	0.3701	0.7128	0.6966	0.6876	0.6781	0.6748
0.6	0.2	0.2	0.1542	0.2436	0.2917	0.3233	0.3453	0.6843	0.6639	0.6553	0.6459	0.6405
0.7	0.1	0.2	0.1439	0.2286	0.2738	0.2979	0.321	0.6383	0.6324	0.6168	0.6021	0.6006
0.1	0.6	0.3	0.1718	0.2884	0.3666	0.4161	0.4549	0.7775	**0.7715**	0.7835	0.7863	0.7846
0.2	0.5	0.3	**0.1751**	**0.2933**	0.3709	0.4217	0.4669	0.7607	0.7698	0.7815	0.7853	0.7855
0.3	0.4	0.3	0.1734	0.2833	0.3521	0.3989	0.4353	0.7597	0.7509	0.761	0.7626	0.7608
0.4	0.3	0.3	0.1653	0.2614	0.3161	0.3527	0.3821	0.7322	0.7097	0.704	0.6964	0.6984
0.5	0.2	0.3	0.1547	0.2463	0.2944	0.3283	0.3521	0.6911	0.6747	0.6589	0.6555	0.6507
0.6	0.1	0.3	0.1413	0.228	0.2751	0.3006	0.3226	0.6288	0.6274	0.6203	0.6076	0.6043
0.1	0.5	0.4	0.1715	0.2886	0.367	0.4159	0.4566	0.7745	0.7665	0.7812	0.7812	0.7791
0.2	0.4	0.4	0.1722	0.2924	0.372	0.4215	0.4576	0.757	0.7686	0.7845	0.7814	0.7772
0.3	0.3	0.4	0.1719	0.2666	0.335	0.3776	0.4125	0.7477	0.7241	0.7421	0.7374	0.7367
0.4	0.2	0.4	0.159	0.2529	0.3039	0.3379	0.3613	0.6972	0.6832	0.6726	0.6662	0.6597
0.5	0.1	0.4	0.1415	0.2286	0.2774	0.3016	0.3234	0.6355	0.632	0.6263	0.6107	0.6088
0.1	0.4	0.5	0.1725	0.2911	0.3684	0.4187	0.4602	0.7749	0.7669	0.7798	0.7807	0.7783
0.2	0.3	0.5	0.1727	0.2877	0.3551	0.4043	0.4422	0.7613	0.7621	0.7647	0.7644	0.7656
0.3	0.2	0.5	0.1639	0.2589	0.3126	0.3488	0.3746	0.722	0.7011	0.6943	0.6845	0.6808
0.4	0.1	0.5	0.1451	0.2356	0.282	0.3077	0.3305	0.6484	0.6481	0.6352	0.6253	0.6214
SupportRatio($\lambda_2 > \lambda_1 > \lambda_3$)			84/104	86/104	83/104	83/104	80/104	69/104	77/104	86/104	85/104	79/104

$k = 15$, according to the evaluation results presented in Tables 4, 6, and 7) as an expanded query. Three ranking lists of APIs in $td(q)$ were then generated using the three service discovery approaches described in Section 5.3, respectively. A final ranking list of APIs was generated by combining the three API lists using Eq. (27). We set $\gamma_1 = \gamma_2 = \gamma_3 = 1$, such that the three approaches were equally treated.

Two commonly-used metrics in the information retrieval community, namely MAP@N (mean average precision) and NDCG@N (normalized discounted cumulative gain) (Xia et al., 2015), were used for evaluating the top N of each ranking list of recommended service goals or retrieved APIs for a query.

$$\text{MAP@N} = \frac{1}{|R|}\sum_{i=1}^{N}\left(\frac{n_i}{i}\cdot I(i)\right), \tag{28}$$

where R denotes the relevant set of service goals (or APIs) of the query that were evaluated as 1, 2, or 3; n_i represents the number of relevant service goals (or APIs) in R that exist in the top i of the ranking list; and $I(i)$ indicates whether the service goal (or API) at

Table 5
Numbers of Service Goals and Service Goal Clusters in Seven Domains.

	Mapping	Music	Photos	Transportation	Travel	Video	Weather
Number of service goals	2218	922	885	1361	1279	1354	618
Number of service goal clusters	50	15	23	29	25	39	24

Table 6
MAP@N of Query Expansion Using Different Settings of θ and k (with λ_1= 0.3, λ_2= 0.6, λ_3= 0.1).

θ	MAP@10					MAP@30					MAP@50				
	$k=5$	$k=10$	$k=15$	$k=20$	$k=$all	$k=5$	$k=10$	$k=15$	$k=20$	$k=$all	$k=5$	$k=10$	$k=15$	$k=20$	$k=$all
0.1	0.124	0.161	0.171	0.172	0.172	0.241	0.324	0.37	0.373	0.373	0.292	0.398	0.452	0.456	0.457
0.2	0.137	0.159	0.17	0.171	0.172	0.24	0.319	0.369	0.372	0.373	0.284	0.395	0.453	0.455	0.457
0.3	0.136	0.164	0.17	0.172	0.172	0.249	0.344	0.37	0.373	0.373	0.3	0.426	0.453	0.456	0.457
0.4	0.156	0.164	0.17	0.172	0.172	0.296	0.342	0.37	0.373	0.373	0.356	0.425	0.453	0.457	0.457
0.5	0.157	0.164	0.17	0.171	0.172	0.299	0.346	0.371	0.372	0.373	0.365	0.423	0.454	0.454	0.457
0.6	0.142	0.164	0.17	0.171	0.172	0.291	0.346	0.371	0.372	0.373	0.36	0.43	0.453	0.454	0.457
0.7	0.148	0.168	0.17	0.172	0.172	0.297	0.355	0.371	0.373	0.373	0.364	0.438	0.454	0.456	0.457
0.8	0.163	0.167	0.17	0.172	0.172	0.318	0.36	0.371	0.373	0.373	0.388	0.442	0.453	0.457	0.457
0.9	0.163	0.167	0.17	0.172	0.172	0.326	0.361	0.371	0.373	0.373	0.394	0.444	0.454	0.457	0.457
1.0	0.163	0.167	0.17	0.172	0.172	0.326	0.361	0.371	0.373	0.373	0.394	0.444	0.454	0.457	0.457

the ranking position i is in R.

$$\text{NDCG@N} = \frac{1}{IDCG_N}\sum_{i=1}^{N}\frac{2^{rel_i}-1}{\log_2(1+i)}, \tag{29}$$

where rel_i is the relevance of the service goal (or API) at the ranking position i and IDCG_N represents the maximum possible DCG score through position N that can achieve for the query.

6.3. Evaluation of query expansion

In this section, we report the evaluation results and analysis of query expansion, including three aspects: 1) impact of λ_1, λ_2, and λ_3; 2) impact of θ and k; and 3) comparison of five similarity measures.

