Mining Domain Knowledge on Service Goals from Textual Service Descriptions

Neng Zhang, Jian Wang, and Yutao Ma, *Member, IEEE*

Abstract—With the rapid development of service-oriented computing, a large number of software applications have been developed based on the services computing framework. It is well known that software engineering is a knowledge-intensive activity, and thus the effective management of service-related knowledge facilitates service-oriented software development. Although many methodologies have been proposed for service-oriented knowledge management, little attention has been paid to mining knowledge (especially domain-specific functionalities) from service resources. To address this issue, we propose an approach to mine domain knowledge on service goals (i.e., service functionalities) from textual descriptions of services. The approach consists of two components: service goal extraction from textual service descriptions based on linguistic analysis and domain service goal construction that merges semantically similar service goals within a domain. The effectiveness of the proposed approach is validated by a series of experiments conducted on a real-world dataset crawled from the ProgrammableWeb.

Index Terms—Service-oriented software development, domain service goal, knowledge mining

1 Introduction

With the rapid development of service-oriented computing (SOC), a large number of software applications are developed by reusing the operations offered by existing services which can be discovered and orchestrated to deliver the desired functionality [1]. This service-oriented software development (SOSD) paradigm can lower the required cost, time, and effort of development and increase reusability, agility, quality, and customer satisfaction [2]. It has been widely applied in various domains, e.g., workflow management, finance, e-Business, and e-Science [4].

In spite of the significant improvements in technologies and tools, it was recognized in [3] that SOSD still faces various challenges. A qualitative study reported in [2] reveals that the integration of knowledge management strategies into the SOSD lifecycle is one of the most significant challenges. As stated in [2], integrating knowledge management techniques and tools into the SOSD lifecycle can help requirements analysts in many activities of SOSD, such as the alignment of requirements and services, functionality design of services, and improving the reuse of services.

Let us consider a scenario in which a competitive service about a specific topic, e.g., *music*, is to be developed. During the requirements analysis stage, a requirements analyst has to decide which functionalities should be offered by the service. This is not an easy task, particularly when the functionalities are not explicitly specified by users, and meanwhile the analyst does not have a comprehensive understanding of the functionalities related to the target topic. For example, the functionalities related to *music* are often various, such as *search music*, *share music*, *provide music download*, and *create personal music profile*. This kind of knowledge can help the analyst understand the scope of functionalities related to the target topic and recommend missing or potentially interesting functionalities to the analyst. Moreover, the popularity of functionalities in existing service registries may contribute to the functionality selection of a service. According to the popularity of functionalities, the analyst can be informed about which functionalities (e.g., *search music*) occur more frequently than the others, and about which functionalities (e.g., *create personal music profile*) are less concerned by existing services.

However, it is difficult to implement effective domain knowledge management because most service-oriented knowledge is not expressed in a structured or formalized form [2]. Hence, great efforts are required to elicit structured knowledge from descriptive texts, software practitioners, domain experts, etc. To the best of our knowledge, although many modeling methodologies [6], [7], [8], [9], [11], [12] have been proposed for service-oriented knowledge management, few studies have been dedicated to mining knowledge, especially domain-specific functionalities, from service resources on the Internet.

To address this problem, we propose an approach to mine domain knowledge on service goals from textual descriptions of services. The two components of the approach can be summarized as follows:

- *Service goal extraction*: Service goals (i.e., service functionalities) such as *share online music on Twitter* and *share music playlist* are extracted from textual service descriptions using natural language processing (NLP) techniques.
- *Domain service goal construction*: Domain service goals that represent the commonality and variability of

• *The authors are with the State Key Laboratory of Software Engineering, School of Computer, Wuhan University, Hubei 430072, China. E-mail: {nengzhang, jianwang, ytma}@whu.edu.cn.*

Manuscript received 13 June 2016; revised 24 Mar. 2017; accepted 6 Apr. 2017. Date of publication 12 Apr. 2017; date of current version 12 June 2020. Digital Object Identifier no. 10.1109/TSC.2017.2693147

domain-specific service functionalities are constructed by merging semantically similar service goals within a domain. For example, a domain service goal, i.e., *share* {*online*} *music* {*playlist*} {*on Twitter*}, can be constructed from the two service goals mentioned above.

We performed the proposed approach on a real-world dataset crawled from a popular service registry, ProgrammableWeb[1] (PW), and recruited 15 subjects for evaluation. According to the evaluation results, the service goal extraction approach achieved encouraging precision and recall; and the constructed domain service goals can facilitate several activities of SOSD, such as understanding domain functionalities and defining service requests for service search.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 gives an overview of the proposed approach for mining domain knowledge on service goals. In Section 4, we introduce the service goal extraction from textual service descriptions. The construction of domain service goals is detailed in Section 5. In Section 6, we summarize the contributions, limitations, and threats to validity. Section 7 concludes the paper.

2 Related Work

2.1 Service-Oriented Requirements Engineering

Service-oriented requirements engineering (SORE) is concerned with defining processes and methodologies to capture, elicit, and specify service requirements from two perspectives: service consumers and service providers [14]. Bano et al. [2] presented a qualitative study of the challenges to SORE highlighted in [3] by conducting interviews with practitioners working on service-oriented projects. The study reveals that the integration of knowledge management strategies into the SOSD lifecycle is one of the top two challenges.

Many modeling methodologies have been proposed for service-oriented knowledge management. Souza et al. [14] listed two methodologies that have been widely investigated: the RGPS (Role, Goal, Process, and Service) framework [8], [9] and the i* framework [11], [12]. The RGPS framework is a modeling approach for complex system requirements within the context of SOC [8], which describes personalized requirements involving domain-related services, towards the ultimate goal of enabling on-demand service provisioning. The i* framework provides a goal and agent oriented approach for modeling the environment of organizations and their information systems. It has been used to develop service-based systems by exploring business strategic goals together with a value-based framework [12].

In addition, Lee et al. [7] proposed a goal-driven approach to model the service request intention. Zhao et al. [6] built a semantic model to describe both business requirements and service capabilities based on environment ontologies. Although many modeling methodologies have been proposed for service-oriented knowledge management, little attention has been paid to mining knowledge related to domain functionalities from service resources. In this paper, we present an approach to extract domain knowledge on service functionalities in terms of goals, which can support goal-oriented methodologies, e.g., the RGPS and i* frameworks.