(1) **Impact of λ_1, λ_2, and λ_3**

In our proposed goal-oriented query expansion approach, there are five parameters: λ_1, λ_2, λ_3, θ, and k. We tested various settings of λ_1, λ_2, and λ_3 for service goal clustering, to validate the importance priority among them (i.e., $\lambda_2 > \lambda_1 > \lambda_3$). We further tested different settings of $\theta \in [0.1,\ 1.0]$ and k (varied from 1 to the total number of clusters). Under every specific setting of these five parameters, we measured the MAP and NDCG values of the top 50 service goals recommended for each query in Table 3, and then computed the average MAP and NDCG of the 21 queries. Afterwards, we found out the optimal average MAP and NDCG achieved with each setting of λ_1, λ_2, and λ_3, as presented in Table 4. Based on the performance results, we introduced a concept "support ratio" to evaluate the priority (denoted by p), as defined below:

$$SupportRatio(p) = \frac{\sum_{x\in Settings(p),\ y\in Settings(\neg p)} I\{M(x) > M(y)\}}{|Settings(p)||Settings(\neg p)|}, \tag{30}$$

where $Settings(p)$ denotes the set of settings of λ_1, λ_2, and λ_3 that satisfy p, e.g., the setting of $\lambda_1 = 0.3$, $\lambda_2 = 0.6$, and $\lambda_3 = 0.1$; $Settings(\neg p)$ denotes the set of settings that do not satisfy p, e.g., the setting of $\lambda_1 = 0.6$, $\lambda_2 = 0.3$, and $\lambda_3 = 0.1$; $M(s)$ means the performance on a specific metric (e.g., MAP@10 and NDCG@10) achieved with setting s; $I\{e\}$ is an indicator function, that is, if the value of expression e is true, then it returns 1, and 0 otherwise. Intuitively, the support ratio of p considers all pairs of two settings $x \in Settings(p)$ and $y \in Settings(\neg p)$, and measures how many pairs of them are expected, i.e., the performance achieved with x is higher than that achieved with y. A higher $SupportRatio(p)$ indicates that p is more reasonable.

The last row in Table 4 presents the support ratios of $\lambda_2 > \lambda_1 > \lambda_3$. As an example, the value "84/104" means that there are 104 pairs of (x, y), where $x \in Settings(\lambda_2 > \lambda_1 > \lambda_3)$ and $y \in Settings(\neg(\lambda_2 > \lambda_1 > \lambda_3))$, and 84 pairs of them satisfy that the MAP@10 performance achieved with x is higher than that achieved with y. It can be seen that most of the support ratios are around 0.8, which demonstrates that in most cases better goal recommendation lists are generated for queries by setting λ_1, λ_2, and λ_3 according to the priority. Moreover, through analysis, the unexpected results of $M(x) \le M(y)$ are mainly caused by two factors: 1) some word similarities in WordNet are unexpected, which leads to improper similarities of service goal clusters to queries and improper similarities of recommended service goals to queries; and 2) some important nouns of a domain are neglected by the subjects and excluded in the *DCN* of the domain. The optimal performance on each specific metric is marked in bold in Table 4. We can see that the setting of $\lambda_1 = 0.3$, $\lambda_2 = 0.6$, and $\lambda_3 = 0.1$ achieves the best performance on three metrics, i.e., MAP@30, MAP@40, and NDCG@30, which is better than the other settings on the whole. Therefore, the experiment results produced using $\lambda_1 = 0.3$, $\lambda_2 = 0.6$, and $\lambda_3 = 0.1$ were adopted in the following evaluations. Table 5 presents the numbers of extracted service goals and the numbers of service goal clusters generated using the optimal setting of λ_1, λ_2, and λ_3 in each of the seven experimental domains.

(2) **Impact of θ and k**

Now that the optimal setting of important priority among parameters λ_1, λ_2, and λ_3 for our experiments is found out, we further determine the other two parameters, i.e., θ (used to determine the words contained in service goal clusters that are semantically similar to a query) and k (the number of similar service goal clusters to be recommended for a query).

Tables 6 and 7 present average MAP and NDCG of the top 10, 30, and 50 of goal recommendation lists generated for the 21 experimental queries using different settings of θ and k, given the optimal setting of λ_1, λ_2, and λ_3. Columns "$k=$all" show the performance values obtained by recommending all service goal clusters of the target domain of each query. We can see that by setting $k=$all, the MAP (or NDCG) values are the same under different settings of θ, because all service goals are involved in recommendation and no relevant goal is excluded. When $k=5$ or 10, the MAP and NDCG values go through a roughly similar process (except for

Table 7
NDCG@N of Query Expansion Using Different Settings of θ and k (with λ_1= 0.3, λ_2= 0.6, λ_3= 0.1).

θ	NDCG@10					NDCG@30					NDCG@50				
	$k=5$	$k=10$	$k=15$	$k=20$	$k=$all	$k=5$	$k=10$	$k=15$	$k=20$	$k=$all	$k=5$	$k=10$	$k=15$	$k=20$	$k=$all
0.1	0.619	0.701	0.753	0.753	0.753	0.554	0.703	0.779	0.786	0.788	0.539	0.69	0.772	0.779	0.782
0.2	0.571	0.692	0.75	0.752	0.753	0.509	0.684	0.781	0.786	0.788	0.496	0.676	0.777	0.779	0.782
0.3	0.569	0.725	0.75	0.753	0.753	0.516	0.726	0.783	0.786	0.788	0.505	0.726	0.776	0.779	0.782
0.4	0.632	0.722	0.75	0.753	0.753	0.602	0.727	0.782	0.787	0.788	0.593	0.728	0.776	0.78	0.782
0.5	0.676	0.72	0.75	0.751	0.753	0.64	0.736	0.785	0.786	0.788	0.631	0.733	0.778	0.779	0.782
0.6	0.667	0.721	0.75	0.751	0.753	0.647	0.735	0.785	0.786	0.788	0.635	0.734	0.778	0.779	0.782
0.7	0.673	0.736	0.75	0.753	0.753	0.651	0.759	0.785	0.787	0.788	0.637	0.756	0.779	0.78	0.782
0.8	0.671	0.736	0.75	0.753	0.753	0.655	0.762	0.785	0.787	0.788	0.644	0.757	0.778	0.781	0.782
0.9	0.662	0.738	0.75	0.753	0.753	0.657	0.766	0.786	0.787	0.788	0.643	0.763	0.779	0.781	0.782
1.0	0.662	0.738	0.75	0.753	0.753	0.657	0.766	0.786	0.787	0.788	0.643	0.763	0.779	0.781	0.782

some fluctuations) as θ increases from 0.1 to 1.0. More specifically, the MAP (or NDCG) of top 10, 30 or 50 improves relatively faster before a certain point is reached; thereafter the improvement slows down until the performance is stabilized. The lower values at the beginning are mainly caused by the fact that a small θ is unsuitable to determine the similar words of a query, which leads to improper similarities of service goal clusters, so that some similar goals outside the top 5 or 10 most similar clusters are not recommended. When $k \geq 15$, the MAP and NDCG are close to the optimal values achieved with $k=$all, under different settings of θ. This is because that $k=15$ is large enough to include almost all relevant goals of queries even when θ is small. In conjunction with Table 5, we can see that compared with $k=$all, $k=15$ can reduce the time required for generating the goal recommendation list for a query, as there is only a subset of service goals needed to be ranked by computing their similarities to the query. Moreover, from another perspective, when θ is fixed, better MAP and NDCG are achieved with a larger k ($\leq$ 20) since more service goals relevant to queries are included by recommending more service goal clusters.