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

2.2 Ontology Learning from Service Descriptions

Ontology provides a way to structure data for comprehensive and transportable machine understanding [15]. Many approaches and tools [16], [17] have been developed for learning ontologies from various textual resources.

There are some studies that investigated ontology learning from service descriptions. For example, Segev et al. [18] proposed an ontology bootstrapping process by exploiting both WSDL (Web Service Description Language) and textual descriptions of services. Sabou et al. [19] developed a framework for ontology learning from textual sources attached to services. The framework implemented two learning methods that use two different kinds of linguistic knowledge: part-of-speech (POS) tagging information and word dependencies. Lee et al. [20] presented an approach to generate ontologies from WADL (Web Application Description Language) by grouping parameter names of services into semantic concepts and capturing the relationships between words in the parameter names. Compared with these studies, in this work, we focus on mining domain-specific functionalities (i.e., *<Verb-Noun-Prepositional Phrases>*) from textual service descriptions.

2.3 Functionality Mining from Text

Functionality mining from text has been widely investigated. For example, Robeer et al. [28] implemented a tool to construct conceptual models from a collection of agile requirements. Entities and relationships were extracted by orchestrating a selection of NLP heuristics. Ghosh et al. [29] proposed an approach to extract requirements specifications from natural languages and create formal models by connecting the specifications with precise formal representations. Casagrande et al. [30] developed an approach to support the extraction and modeling of goals from textual documents. In [33], a rule-based approach was proposed to extract goal and use case models from software requirements specifications. In contrast to these works, we propose an approach to extract functional goals from textual service descriptions instead of requirement documents. The extraction approaches and patterns are different.

Similar to our work, Jung et al. [21] also presented an approach to extract functional goals from textual service descriptions by employing 20 manually crafted word dependency patterns such as *nsubj+dobj* and *nsubj+xcomp*. Our work differs from theirs in two main aspects. First, through the analysis of various sentences, we determine the usage of different word dependencies for service goal extraction: some of them are used for extracting the skeletons of service goals, while others are used for obtaining additional information regarding the skeletons or discovering potential service goals. Based on these observations, we design a three-stage approach for service goal extraction. Second, we further construct domain service goals that can represent the commonality and variability of functionalities within a domain.

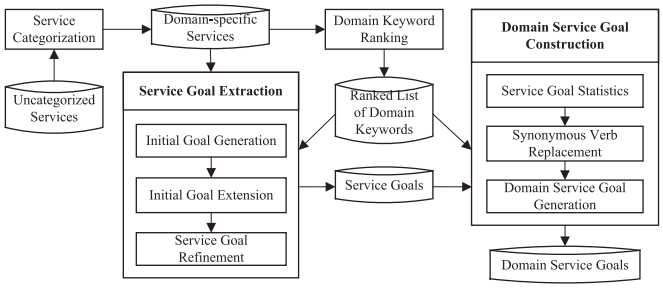


Fig. 1. Overview of the proposed approach.

3 Approach Overview

Fig. 1 illustrates the proposed approach for mining domain knowledge on service goals. Currently, there are two mainstream types of service descriptions: XML (eXtensible Markup Language)-based description (e.g., WSDL and WADL) and textual description. In this work, we focus on mining domain knowledge from the textual description because it is much more widely used to describe services, especially the increasingly popular RESTful services (or Web APIs) [5].

The proposed approach consists of two major components: service goal extraction and domain service goal construction. In the first component, given a set of domain-specific services, service goals are extracted from the textual descriptions of services using an NLP-based method. If the original services are uncategorized, they need to be first classified using classification approaches [13]. In the second component, domain service goals that represent the commonality and variability of domain-specific service functionalities are constructed by merging similar service goals in the domain. Note that a ranked list of domain keywords will be used in the two components. It can be produced using various techniques such as term frequency - inverse document frequency (TF-IDF).

A service goal is used to represent the functionality of a service. In [31], according to the existing studies on goal modeling [7], [23], a service goal *sg is represented by the triple <sgv, sgn, sgp> where sgv is a verb* (*phrase*) *that describes an action to be performed by sg, sgn is a noun (phrase) that describes an entity affected by the action, and sgp is an optional set of parameters* (*typically in the form of prepositional phrases*) *that describe additional information such as how the action affects the entity and the initial or final state of the entity*.

For clarity, *sgp* is divided into two parts: 1) *vsgp*, a set of parameters related to *sgv*, and 2) *nsgp*, a set of parameters related to *sgn*, such that *sg* is rewritten as <*sgv*, *sgn*, *vsgp*, *nsgp*>, e.g., <*retrieve*, *discount information*, {*in New York*}, {*of hotels*}>.

4 Service Goal Extraction

The textual description of a service generally consists of several sentences. The following sentences are excerpted from two Web APIs registered in the PW.

- **ESen-1** : It lets you upload data, share and mark up your data with collaborators, merge data from multiple tables, and create visualizations such as charts and maps (from the *Google Fusion Tables* API).
- **ESen-2** : It generates listings of flight options and last-minute airfare deals, with related information on package deals, hotel prices, etc. (from the *Tgels* API).

To extract service goals from textual service descriptions, it is necessary to obtain the grammatical structure of sentences. According to [19], the grammatical structure of a sentence *sen* is represented by two types of linguistic knowledge: 1) the POS tags of words, referred to as *POS*(*sen*), and 2) the word dependencies, referred to as *D*(*sen*). The results presented in [24] suggest that the Stanford Parser[2] can generate accurate analysis for most sentences. Fig. 2a depicts *POS*(*ESen*-1) and *D*(*ESen*-1) generated using the Stanford Parser. The meanings of the Stanford typed dependencies such as *dobj* and *conj* are detailed in [10]. More specifically, dependency *reln*(*a*, *b*) describes a binary relation named *reln* between two words *a* and *b* (*a* is called the governor and *b* is called the dependent). For example, the dependency *dobj*(*upload*-4, *data*-5) in Fig. 2a means that *data* is the accusative object of *upload*, where *upload* is the governor and *data* is the dependent, and 4 and 5 are the position indexes of *upload* and *data*, respectively, in *ESen*-1. The definitions of POS tags such as VB, VBP, and NNS can be referenced in [32].

In [31], we used the Stanford Parser to analyze the sentences extracted from textual service descriptions, and proposed an approach to extract service goals from the linguistic knowledge of sentences. In this paper, we improve the approach based on an in-depth analysis of sentence structures. The three stages of the improved approach are described in the following subsections.

4.1 Initial Goal Generation

Because of the diversity of sentence structures, extracting service goals from textual service descriptions is not an easy task. According to our analysis, a sentence may contain several service goals, and the skeletons of some service goals (referred to as *initial goals*) can be extracted directly from the dependencies. For example, the initial goal <*upload*, *data*, *null*, *null*> of *ESen*-1 shown in Fig. 2b is generated directly from the dependency *dobj*(*upload*-4, *data*-5). Based on the initial goals, the expected service goals of a sentence (including some potential service goals, e.g., <*share*, *data*, {*with collaborators*}, *null*> in *ESen*-1) can be obtained by utilizing more dependencies.

Table 1 presents three dependencies (i.e., *nsubjpass*(*a*, *b*), *dobj*(*a*, *b*), and *prep*(*a*, *b*)) that are used to generate initial goals. Note that *prep*(*a*, *b*) is used to cope with some special verb phrases such as *search for* and *deal with*. There are some exceptions, e.g., *prep_in*(*information*-7, *York*-10) and *prep_with* (*retrieved*-13, *API*-3) of the first example sentence in Table 1. To generate correct initial goals from *prep*(*a*, *b*), it should be ensured that: 1) *a* is a verb, 2) *b* is a noun, and 3) *a* does not appear in any *nsubjpass*(*a*, *b*) or *dobj*(*a*, *b*) of the same sentence. Restrictions 1) and 2) can be verified by the POS tags of *a* and *b*, i.e., *POS*(*a*) ∈ {VB, VBD, VBG, VBN, VBP, VBZ} and *POS*(*b*) ∈ {NN, NNS, NNP, NNPS}. As for restriction 3), *nsubjpass*(*a*, *b*) and *dobj*(*a*, *b*) should be processed before *prep*(*a*, *b*) so that any *prep*(*a*, *b*) with *a* appearing in *nsubjpass*(*a*, *b*) or *dobj*(*a*, *b*) can be filtered out. Moreover, there can be several *prep*(*a*, *b*) dependencies related to verb *a*, and only the first *prep*(*a*, *b*) dependency behind *a* can be used for initial goal generation. For example, only one initial goal <*search*, *information*, *null*, *null*> can be generated from *prep_for*(*search*-7, *information*-10) of the third example sentence in Table 1.

2. http://nlp.stanford.edu/software/lex-parser.shtml

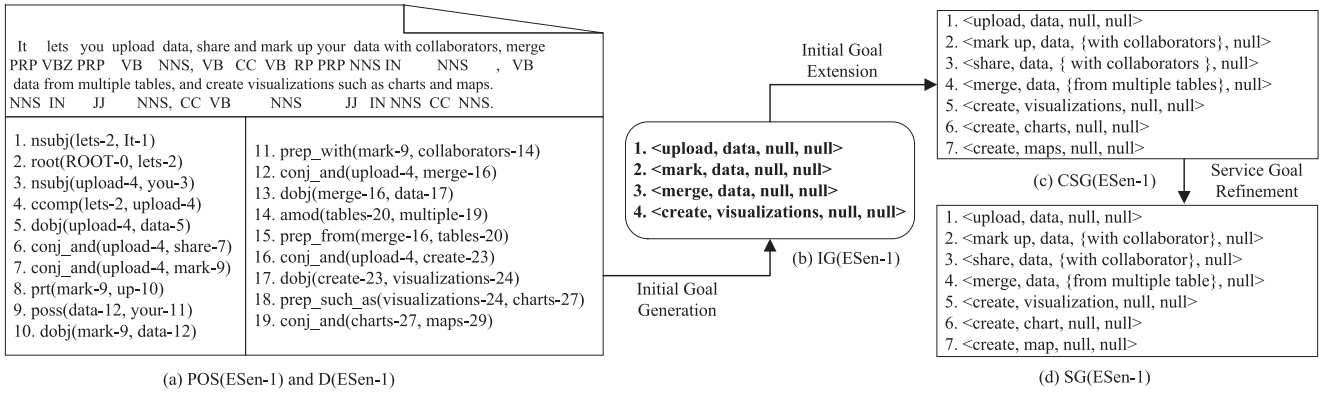


Fig. 2. Service goal extraction for *ESen*-1.

TABLE 1
Dependencies for Initial Goal Generation

Dependency	Definition	Usage	Example Sentence	Dependencies in the Example Sentence
nsubjpass(*a*, *b*)	noun *b* or head noun *b* of a noun phrase is the syntactic subject of a passive clause containing verb *a*	genetate an initial goal <*a*, *b*, *null*, *null*>	With this API, the hotel information in New York can be retrieved.	**nsubjpass(retrieved-13, information-7)** prep_with(retrieved-13, API-3) prep_in(information-7, York-10)
dobj(*a*, *b*)	noun *b* or head noun *b* of a noun phrase is the (accusative) object of verb *a*	generate an initial goal <*a*, *b*, *null*, *null*>	With this API, travelers can retrieve hotel information in New York.	**dobj(retrieve-7, information-9)** prep_with(retrieve-7, API-3) prep_in(retrieve-7, York-12)
prep(*a*, *b*) where *a* is a verb and *b* is a noun	verb *a* and its (accusative) object *b* are connected by a preposition	generate an initial goal <*a*, *b*, *null*, *null*>	With this API, travelers can search for hotel information in New York.	**prep_for(search-7, information-10)** prep_with(search-7, API-3)

According to Table 1, the initial goals of sentence *sen*, referred to as *IG*(*sen*), are obtained. For example, *IG*(*ESen*-1) is presented in Fig. 2b.

4.2 Initial Goal Extension

As can be seen from Fig. 2b, the initial goals of a sentence are usually not the expected service goals. On the one hand, the *sgv* and *sgn* parts of some initial goals may be incomplete, and the *vsgp* and *nsgp* parts of some initial goals also need to be supplemented. On the other hand, some potential service goals need to be further discovered. To solve these issues, the initial goals are extended using more dependencies, as shown in Table 2. Several important heuristics used for initial goal extension are presented here.

Heuristic 1. The *sgn* part of an initial goal may be the head noun of a noun phrase. The *amod* and *nn* dependencies are used to obtain the prefix modifier of *sgn*, e.g., for the phrase *newest google apps*, the prefix modifier of *apps* is *newest google*.