It is worthy to note that it is difficult to find out the optimal settings of θ and k for all potential queries in practice. Based on the results in Tables 6 and 7, relatively good performance values are achieved for our experimental queries under the settings of $\theta \in [0.1, 1.0]$ and $k \geq 15$, which can be used as a reference for practical applications. In addition, to ensure good performance, θ is not suggested to be a too small value (e.g., < 0.3) or a too large one (e.g., > 0.9). The reasons are explained as follows. A too small θ will not be efficient to determine the similar words of a query and in turn will lead to improper similarities of service goal clusters to the query. A too high θ will have a high probability of ignoring many similar words of a query and thus also affect the similarities of service goal clusters.

In the following evaluations, we used the experiment results produced using $\theta=0.9$ and $k=15$.

(3) Comparison of five similarity measures

A vital step of the proposed goal-oriented query expansion approach is to rank the set of recommended service goals by measuring their similarities to the query. Several similarity measures could be used for this task, such as the classic Jaccard similarity and Cosine similarity (Manning et al., 2009), as well as our proposed semantics-based similarity measure. Recall that we adopt an asymmetric measure and distinguish the importance of three different types of words, to obtain better similarities. To validate the proposed similarity measure, we compared the performance of goal recommendation lists generated using five similarity measures. The details of the five similarity measures used for comparison are given below:

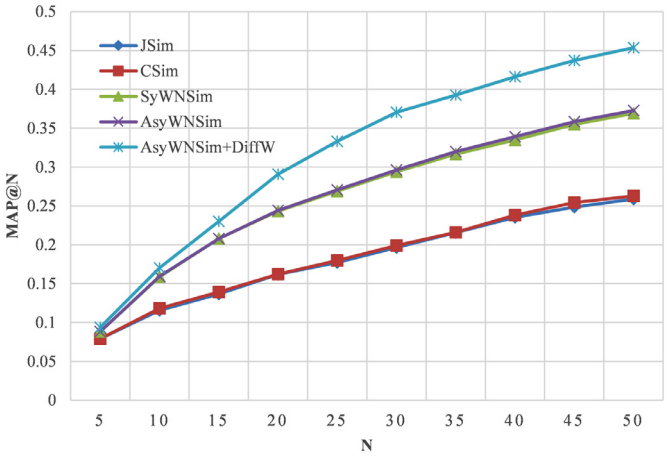


Fig. 4. MAP@N of query expansion using five similarity measures.

1) *Jaccard similarity* (JSim): The Jaccard similarity between service goal sg and query q was calculated as $\frac{|W(sg)\cap W(q)|}{|W(sg)\cup W(q)|}$.
2) *Cosine similarity* (CSim): The Cosine similarity between a service goal and a query was calculated based on their word frequency vectors.
3) *Symmetric semantics-based similarity* (SyWNSim): This measure computed the symmetric similarity of service goal sg to query q based on the word similarities in WordNet. More specifically, $SyWNSim(sg, q)$ is calculated as $WSim^{asy}(W(sg), W(q))$ using Eq. (7) if $|W(q)| \leq |W(sg)|$, and $WSim^{asy}(W(q), W(sg))$ otherwise.
4) *Asymmetric semantics-based similarity* (AsyWNSim): This measure computed the asymmetric similarity of service goal sg to query q based on the word similarities in WordNet, i.e., $AsyWNSim(sg, q) = WSim^{asy}(W(sg), W(q))$.
5) *AsyWNSim improved by distinguishing the importance of three different types of words* (AsyWNSim + DiffW): This is the proposed measure, as demonstrated in Eq. (22). Parameters were set as $\lambda_1=0.3$, $\lambda_2=0.6$, and $\lambda_3=0.1$.

Figs. 4 and 5 show the average MAP and NDCG of goal recommendation lists generated using five similarity measures. The performance values of JSim and CSim are very close and are much worse than those of the other three semantics-based measures. This is because JSim and CSim do not measure the semantic similarities between service goals and queries; therefore they are unable to rank semantically similar goals of a query at higher positions in the goal recommendation list. Among the three semantics-based measures, AsyWNSim + DiffW achieves the best performance, which demonstrates the effectiveness of our proposed similarity measure. Also, AsyWNSim performs only a little

Table 8
Ranking positions of highly relevant service goals of "book hotel" in the goal recommendation lists generated using five similarity measures.

	High relevant service goals	SM1	SM2	SM3	SM4	SM5
1	<book, hotel, null>	1	1	1	1	1
2	<book, flight hotel, null>	2	2	2	2	2
3	<include, hotel booking, null>	3	3	3	3	3
4	<retrieve, hotel book, null>	4	4	4	4	4
5	<enable, user, {to book hotel}>	5	8	5	5	5
6	<include, hotel booking search, null>	6	9	6	6	6
7	<provide, hotel booking, {for hotel in city}>	7	5	7	7	7
8	<provide, booking, {for flight, for hotel, for train}>	8	10	8	8	8
9	<provide, hotel search, {to book accommodation}>	9	11	9	9	9
10	<retrieve, pricing information, {for booking hotel}>	10	12	10	10	10
11	<provide, information booking service, {for car rental, for hotel, for tour}>	86	41	13	13	13
12	<allow, customer, {to book car, to book flight, to book hotel, …}>	87	7	14	14	14
13	<enable, real-time booking, {for car rental, for flight, for hotel, …}>	88	97	15	15	15
14	<implement, back end, {for hotel reservation booking engine, …}>	121	45	16	16	16
15	<cover, wide range, {of industry service including car rental, of industry service including hotel booking, …}>	145	13	17	17	17
16	<provide, global online booking option, {for travel product including choice of hotel, for travel product including choice of insurance, …}>	157	182	18	18	18
17	<cancel, hotel reservation, null>	42	52	94	94	19
18	<connect, hotel reservation, null>	46	56	67	67	20
19	<find, hotel reservation, null>	56	66	65	65	21
20	<offer, access, {to hotel room apartment for short-term rental, to hotel room reservation, …}>	178	46	33	33	22
21	<book, lodging, null>	12	18	54	54	48
22	<explore, book accommodation, null>	51	61	113	113	217

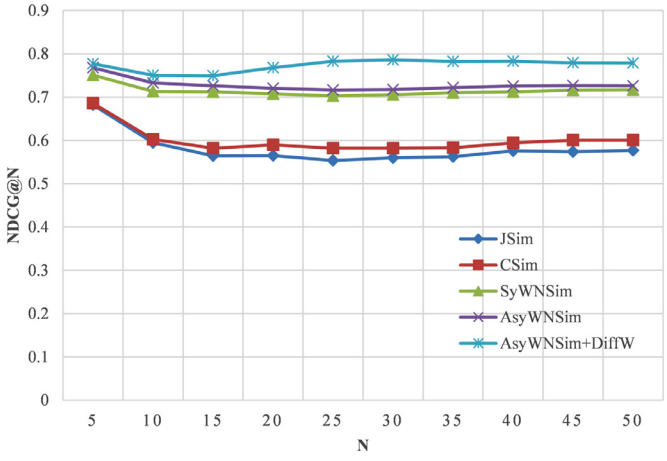


Fig. 5. NDCG@N of query expansion using five similarity measures.

better than SyWNSim. This is because that each of the experimental queries contains only two or three words, which are less than the numbers of words in most service goals. Since when $|W(q)| \leq |W(sg)|$, $SyWNSim(sg, q) = WSim^{asy}(W(sg), W(q))$ (which is equal to $AsyWNSim(sg, q)$), the majority of similarities calculated by SyWNSim are the same with those calculated by AsyWNSim.

For example, Tables 8 and 9 present the ranking positions of highly relevant service goals of two queries "book hotel" and "get traffic information" in the goal recommendation lists generated using five similarity measures. SM1, SM2, SM3, SM4, and SM5 stand for JSim, CSim, SyWNSim, AsyWNSim, and AsyWNSim + DiffW, respectively. The gray colored entries indicate the highly relevant goals that are not contained in the top 50 recommendations. We can see that the ranking positions of highly relevant goals in the recommendation lists generated using JSim and CSim are much worse than those of the other three measures. The goal recommendation lists generated by AsyWNSim + DiffW are the best. All these results are consistent with the average MAP and NDCG values shown in Figs. 4 and 5. Some highly relevant goals that are not ranked highly in the recommendation lists generated using AsyWNSim + DiffW, e.g., <explore, book accommodation, null> of "book hotel." Based on our analysis, this is mainly because of the unexpected word similarities in WordNet, e.g., $WNSim(hotel, accommodation) = 0.0$.

6.4. Evaluation of service discovery

In this section, we report the evaluation results and analysis of service discovery, including two aspects: 1) comparison of five service discovery approaches and 2) comparison of three query expansion approaches.

(1) Comparison of five service discovery approaches

As discussed previously, logic-based semantics-aware service discovery approaches that can achieve good performance are difficult to be implemented due to the unavailability of suitable domain ontologies and considerable manual efforts required for annotating services and queries using semantic Web service description languages. We compared two service discovery approaches proposed in this work with three widely used approaches. The five approaches are described as follows.

1) *Keyword-based service discovery* (KWSD): Existing service registries usually adopt Keyword-based search technology. For each experimental query, we calculated the Cosine similarities between services and the query based on their word frequency vectors.

Table 9
Ranking positions of highly relevant service goals of "get traffic information" in the goal recommendation lists generated using five similarity measures.

	High relevant service goals	SM1	SM2	SM3	SM4	SM5
1	<get, alert traffic information, {from location}>	1	1	1	1	1
2	<deliver, information, {on border crossing, on ferry information, on highway alert, on traffic flow, on travel time}>	60	11	2	2	2
3	<provide, developer access, {to traffic data in United States}>	111	111	14	14	3
4	<connect, developer, {with NZ traffic monitoring system}>	84	88	96	94	6
5	<allow, retrieval, {of map-based traffic information for destination}>	14	5	30	30	7
6	<provide, traffic information, {for location in Europe}, {with algorithm for route optimization, with algorithm for route selection}>	48	66	71	67	8
7	<present, accurate relate information, {on traffic safety}, …>	17	20	15	15	11
8	<give, access, {to bicycling information, to current traffic rideshare}>	15	14	12	12	14
9	<give, real-time data, {about traffic condition on CHP incident report, about traffic condition on information, about traffic condition on region freeway}>	61	13	13	13	15
10	<provide, user, {to traffic information, to trip planning assistance, to weather}>	18	21	11	11	16
11	<support, retrieval, {of report from traffic summary}>	80	85	76	72	17
12	<expose, traffic data, null>	21	25	42	39	19
13	<expose, set, {of traffic data}, {for christchurch area}>	85	89	37	36	20
14	<provide, traffic information>	3	3	69	65	27
15	<provide, public sector information, {for real-time traffic data}>	16	15	70	66	28
16	<report, Vessel traffic>	39	43	75	71	29
17	<provide, real-time data, {on traffic condition in New Zealand, on traffic condition in Auckland}>	118	52	44	41	30
	<provide, customer, {with current disruption in SL traffic}>	90	94	227	216	37
18	<provide, customer, {with current status of SL traffic situation}>	108	109	232	221	38
19	<provide, traffic tracking, {for communication campaign, for social medium messaging}>	121	119	186	179	40
20	<provide, traffic feature>	38	42	219	208	42

2) *LDA-based service discovery* (LDASD): Many service discovery approaches have been recently proposed by leveraging topic models like PLSA and LDA. We implemented an approach based on LDA according to Naim et al. (2016), as detailed in Section 5.3. Note that the original keywords of the 21 experimental queries were used as input of this approach.
3) *Service discovery provided by PW* (PWSD): This is the service discovery approach used by PW. We submitted each experimental query to the PW service search engine and collected the returned API lists. To compare with the other approaches, we filtered the APIs in the lists that are not contained in the experimental dataset.
4) *Goal-based service discovery* (GSD): This is an approach proposed in this work that retrieves services by matching service goals of services with those contained in an expanded query, as demonstrated in Eqs. (23) and (24). As described in Section 6.2, the expanded result of each experimental query was obtained by collecting the relevant service goals in the top 50 of the goal recommendation list generated using AsyWNSim + DiffW under the setting of $\lambda_1 = 0.3$, $\lambda_2 = 0.6$, $\lambda_3 = 0.1$, $\theta = 0.9$, and $k = 15$.
5) *Hybrid service discovery based on goal-oriented query expansion* (HSD + GoQE): This is the proposed hybrid service discovery approach that integrates KWSD, LDASD, and GSD. The expanded result of each experimental query was used as input of KWSD and LDASD, as detailed in Section 5.3. The three component approaches were equally treated by setting $\gamma_1 = \gamma_2 = \gamma_3 = 1$.