Heuristic 2. Several *conj* dependencies may be related to the *sgv* (or *sgn*) part of an initial goal, from which the coordinate verbs (or coordinate nouns) of *sgv* (or *sgn*) can be obtained. Moreover, the appositional modifiers of *sgn*, represented as *appos* dependencies, can also be viewed as the coordinate nouns of *sgn*, e.g., *get geographic information service* (*GIS*) in Table 2. The coordinate verbs and coordinate nouns will be used to extract potential service goals. However, some of them may be incorrect. For example, several coordinate verb pairs, e.g., (*upload*, *share*) and (*upload*, *mark*), can be explicitly identified from the *conj* dependencies in *D*(*ESen*-1), but only one implicit pair (*mark*, *share*) is expected according to the structure of *ESen*-1.

Through the analysis of coordinate verb structures, we devise a strategy to obtain the coordinate verbs of *sgv* for each $ig \in IG(sen)$. First, the initial goals in *IG*(*sen*) are sorted in ascending order by the position indexes of their *sgv* parts in *sen*. Afterwards, for each $ig \in IG(sen)$, all explicit and implicit coordinate verbs of *ig.sgv* are obtained using Algorithm 1. The function getPartner(*d*, *) obtains the partner of * in dependency *d*. If * is the governor, getPartner(*d*, *) returns the dependent; otherwise, it returns the governor. Steps 2-5 are used to obtain the explicit coordinate verbs of *ig.sgv* from the *conj* dependencies containing it. The coordinate verbs are stored as the keys of map *M* with initial value 0, where 0 indicates that the implicit coordinate verbs of *ig.sgv* related to the explicit coordinate verbs have not been identified. In the following steps 6-13, implicit coordinate verbs are obtained from the *conj* dependencies related to each coordinate verb in *M*. We also attempt to obtain the coordinate verbs of implicit coordinate verbs by putting them into *M* with initial value 0. At this point, any incorrect coordinate verb *v* is filtered out by the following rules:

1. *v* should not be the *sgv* part of any other initial goal in *IG*(*sen*);
2. *v* should not appear after *ig.sgn* in *sen*;
3. *v* should not be any coordinate verb retained for the *sgv* parts of initial goals that are handled before *ig*.

TABLE 2
Dependencies for Initial Goal Extension

Dependency	Definition	Usage	Example Phrase	Dependencies in the Example Phrase
prt(*a*, *b*)	*b* is the particle of verb *a* in a phrasal verb	identify the particle of *sgv*	mark up data	prt(mark-1, up-2)
prep(*a*, *b*) where *a* is a verb and *b* is a noun	noun *b* or head noun *b* of a noun phrase is related to verb *a* via a preposition	identify the prepositional phrase(s) related to *sgv*	mark up your data with collaborators	prep_with(mark-1, collaborators-6)
prep(*a*, *b*) where *a* and *b* are both nouns	noun *b* or head noun *b* of a noun phrase is related to noun *a* via a preposition	identify the prepositional phrase(s) related to *sgn*	get information of hotels	prep_of(information-2, hotels-4)
amod(*a*, *b*)	*b* is an adjective modifier of noun *a* or head noun *a* of a noun phrase	identify the adjectival modifier(s) of *sgn*	get newest google apps	amod(apps-4, newest-2)
nn(*a*, *b*)	*b* is a noun compound modifier of head noun *a* of a noun phrase	identify the noun compound modifier(s) of *sgn*		nn(apps-4, google-3)
conj(*a*, *b*)	*a* and *b* are two coordinate elements (e.g. verbs or nouns) connected by a coordinating conjunction, e.g. *and* and *or*	identify the coordinate verb(s) of *sgv* or coordinate noun(s) of *sgn*	1) get hotel and weather information 2) share and mark up data	1) conj_and(hotel-2, weather-4); 2) conj_and(share-1, mark-3)
appos(*a*, *b*)	noun *b* (or a noun phrase with head noun *b*) is an appositional modifier of noun *a* (or a noun phrase with head noun *a*)	identify the appositional modifier(s) of *sgn*, which can be viewed as coordinate nouns of *sgn*	get geographic information service (GIS)	appos(service-4, GIS-6)

Algorithm 1. Coordinate Verb Finder

Input: dependencies $D(sen)$ of sentence *sen*, and initial goal $ig \in IG(sen)$
Output: coordinate verbs of $ig.sgv$

1. $M \leftarrow \emptyset$; // a map for storing coordinate verbs of $ig.sgv$
2. **For each** $d \in D(sen)$
3. **If** d is a *conj* dependency **&&** d contains $ig.sgv$ **then**
4. $v \leftarrow$ getPartner(d, $ig.sgv$);
5. M.put(v, 0);
6. Try to select key v from M with value 0. If such v cannot be found, then go to step 14.
7. M.put(v, 1);
8. **For each** $d \in D(sen)$
9. **If** d is a *conj* dependency **&&** d contains v **then**
10. $v' \leftarrow$ getPartner(d, v);
11. **If** M.containsKey(v') is **false then**
12. M.put(v', 0);
13. Go to step 6.
14. **Return** the keys of M;

Fig. 3 shows the coordinate verbs of *merge*, i.e., the *sgv* part of $<merge, data, null, null> \in IG(ESen\text{-}1)$, obtained using Algorithm 1. The coordinate verbs *upload*, *mark*, and *create* will be filtered out by rule 1, and *share* will be filtered out by rule 3 because it has been retained for *mark*, i.e., the *sgv* part of $<mark, data, null, null>$.

Heuristic 3. Several *prep* dependencies may be related to the *sgv* (or *sgn*) part of an initial goal, which can be used to obtain the prepositional phrases related to *sgv* (or *sgn*). Because multiple prepositional phrases can be chained together, e.g., *with related information on package deals*, noun *b* identified from *prep*(*a*, *b*) needs to be further extended by *prep*.

Heuristic 4. The verbs identified from *conj* dependencies and the nouns identified from *conj*, *prep*, and *appos* dependencies need to be further extended. For example, for the phrase *North America and South Africa*, *Africa* identified from *conj_and*(*America*-2, *Africa*-5) needs to be extended by *nn*(*Africa*-5, *South*-4). Considering the fact that most prepositional phrases are not related to coordinate verbs, we extend only the identified verbs using *prt* and the identified nouns using *amod*, *nn*, and *prep*.

According to the dependencies depicted in Table 2 and the heuristics described above, we introduce a structure to store the extension result of an initial goal, which is defined as follows.