Figs. 6 and 7 show the average MAP and NDCG of the API lists retrieved for the 21 queries using five service discovery approaches. HSD + GoQE achieves the best performance in most cases, while LDASD performs the worst. Through analysis, latent topics learned by LDA are insufficient to reflect the semantics of APIs accurately; and thus LDASD cannot retrieve appropriate APIs

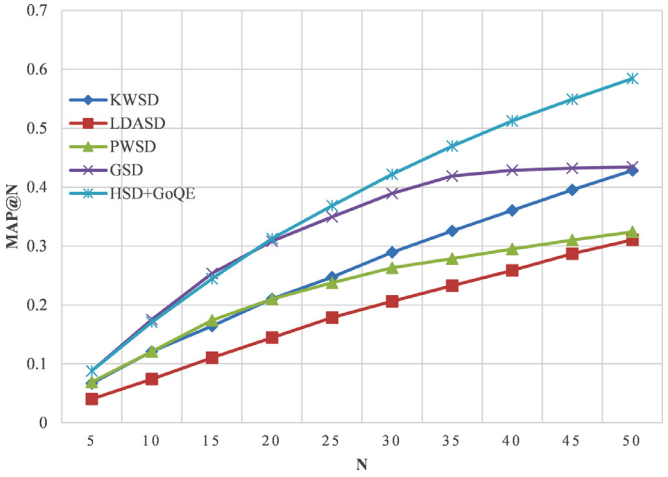


Fig. 6. Comparison of five service discovery approaches on MAP@N.

for queries. The performance values of HSD + GoQE and GSD are quite close when N is less than 15, indicating that GSD largely determines the top 15 APIs retrieved by HSD + GoQE. After that point, the MAP of GSD improves more and more slowly until stabilized while the NDCG of GSD degrades, as N increases. This is caused by the fact that the numbers of APIs retrieved by GSD are limited (no more than 35 for most of the 21 queries) due to the reasons explained in Section 5.3. Thanks to the other two component approaches, the MAP of HSD + GoQE keeps growing when N becomes larger than 15.

When N is less than 15, the performance values of PWSD and KWSD are very close. After that, PWSD becomes worse than KWSD,

Table 10
Comparison on MAP@N of KWSD and LDASD with or without three query expansion approaches.

Approaches	N									
	5	10	15	20	25	30	35	40	45	50
KWSD	0.0667	0.1209	0.1644	0.2104	0.2471	0.2893	0.3255	0.3606	0.3953	0.4283
KWSD + GoQE	0.0671	0.1308	0.1928	0.2472	0.3011	0.343	0.383	0.4232	0.4608	0.4969
KWSD + WNQE-1	0.0653	0.1142	0.1501	0.1877	0.2223	0.2582	0.2892	0.32	0.3457	0.3728
KWSD + WNQE-2	0.0597	0.1044	0.1356	0.1678	0.1985	0.2264	0.2504	0.2728	0.2964	0.3176
LDASD	0.0404	0.0742	0.1103	0.1444	0.1785	0.2063	0.2326	0.2587	0.2869	0.3103
LDASD + GoQE	0.042	0.0805	0.1159	0.1553	0.1829	0.2092	0.2375	0.2644	0.2918	0.3185
LDASD + WNQE-1	0.0407	0.0699	0.0955	0.1207	0.1464	0.1683	0.1897	0.2135	0.2334	0.2537
LDASD + WNQE-2	0.0298	0.0498	0.0703	0.089	0.1068	0.1217	0.1364	0.1497	0.1647	0.1779

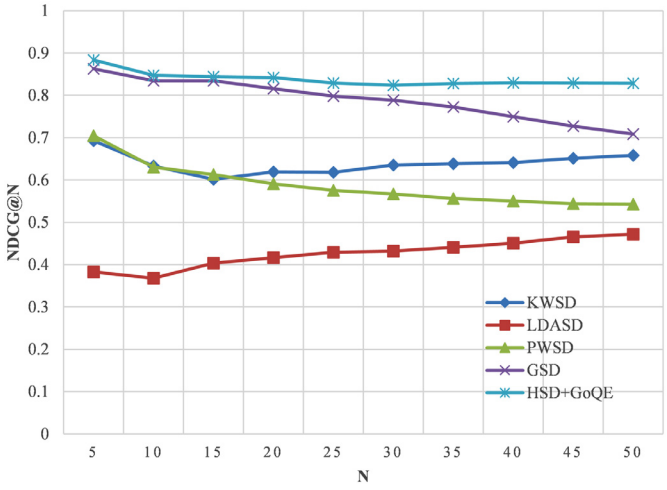


Fig. 7. Comparison of five service discovery approaches on NDCG@N.

and the performance gaps are getting increasingly more significant as *N* continues to increase. Moreover, PWSD has the similar trends as GSD in terms of both MAP and NDCG. Through observation, PW uses keyword-based matching, that is, PW will return an API only if it contains all keywords of a query in its descriptive data. As a result, many relevant APIs of queries are not retrieved by PW. For example, PW only returns one, two, and seven APIs for queries "get weather report," "find bus stop," and "find airport," respectively.

(2) **Comparison of three query expansion approaches**

In our proposed service discovery approach, a goal-oriented query expansion approach (referred to as GoQE) is used to help users refine their initial queries by leveraging service goal knowledge, which can promote the performance of service discovery. As stated previously, there are some existing query expansion approaches proposed for service discovery, which are mainly based on WordNet or domain ontologies. Since there is no available domain ontology for our experiments, we compared GoQE with two kinds of WordNet-based query expansion approaches, which are referred to as WNQE-1 and WNQE-2, respectively. WNQE-1 expands a query with the synonyms of its words in WordNet. WNQE-2 expands a query with synonyms, hypernyms, and hyponyms of its words in WordNet.

In the experiment of HSD + GoQE, we obtained two API lists for each experimental query *q* by performing KWSD and LDASD on the expanded result of *q* generated using GoQE. Here, we additionally constructed two expanded results of *q* using WNQE-1 and WNQE-2, respectively; and then performed KWSD and LDASD on each of the two expanded queries. For each API list retrieved by these different versions of KWSD and LDASD, we computed the MAP and NDCG of the top 50 of the list. Afterward, we computed the average MAP and NDCG of each approach on the 21 queries.

Tables 10 and 11 present the average MAP and NDCG of four versions of KWSD and LADSD. Except for the original KWSD and LDASD, the other three versions of KWSD and LDASD are combined with GoQE, WNQE-1, and WNQE-2, respectively. For example, KWSD + GoQE, KWSD + WNQE-1, and KWSD + WNQE-2 denote the KWSD combined with GoQE, WNQE-1, and WNQE-2, respectively. In most cases, the performance values of KWSD + GoQE and LDASD + GoQE are better than those of KWSD and LDASD. More specifically, the MAP and NDCG of KWSD are improved on average by 15.16% and 1.98%, respectively. The MAP and NDCG of LDASD are improved on average by 3.76% and 4.13%, respectively. The performance values of the other two versions of KWSD and LDASD (combined with WNQE-1 and WNQE-2) are worse than those of KWSD and LDASD. Based on our analysis, the degraded performance is caused by the fact that many semantically relevant words of queries extracted from WordNet are unexpected, e.g., the synonym *record* of *book* and some desired words are not included, e.g., the semantically equivalent word *accommodation* of *hotel*. Moreover, the performance values of KWSD + WNQE-2 and LDASD + WNQE-2 are much worse than those of KWSD + WNQE-1 and LDASD + WNQE-1. This is because more unexpected words are introduced by considering hypernyms and hyponyms. All these results demonstrate that our proposed GoQE can contribute to better performance of some state-of-the-art approaches, e.g., KWSD and LDASD, while WNQE-1 and WNQE-2 will decrease the performance.

7. Distributions, Limitations, and threats to validity

7.1. Contributions

The significant contributions of this work lie in three aspects. Firstly, we present an approach to mining knowledge on service functionalities from a service registry, which contains two main steps: service goal extraction and service goal clustering. The first step was studied in Zhang et al. (2017). In this paper, we propose a method for the second step. In particular, we design a method for measuring semantic similarities between service goals by employing the word similarities in WordNet and distinguishing the importance of three different types of words in a specific domain. Secondly, we propose a query expansion approach based on the mined service goal knowledge, which can help service requesters refine their initial queries. For a given query, service goals assigned to the similar service goal clusters are recommended. From the recommendations, the service requester can gain a better understanding of service functionalities related to his/her functional requirements, and select some desired goals as an expanded query. Thirdly, we propose a hybrid service discovery approach based on the expanded query. The approach contains three components: a goal-based approach that matches the service goals of each service with those in the expanded query and two existing approaches

Table 11
NDCG@N of KWSD and LDASD with or without three query expansion approaches.

	N									
Approaches	5	10	15	20	25	30	35	40	45	50
KWSD	0.6931	0.633	0.6018	0.6192	0.6182	0.6351	0.6387	0.6411	0.6513	0.6584
KWSD + GoQE	0.6406	0.621	0.6298	0.638	0.6446	0.6538	0.6617	0.6649	0.6732	0.683
KWSD + WNQE-1	0.659	0.5983	0.5779	0.5808	0.5846	0.5956	0.6007	0.6083	0.6083	0.6155
KWSD + WNQE-2	0.5663	0.532	0.5153	0.5195	0.5258	0.5307	0.5344	0.5412	0.5485	0.5602
LDASD	0.3828	0.3681	0.4036	0.4166	0.4295	0.4317	0.441	0.4508	0.4654	0.4721
LDASD + GoQE	0.4137	0.4133	0.4177	0.4355	0.4382	0.4464	0.4505	0.4586	0.4691	0.4855
LDASD + WNQE-1	0.3573	0.3425	0.3411	0.3513	0.3649	0.3754	0.3839	0.4061	0.4136	0.4276
LDASD + WNQE-2	0.2802	0.2756	0.2825	0.2897	0.2998	0.3054	0.3173	0.3257	0.3391	0.3466

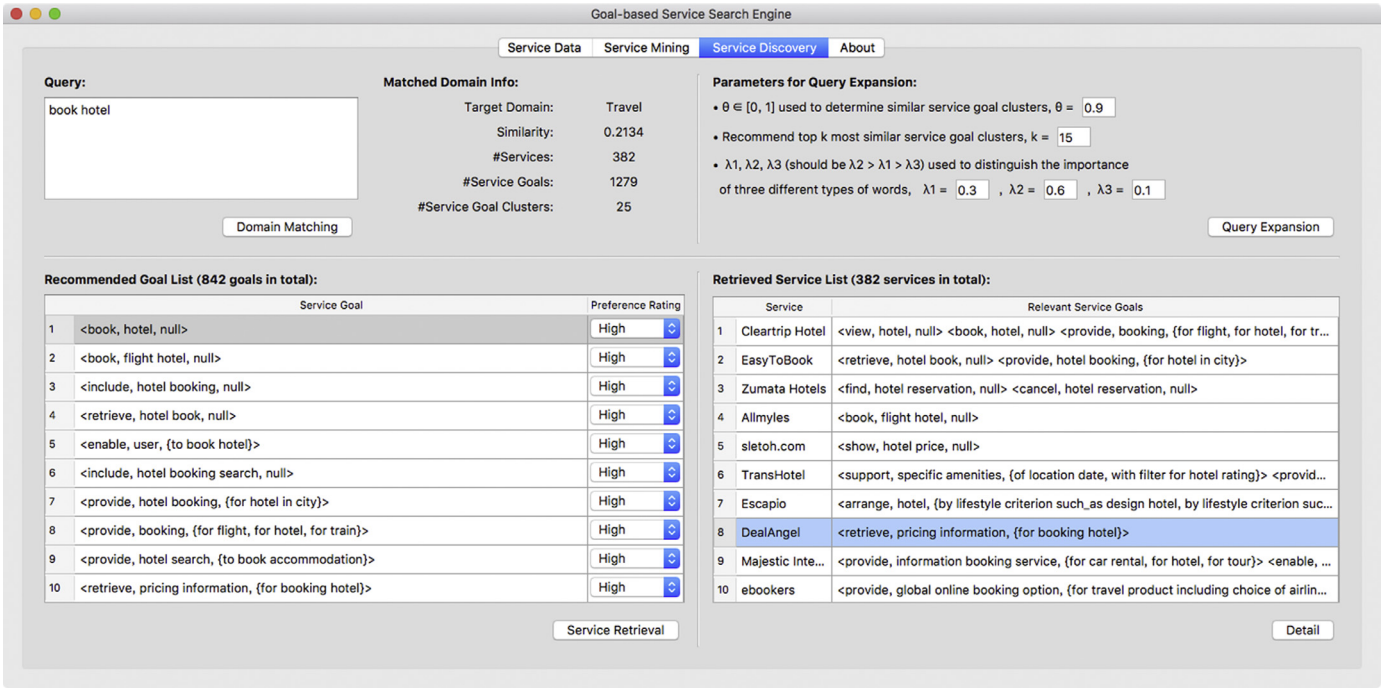


Fig. 8. Snapshot of our prototype system.

(i.e., a keyword-based and an LDA-based) improved by taking the expanded query as input.

Experiment results show that the proposed query expansion approach can effectively recommend semantically similar service goals for queries and the proposed service discovery approach outperforms several popular approaches. Another advantage of our proposed approach is that the retrieved services can be displayed along with their service goals relevant to the query, which can help the requester in getting a quick understanding of services' functionalities and choosing the services of interest quickly. We have developed a prototype system according to the proposed approach. Fig. 8 shows a snapshot of the recommended service goal list and the retrieved API list for query "book hotel" in our prototype system.