Definition 1. *The extension result of initial goal* $ig = < sgv, sgn, null, null >$ *is represented as* $Ext(ig) = < ig, vExt(ig), nExt(ig) >$. *Further,* $vExt(ig) = < prt, vCONJ, vPREP >$ *is the extension result of ig.sgv, where*

(a) Explicit coordinate verbs obtained from the *conj* dependencies related to *merge*

(b) Implicit coordinate verbs identified from the *conj* dependencies related to *upload*

Fig. 3. Coordinate verbs of *merge* in *ESen*-1.

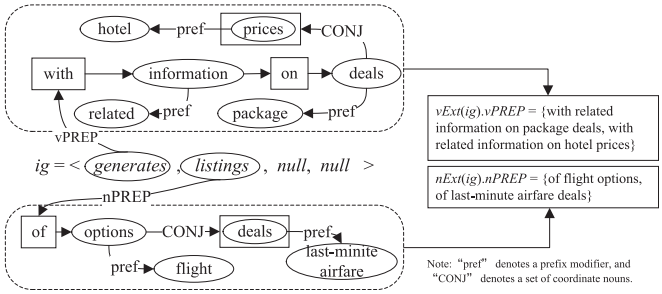


Fig. 4. Extension result of an initial goal.

- *prt is the particle of ig.sgv;*
- *vCONJ is the set of coordinate verbs of ig.sgv. Each cv ∈ vCONJ is represented as cv = <v, prt>, where v is a coordinate verb of ig.sgv and prt is the particle of v;*
- *vPREP is the set of prepositional phrases related to ig.sgv.*

And $nExt(ig) = < pref, nCONJ, nPREP >$ *is the extension result of ig.sgn, where*

- *pref is the prefix modifier of ig.sgn;*
- *nCONJ is the set of coordinate nouns of ig.sgn. Each cn ∈ nCONJ is represented as cn = <n, pref, PREP>, where n is a coordinate noun of ig.sgn, pref is the prefix modifier of n, and PREP is the set of prepositional phrases related to n;*
- *nPREP is the set of prepositional phrases related to ig.sgn.*

The coordinate nouns in *vExt(ig).vPREP*, *nExt(ig).nPREP*, and *cn.PREP* (*cn* ∈ *nExt(ig).nCONJ*) are expanded. Fig. 4 illustrates the extension result of <*generates*, *listings*, *null*, *null*> ∈ *IG*(*ESen*-2).

Based on the extension results of initial goals, the candidate service goals of a sentence are obtained. More specifically, given the extension result of initial goal *ig*, (|*vExt*(*ig*).*vCONJ*|+1) × (|*nExt*(*ig*).*nCONJ*|+1) candidate service goals, denoted by *CSG*(*ig*), are obtained. For each $csg_i \in CSG(ig)$,

- $csg_i.sgv$ can be the *sgv* part of initial goal *ig* or any coordinate verb *cv* ∈ *vExt*(*ig*).*vCONJ*, i.e.,

$$csg_i.sgv \in \left\{ (ig.sgv \circ vExt(ig).prt) \cup \left(\bigcup_{cv \in vExt(ig).vCONJ} cv.v \circ cv.prt \right) \right\}, \tag{1}$$

  where ∘ is the string concatenation operator.
- $csg_i.sgn$ can be the *sgn* part of initial goal *ig* or any coordinate noun *cn* ∈ *nExt*(*ig*).*nCONJ*, i.e.,

$$csg_i.sgn \in \left\{ nExt(ig).pref \circ ig.sgn \cup \bigcup_{cn \in nExt(ig).nCONJ} \mathrm{N}(cn) \right\}, \tag{2}$$

$$\mathrm{N}(cn) = \begin{cases} cn.pref \circ cn.n, & \text{if } cn.pref \neq null, \\ nExt(ig).pref \circ cn.n, & \text{otherwise.} \end{cases} \tag{3}$$

- Because most prepositional phrases are not related to coordinate verbs, the *vsgp* parts of all candidate service goals in *CSG*(*ig*) are set to the set of prepositional phrases related to *ig.sgv*, i.e.,

$$csg_i.vsgp = vExt(ig).vPREP. \tag{4}$$

- $csg_i.nsgp$ can be the set of prepositional phrases related to *ig.sgn* or the set of prepositional phrases related to a coordinate noun *cn*∈*nExt*(*ig*).*nCONJ*, i.e.,

$$csg_i.nsgp = \begin{cases} cn.PREP, & \text{if } csg_i.sgn \text{ takes a coordinate noun } cn \text{ from } nExt(ig).nCONJ \text{ and } cn.PREP \neq null, \\ nExt(ig).nPREP, & \text{otherwise.} \end{cases} \tag{5}$$

Note that if $csg_i.sgn$ takes a coordinate noun *cn* from *nExt*(*ig*).*nCONJ*, two structural ambiguity cases will arise when either *cn.pref* or *cn.PREP* is *null*. To avoid losing any meaningful information, we supplement *cn.pref* (or *cn.PREP*) with *nExt*(*ig*).*pref* (or *nExt*(*ig*).*nPREP*) in such cases. For example, for the phrase *traffic incidents and issues*, the word *issues* is supplemented with *traffic*. This strategy is also adopted when expanding the coordinate nouns in *vExt*(*ig*).*vPREP*, *nExt*(*ig*).*nPREP*, and *cn.PREP* (*cn* ∈ *nExt*(*ig*).*nCONJ*).

Some special prepositions, e.g., *such_as* and *including*, may exist in the *vsgp* and *nsgp* parts of candidate service goals, e.g., <*create*, *visualizations*, {*such_as charts*, *such_as maps*}, *null*>. Generally speaking, there is a hypernym-hyponym relation between the noun phrases connected by these prepositions [16]. Our observations indicate that the hyponym noun phrases may be important, e.g., *charts* and *maps*. To obtain more meaningful service goals and simultaneously simplify the prepositional phrases in *vsgp* and *nsgp*, we remove these prepositions and transfer the hyponym noun phrases to the same level as the hypernym noun phrases. For example, <*create*, *visualizations*, {*such_as charts*, *such_as maps*}, *null*> is divided into three goals: <*create*, *visualizations*, *null*, *null*>, <*create*, *charts*, *null*, *null*>, and <*create*, *maps*, *null*, *null*>.