7.2. Limitations

There are some limitations of the proposed approach. As for the service goal clustering and goal-oriented query expansion approaches, the key is to measure the similarities between service goals and the similarities of recommended goals to a query. We implement these two tasks based on the semantic similarities of words in WordNet. However, not all word similarities in WordNet are suitable for the tasks. As a result, some similar service goals fail to be grouped, and some similar goals of a query cannot be ranked at high positions in the goal recommendation list (see Tables 8 and 9). Moreover, to guarantee the quality of constructed domain-specific verbs and core nouns, domain analysts are involved in the construction process based on two automatically generated lists, as described in Section 4.2. Although this semi-automated process in the first round of construction is time-consuming and error-prone, the participation of domain analysts can be minimized by adopting a suitable updating mechanism. For example, when a certain number of new services are added to the domain, a ranked keyword list can be automatically generated and then be compared with existing domain assets. The discrepancies could be quickly judged by domain analysts to update the domain assets. We will investigate this issue in the future.

As for the service discovery approach, we focus in this research on retrieving services that can satisfy functional requirements. In practice, there can be many services that provide the same (or semantically equivalent) functionalities, e.g., APIs *Cleartrip Hotel*, *EasyToBook*, and *DealAngel* retrieved for "book hotel" shown in Fig. 8. To facilitate the service selection for service requesters, these similar services can be re-ranked by considering their non-functional properties, e.g., cost, response time, and the frequency used by mashups. Our future work will investigate this problem.

7.3. Threats to validity

In this section, we discuss the critical threats to the internal and external validity of our evaluation.

The *internal validity* is concerned about the repeatability of the experiments. In the experiments, several resources are manually built by subjects, including a set of verbs and a set of core nouns for each domain, a set of queries, and the relevant service goals and the relevant APIs of each query. A threat to the internal validity is that the subjects' understanding of their assigned domains may have an impact on the resulting resources, which will, in turn, affect the experiment results. To mitigate this threat, we assigned each selected domain to two subjects and adopted the following two strategies. First, before conducting the experiments, we held a meeting for introducing to the subjects the proposed approach and the experiment tasks needed to be done by them. The subjects were then given enough time (i.e., one week) to familiarize themselves with the background knowledge of their assigned domains using Wikipedia pages, the ranked domain keyword lists, the service goals extracted for domain-specific services, and so on. Second, after the familiarization process, we held a meeting with the subjects again to build each type of the resources for each domain in a two-step way. Two subjects assigned to a domain built the resource independently and then discussed debatable parts together to reach an agreement.

The *external validity* is concerned about the generalizability of the experiment results. A concern with the external validity is the representativeness of APIs and queries used for experimentation. The APIs were crawled from one service registry, PW. Specifically, we selected seven domains according to their scale and popularity. Since the content of PW relies on the input from API providers and other users, this will introduce the danger of the selection of domains. Moreover, the scale and popularity of a domain will change over time as new APIs, and new mashups are continuously registered at PW. As for the queries, we asked the recruited 14 subjects to construct three representative queries for each domain, resulting in 21 experimental queries. Although the evaluations with this small number of queries show positive results, we plan to expand our evaluations by recruiting more subjects and using them to evaluate queries on multiple datasets besides PW.

8. Conclusions and future work

This paper proposes a service discovery approach by leveraging service goal knowledge mined from textual service descriptions. We firstly present a method for clustering the service goals extracted from textual service descriptions by measuring their semantic similarities. Afterward, a query expansion approach is proposed to help service requesters refine initial queries by recommending similar service goals. Finally, a hybrid service discovery approach is proposed to retrieve services according to the goals selected by the requester from recommendations. Experiments conducted on a real-world service dataset crawled from ProgrammableWeb show the effectiveness of the proposed approach.

The limitations discussed in Section 7.2 will serve as the directions of our future works. For example, we plan to improve the goal-oriented query expansion approach by utilizing service goals selected by service requesters for queries. Moreover, we will consider non-functional properties of services in addition to functional requirements to develop an integrated method of service discovery. Also, another potential usage of extracted service goals is that they can help developers in functional design of SBSs by identifying essential functionalities related to a given topic. For example, which functionalities are more popular, i.e., with higher frequencies; which functionalities have not been fully exploited in existing services, i.e., with lower frequencies or non-occurrence. In the future, we plan to further mine semantic associations among service goals, to provide more insightful knowledge for the functional design of SBSs.

Acknowledgment

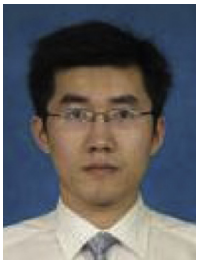
This research was supported by the National Basic Research Program of China (No. 2014CB340404), National Key Research and Development Program of China (No. 2017YFB1400602), National Science Foundation of China (Nos. 61572371, 61672387, and 61402150), and the Strategic Team-Building of Scientific and Technological Innovation in Hubei Province of China.

References

Aljoumaa, K., Assar, S., Souveyet, C., 2011. Reformulating user's queries for intentional services discovery using an ontology-based approach. In: Int'l Conf. New Technol., Mobility and Security, pp. 1–4.
Aznag, M., Quafafou, M., Jarir, Z., 2014. Leveraging formal concept analysis with topic correlation for service clustering and discovery. In: IEEE Int'l Conf. Web Serv, pp. 153–160.
Bano, M., Zowghi, D., Ikram, N., et al., 2014. What makes service oriented requirements engineering challenging? A qualitative study. IET Softw. 8 (4), 154–160.
Bird, S., Loper, E., Klein, E., 2009. Natural Language Processing With Python. O'Reilly Media Inc.
Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent dirichlet allocation. J. Mach. Learn. Research 3, 993–1022.
Cassar, G., Barnaghi, P., Moessner, K., 2014. Probabilistic matchmaking methods for automated service discovery. IEEE Trans. Serv. Comput. 7 (4), 654–666.
Chang, C., Lin, C.. LIBSVM: a library for support vector machines.
Chen, F., Li, M., Wu, H., et al., 2017a. Web service discovery among large service pools utilizing semantic similarity and clustering. Enterprise Info. Syst. 11 (3), 452–469.
Chen, F., Lu, C., Wu, H., et al., 2017b. A semantic similarity measure integrating multiple conceptual relationships for Web service discovery. Expert Syst. with Appl. 67, 19–31.
Chen, L., Wang, Y., Yu, Q., et al., 2013. WT-LDA: user tagging augmented LDA for web service clustering. In: Int'l Conf. Service-Oriented Comput, pp. 162–176.
Cheng, X., Lan, Y., Guo, J., et al., 2014. BTM: topic modeling over short texts. IEEE Trans. Knowl. and Data Eng. 26 (12), 2928–2941.
Cong, Z., Fernandez, A., Billhardt, H., et al., 2015. Service discovery acceleration with hierarchical clustering. Info. Syst. Frontiers 17 (4), 799–808.
Crasso, M., Zunino, A., Campo, M., et al., 2011. A survey of approaches to Web service discovery in service-oriented architectures. J. Database Manag. 22 (1), 102–132.
Dan, P., Moore, A.W., 2000. X-means: extending *K*-means with efficient estimation of the number of clusters. In: Int'l Conf. Machine Learning, pp. 727–734.
Dong, X., Halevy, A., Madhavan, J., et al., 2004. Similarity search for Web services. VLDB 372–383.
García J, M., Ruiz, D., Garc, A.R., et al., 2012. Improving semantic Web services discovery using SPARQL-based repository filtering. In: Web Seman. Sci. Serv. and Agents on the World Wide Web, 17, pp. 12–24.
Garg, A., Enright, C.G., Madden, M.G., 2015. On asymmetric similarity search. In: IEEE Int'l Conf. Mach. Learn. and Appl, pp. 649–654.
Ghali, B.E., Qadi, A.E., 2017. Context-aware query expansion method using language models and latent semantic analyses. Knowl. Info. Syst. 5 (3), 751–762.
He, Q., Yan, J., Jin, H., et al., 2014. Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction. IEEE Trans. Softw. Eng. 40 (2), 192–215.
He, Q., Zhou, R., Zhang, X., et al., 2016. Keyword search for building service-based systems. IEEE Trans. Softw. Eng. doi:10.1109/TSE.2016.2624293.
Hofmann, T., 1999. Probabilistic latent semantic analysis. In: 15th Conf. Uncertainty in Artificial Intell, pp. 289–296.
Khanmohammadi, S., Adibeig, N., Shanehbandy, S., 2017. An improved overlapping k-means clustering method for medical applications. Expert Syst. Appl. 67, 12–18.
Klusch, M., Fries, B., Sycara, K., 2009b. OWLS-MX: A hybrid semantic Web service matchmaker for OWL-S services. In: Web Seman. Sci. Serv. and Agents on the World Wide Web, 7, pp. 121–133.
Klusch, M., Kapahnke, P., Schulte, S., et al., 2016. Semantic Web service search: A brief survey. KI Künstliche Intell 30 (2), 139–147.
Klusch, M., Kapahnke, P., Zinnikus, I., 2009a. Hybrid adaptive Web service selection with SAWSDL-MX and WSDL-Analyzer. In: European Seman. Web Conf, pp. 550–564.
Klusch, M., Kaufer, F., 2009. WSMO-MX: a hybrid semantic Web service matchmaker. J. Web Intell. and Agent Syst. 7 (1), 23–42.
Kokash, N., Heuvel, W., D'Andrea, V., 2006. Leveraging Web services discovery with customizable hybrid matching. In: Int'l Conf. Service-Oriented Comput, pp. 522–528.
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.