4.3 Service Goal Refinement

Candidate service goals may have several insufficiencies. First, there are words with the same lemma, for example, *retrieve*, *retrieved*, and *retrieving* have the same lemma *retrieve*. Second, to obtain important service goals of a given domain, stop words, e.g., *let* and *them*, and the words that are not highly related to the domain, e.g., *machine* and *leader* in the *Music* domain, need to be removed. The first issue is addressed by lemmatizing the words of candidate service goals using the WordNet[3] Lemmatizer packaged in the NLTK[4] toolkit. For the second issue, the built-in stop word list in the NLTK is adopted to remove common stop words; a ranked list of domain keywords is then generated by applying TF-IDF on the domain-specific services; and finally only the top *k* (e.g., 150) words of the ranked list are retained. Afterwards, there are two special types of goals: 1) *sgn* is *null* and 2) *sgn* contains only adjectives and abstract nouns, e.g., *listing* and *newest information*. The *sgn* parts of these goals are supplemented with important noun phrases

3. https://wordnet.princeton.edu/
4. http://www.nltk.org/

TABLE 3
Quality Grades for Service Goal Evaluation

Quality Grade	Explanation
Completely Matched (*CM*)	The *sgv* part and the domain core nouns contained in the *sgn*, *vsgp* and *nsgp* parts of the extracted service goal are all matched with the manually identified goal.
Matched (*M*)	The *sgv* part of the extracted service goal is matched, but the *sgn*, *vsgp*, and *nsgp* parts lack some domain core nouns or contain some unexpected domain core nouns, compared with the manually identified goal.
Missing (*Mis*)	A meaningful service goal is manually identified but not extracted by the proposed approach.
Redundant (*R*)	The extracted service goal is not important; or it is wrong (mainly caused by the incorrect sentence analysis) and not meaningful.
Wrong (*W*)	The extracted service goal is meaningful but wrong.

in the *nsgp* parts. To this end, a set of abstract nouns and a set of domain core nouns can be selected by domain analysts from the ranked domain keyword list. According to the selected domain core nouns, an important noun phrase is determined by checking whether it contains any domain core noun. For example, <*generate*, *listing*, *null*, {*of flight option*, *of last-minute airfare deal*}> is transformed into two goals: <*generate*, *flight option listing*, *null*, *null*> and <*generate*, *last-minute airfare deal listing*, *null*, *null*>.

The service goals of a service are obtained by collecting the service goals extracted from all sentences contained in its textual description.

4.4 Evaluation of Service Goal Extraction

We crawled 10,198 Web APIs belonging to 66 domains from the PW on April 24, 2014. For each API, we collected its descriptive data, including API name, category, summary, and textual description. To evaluate the proposed service goal extraction approach, we selected ten domains according to their scale and popularity: *Database*, *Mapping*, *Music*, *Photos*, *Sports*, *Storage*, *Transportation*, *Travel*, *Video*, and *Weather*. The popularity of a domain was measured according to the ratio of the APIs that have been used by existing mashups (i.e., composite services developed using APIs) in the PW to all APIs in the domain.

We randomly selected 50 APIs from each domain. The service goals extracted from the selected APIs were evaluated by human subjects. We recruited two groups of subjects: 13 computer science students (including five undergraduates, six MScs, and two PhDs) and two software engineers with more than six years of programming experience, to examine whether there are differences between the evaluation results of the two groups (with or without IT industry background). We randomly assigned two domains to each subject while ensuring that each domain was assigned to three subjects. The subjects were asked to familiarize themselves with the background knowledge of the assignments using Wikipedia and ranked domain keyword lists. We then held a meeting with the subjects to explain the process of the experiment, which consists of two phases: 1) build their own benchmarks of service goals for each selected API in their assigned domains by manually examining the textual descriptions, and 2) evaluate the service goals extracted by the proposed approach in terms of the quality grades defined in Table 3. The meanings of the quality grades were carefully explained to the subjects using some example APIs. During the meeting, for each domain, a set of abstract nouns and a set of domain core nouns were selected from the ranked domain keyword list by the three subjects assigned to the domain. The subjects were requested to perform the experiment within two weeks.

TABLE 4
Performance of Service Goal Extraction

Domain	Precision	Recall
Database	0.8635	0.8538
Mapping	0.9377	0.8692
Music	0.8677	0.8639
Photos	0.8706	0.821
Sports	0.8377	0.8907
Storage	0.8567	0.8589
Transportation	0.8456	0.8618
Travel	0.8895	0.9046
Video	0.9036	0.9008
Weather	0.8499	0.9173

Based on the evaluation results of the subjects, the performance of the service goal extraction approach was measured in terms of two popular metrics, i.e., *Precision* and *Recall*, which are defined as

$$Precision = \frac{N_{CM}+N_M}{N_{CM}+N_M+N_W+N_R}, \tag{6}$$

$$Recall = \frac{N_{CM}+N_M}{N_{CM}+N_M+N_{Mis}}, \tag{7}$$

where N_g is the number of service goals that are evaluated as g ($g \in \{CM, M, Mis, R, W\}$).

For each selected API, we computed the *Precision* and *Recall*, respectively, with respect to the evaluation results of each of three subjects, and then computed the average of the three subjects' results. Finally, we measured the performance of the approach according to the average result of 50 APIs within each domain.

Table 4 presents the performance of the approach on the ten domains. The statistics of service goals with different quality grades evaluated by three subjects in each domain are presented in Table 5. The *Precision* and *Recall* are higher than 0.8, which indicates that the majority of the extracted service goals are meaningful and most desired service goals are automatically extracted. However, we also observe that there are differences among the subjects' evaluation results, especially on *Mis* and *R* of some domains. Based on our analysis, the differences are largely caused by the fact that the subjects' judgements on the meaningfulness of some goals are different. For example, the goal <*connect*, *twitter account*, *null*, *null*> in the *Database* domain was evaluated as *CM* and *R* respectively by subjects 1 and 2. As for the differences on *Mis*, besides the differences on the subjects' judgements, another important reason is that for functional phrases containing coordinate verbs or nouns, e.g., *retrieving*

TABLE 5
Statistics of Service Goals with Different Quality Grades

Domain	Subject	# Service goals with different quality grades				
		CM	M	Mis	W	R
Database	1	126	4	25	8	10
	2	123	8	24	8	9
	3	118	7	30	4	19
	average	**122**	**6**	**26**	**6**	**12**
Mapping	1	183	5	34	5	6
	2	180	7	35	7	5
	3	179	5	32	7	8
	average	**180**	**5**	**33**	**6**	**6**
Music	1	156	17	29	9	17
	2	165	13	28	10	11
	3	146	14	19	11	28
	average	**155**	**14**	**25**	**10**	**18**
Photos	1	140	2	34	7	12
	2	126	4	25	10	21
	3	131	8	28	8	14
	average	**132**	**4**	**29**	**8**	**15**
Sports	1	111	7	14	10	14
	2	111	4	30	10	17
	3	115	5	10	12	10
	average	**112**	**5**	**18**	**10**	**13**
Storage	1	182	11	39	18	19
	2	187	6	48	15	22
	3	187	8	46	14	21
	average	**185**	**8**	**44**	**15**	**20**
Transportation	1	143	23	21	13	23
	2	159	19	32	12	12
	3	151	23	25	14	14
	average	**151**	**21**	**26**	**13**	**16**
Travel	1	174	7	20	5	23
	2	172	9	27	6	22
	3	173	17	23	4	15
	average	**173**	**11**	**23**	**5**	**20**
Video	1	190	8	27	8	18
	2	190	13	28	15	6
	3	181	5	21	13	25
	average	**187**	**8**	**25**	**12**	**16**
Weather	1	182	22	21	7	32
	2	183	28	16	8	24
	3	169	30	17	9	35
	average	**178**	**26**	**18**	**8**	**30**

data on houses and accommodations in the *Travel* domain, one or multiple missing goals were identified by different subjects. The gray colored rows in Table 5 indicate the results of the two engineers. It can be seen that there is no significant difference between the results of students and those of software engineers.

5 Domain Service Goal Construction

In this section, we introduce the definition of domain service goal, and then describe how to construct domain service goals from the service goals within a domain.

5.1 Domain Service Goal

We observe that there are many similar service goals that share a basic functionality in a domain, e.g., the service goals presented in Fig. 5b are similar because of their common basic functionality *share music*. More specifically, the basic functionality of a service goal is defined by its *sgv* part and the domain core nouns contained in its *sgn* part. Inspired by the idea of inheritance-derivation, similar service goals can be viewed as specific service goals derived from their common basic functionality, as depicted in Fig. 5. From the domain engineering perspective, the common basic functionality of similar service goals can represent the commonality knowledge on domain functionalities, while the additional parts of similar service goals can reflect the variability knowledge on domain functionalities.

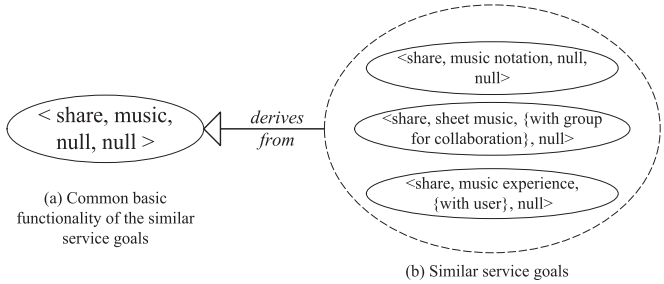


Fig. 5. Inheritance-derivation perspective of similar service goals.

To facilitate service organization and reuse, it is worthy to extract the commonality and variability of domain-specific service functionalities from similar service goals. Therefore, we must find a way to merge similar service goals in a domain. For example, the merged result of similar service goals in Fig. 5b should be <*share*, {*sheet*} *music* {*experience*, *notation*}, {*with user*, *with group for collaboration*}, *null*>, from which the domain-specific common functionality *share music* can be obtained directly. Such a merged result provides a comprehensive view of the common functionality, as well as the variability. We introduce a new concept, *domain service goal*, to define the merged result of similar service goals.

Definition 2. *A domain service goal is defined as dsg = <dsgv, dsgn, LNS, RNS, vPREP, nPREP>, where*

- *dsgv is its verb (phrase);*
- *dsgn is its core noun (phrase);*
- *LNS is the set of prefix modifiers of dsgn;*
- *RNS is the set of post modifiers of dsgn;*
- *vPREP is the set of prepositional phrases related to dsgv;*
- *nPREP is the set of prepositional phrases related to dsgn.*

5.2 Constructing Domain Service Goals

As shown in Fig. 1, the construction of domain service goals has three steps, which are described as follows.

5.2.1 Service Goal Statistics

Given a specific domain, we calculate the statistics of the extracted service goals of all services, so as to obtain a service goal list (referred to as *SGL*) sorted in descending order by frequency.

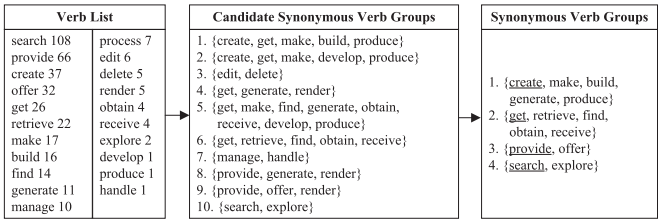


Fig. 6. Example of synonymous verb groups.

5.2.2 Synonymous Verb Replacement

Many synonymous verbs are included in the *sgv* parts of service goals, e.g. *get*, *retrieve*, and *find*. Clearly the corresponding semantically similar goals such as <*get*, *music*, *null*, *null*>, <*retrieve*, *music lyric*, *null*, *null*>, and <*find*, *sheet music*, *null*, *null*> should be merged to avoid redundancy. This issue is addressed by defining a set of synonymous verb groups. First, we count the *sgv* parts of service goals in *SGL*, resulting in a verb list V. A set of candidate synonymous verb groups (*CSVG*) are then built from V using Algorithm 2. The function getWNSyn(v) is to obtain the synonymous verbs of verb v in WordNet, denoted by *WNS*. A candidate synonymous verb group G is defined as the intersection of *WNS* and V (i.e., the synonymous verbs outside V are not considered), including v itself (steps 3–4). G is then added to *CSVG* if it is not a subset of any element in *CSVG* (steps 5–13). *CSVG* is finally refined by domain analysts according to their understanding of the corresponding domain, resulting in a set of synonymous verb groups $SVG = G_1, G_2, \ldots, G_k$. Note that $\forall G_i, G_j \in SVG, i \neq j, G_i \cap G_j = \emptyset$. A representative verb is selected for each group $G_i \in SVG$. Afterwards, each service goal *sg* in *SGL* is adjusted by replacing its *sgv* part with the representative verb of the synonymous verb group containing *sg.sgv*.