Lu, W., Cai, Y., Che, X., et al., 2016. Joint semantic similarity assessment with raw corpus and structured ontology for semantic-oriented service discovery. Personal and Ubiquitous Comput. 20 (3), 311–323.
Ma, S.P., Li, C.H., Tsai, Y.Y., et al., 2013. Web service discovery using lexical and semantic query expansion. In: IEEE Int'l Conf. E-Business Eng, pp. 423–428.
Manning, C.D., Raghavan, P., Schutze, H., 2009. An Introduction to Information Retrieval. Cambridge University Press.
Marcus, M.P., Marcinkiewicz, M.A., Santorini, B., 1993. Building a Large Annotated Corpus of English: the Penn Treebank. MIT Press.
Marneffe, M., MacCartney, B., Manning, C.D., 2006. Generating typed dependency parses from phrase structure parses. LREC.
Miller, G.A., 1995. WordNet: a lexical database for english. Commun. ACM 38 (11), 39–41.
Naim, H., Aznag, M., Quafafou, M., et al., 2016. Probabilistic approach for diversifying web services discovery and composition. IEEE Int'l Conf. Web Serv. 73–80.
Paliwal, A.V., Shafiq, B., Vaidya, J., et al., 2012. Semantics-based automated service discovery. IEEE Trans. Serv. Comput. 5 (2), 260–275.
Plebani, P., Pernici, B., 2009. URBE: Web service retrieval based on similarity evaluation. IEEE Trans. Knowl. Data Eng. 21 (11), 1629–1642.
Porter, M.F., 2006. An algorithm for suffix stripping. Readings Info. Retriev. 130–137.
Rodriguez Mier, P., Pedrinaci, C., Lama, M., et al., 2016. An integrated semantic Web service discovery and composition framework. IEEE Trans. Serv. Comput. 9 (4), 537–550.
Rolland, C., Souveyet, C., Achour, C.B., 1998. Guiding goal modeling using scenarios. IEEE Trans. Softw. Eng. 24 (12), 1055–1071.
Roman, D., Kopecký, J., Vitvar, T., et al., 2015. WSMO-Lite and hRESTS: Lightweight semantic annotations for Web services and RESTful APIs. J. Web Seman. 31, 39–58.
Stevenson, M., Greenwood, M.A., 2006. Comparing information extraction pattern models. Assoc. Comput. Lingu. 12–19.
Wang, J., Gao, P., Ma, Y., et al., 2017a. A web service discovery approach based on common topic groups extraction. IEEE Access. doi:10.1109/ACCESS.2017.2712744.
Wang, J., Zhang, N., Zeng, C., et al., 2013. Towards services discovery based on service goal extraction and recommendation. In: IEEE Int'l Conf. Serv. Comput, pp. 65–72.
Wang, Y., Lin, X., Wu, L., et al., 2017b. Effective multi-query expansions: collaborative deep networks for robust landmark retrieval. IEEE Trans. Image Process. 26 (3), 1393–1404.
Wang, Y., Stroulia, E., 2003. Flexible interface matching for Web service discovery. In: IEEE Int'l Conf. Web Info. Syst. Eng, pp. 147–156.
Wei, D., Wang, T., Wang, J., et al., 2011. SAWSDL-iMatcher: a customizable and effective semantic Web service matchmaker. In: Web Seman. Sci. Serv. and Agents on the World Wide Web, 9, pp. 402–417.
Xia, B., Fan, Y., Tan, W., et al., 2015. Category-aware API clustering and distributed recommendation for automatic mashup creation. IEEE Trans. Serv. Comput. 8 (5), 674–687.
Zhang, J., Wang, J., Hung, P., et al., 2012. Leveraging incrementally enriched domain knowledge to enhance service categorization. Int'l J. Web Serv. Research 9 (3), 43–66.
Zhang, N., Wang, J., Ma, Y., 2017. Mining domain knowledge on service goals from textual service descriptions. IEEE Trans. Serv. Comput. doi:10.1109/TSC.2017.2693147.

Neng Zhang is a Ph.D. student of the School of Computer Science, 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 School of Computer Science, 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.

Yutao Ma is an associate professor of the School of Computer Science, Wuhan University, China. He received the Ph.D. degree in 2007 from Wuhan University, China. His current research interests focus on software engineering and services computing.

Keqing He is a professor of the School of Computer Science, Wuhan University, China. He received his Ph.D. 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.

Xiaoqing (Frank) Liu is currently a professor of the Computer Science and Computer Engineering Department, University of Arkansas. He received his Ph.D. degree in computer science from the Texas A&M University in 1995. His current interests include software engineering, service computing, online argumentation and collaborative systems, fuzzy logic, knowledge-based systems, and artificial intelligence applications.