Algorithm 2. Candidate Synonymous Verb Groups Builder

Input: verb list V
Output: candidate synonymous verb groups *CSVG*
1 *CSVG* ← ∅;
2 **For each** $v \in V$
3 *WNS* ← getWNSyn(v);
4 G ← *WNS* ∩ V; G.add(v);
5 **boolean** b ← **false**, t ← **false;**
6 **For each** $G_i \in CSVG$
7 **If** G_i.containsAll(G) **then**
8 b ← **true**; **break**;
9 **If** G.containsAll(G_i) **then**
10 *CSVG*.remove(G_i);
11 t ← **true**;
12 **If** !b | | t **then**
13 *CSVG*.add(G);
14 **Return** *CSVG*;

Given the synonymous verb groups depicted in Fig. 6, <*retrieve*, *music lyric*, *null*, *null*> and <*find*, *sheet music*, *null*, *null*> are transformed into <*get*, *music lyric*, *null*, *null*> and <*get*, *sheet music*, *null*, *null*>, respectively, which can be further merged with <*get*, *music*, *null*, *null*>.

TABLE 6
Service Goals Belonging to *SG*(*share*)

< share, music, {on Twitter}, null >
< share, playlist, {with friend}, null >
< share, music playlist, null, null >
< share, sheet music, {with group for collaboration, with people for collaboration}, null >
< share, Spotify playlist, null, null >
< share, music notation, null, null >
< share, online music, null, null >
< share, music experience, {with user}, null >

5.2.3 Domain Service Goal Generation

In this step, domain service goals are generated by merging similar service goals in *SGL*. Recall that similar service goals share the same *sgv* part, the service goals in *SGL* are divided into clusters according to their *sgv* parts. Table 6 presents several service goals belonging to the cluster whose *sgv* is *share*, referred to as *SG*(*share*), in the *Music* domain. Next, a set of domain service goals *DSG*(*sgv*) are generated from a service goal cluster *SG*(*sgv*). More specifically, each *sg* ∈ *SG*(*sgv*) is added to *DSG*(*sgv*) using the following merging rules on each *dsg* ∈ *DSG*(*sgv*):

- **MR-1**: If *sg.sgn* equals *dsg.dsgn*, *dsg* is updated as follows:

$$\begin{aligned} dsg.vPREP &= dsg.vPREP \cup sg.vsgp, \\ dsg.nPREP &= dsg.nPREP \cup sg.nsgp, \\ f(dsg) &= f(dsg) + f(sg), \end{aligned} \tag{8}$$

where *f*(*dsg*) is the frequency of *dsg* in *DSG*(*sgv*) and *f*(*sg*) is the frequency of *sg* in *SGL*.

- **MR-2**: If *sg.sgn* subsumes *dsg.dsgn*, i.e.,

$$\begin{aligned} &sg.sgn = l_{sg.sgn-dsg.dsgn} \circ dsg.dsgn \circ r_{sg.sgn-dsg.dsgn}, \\ &l_{sg.sgn-dsg.dsgn} \neq null \vee r_{sg.sgn-dsg.dsgn} \neq null, \end{aligned} \tag{9}$$

dsg is updated as follows:

$$\begin{aligned} dsg.LNS &= dsg.LNS \cup l_{sg.sgn-dsg.dsgn}, \\ dsg.RNS &= dsg.RNS \cup r_{sg.sgn-dsg.dsgn}, \\ dsg.vPREP &= dsg.vPREP \cup sg.vsgp, \\ dsg.nPREP &= dsg.nPREP \cup sg.nsgp, \\ f(dsg) &= f(dsg) + f(sg), \end{aligned} \tag{10}$$

where ∘ is the string concatenation operator; $l_{sg.sgn-dsg.dsgn}$ and $r_{sg.sgn-dsg.dsgn}$ are the substrings on the left and right sides, respectively, of *sg.sgn* by subtracting *dsg.dsgn* from *sg.sgn*. For example, if *sg.sgn* is *personal music profile* and *dsg.dsgn* is *music*, $l_{sg.sgn-dsg.dsgn}$ and $r_{sg.sgn-dsg.dsgn}$ are set to *personal* and *profile*, respectively.

- **MR-3**: If *dsg.dsgn* subsumes *sg.sgn*, i.e.,

$$\begin{aligned} &dsg.dsgn = l_{dsg.dsgn-sg.sgn} \circ sg.sgn \circ r_{dsg.dsgn-sg.sgn}, \\ &l_{dsg.dsgn-sg.sgn} \neq null \vee r_{dsg.dsgn-sg.sgn} \neq null, \end{aligned} \tag{11}$$

dsg is replaced with a new domain service goal *ndsg*:

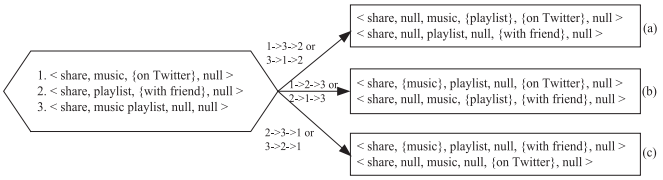


Fig. 7. Domain service goals generated by different merging orders.

$$\begin{aligned} ndsg.dsgv &= sg.sgv,\ \ ndsg.dsgn = sg.sgn,\\ ndsg.LNS &= \bigcup_{l\in dsg.LNS} l \circ l_{dsg.dsgn-sg.sgn},\\ ndsg.RNS &= \bigcup_{r\in dsg.RNS} r_{dsg.dsgn-sg.sgn} \circ r,\\ ndsg.vPREP &= dsg.vPREP \cup sg.vsgp,\\ ndsg.nPREP &= dsg.nPREP \cup sg.nsgp,\\ f(ndsg) &= f(dsg) + f(sg),\\ DSG(sgv) &= \{DSG(sgv) - dsg\} \cup ndsg, \end{aligned} \tag{12}$$

where $l_{dsg.dsgn\text{-}sg.sgn}$ and $r_{dsg.dsgn\text{-}sg.sgn}$ are the substrings on the left and right sides, respectively, of *dsg.dsgn* by subtracting *sg.sgn* from *dsg.dsgn*.

- **MR-4**: If MR-1, MR-2, and MR-3 fail after traversing the entire *DSG*(*sgv*), a new domain service goal *ndsg* is generated from *sg* and added to *DSG*(*sgv*):

$$\begin{aligned} ndsg.dsgv = sg.sgv,\ \ ndsg.dsgn = sg.sgn,\\ ndsg.vPREP = sg.vsgp,\ \ ndsg.nPREP = sg.nsgp,\\ f(ndsg) = f(sg),\ \ DSG(sgv) = DSG(sgv) \cup ndsg. \end{aligned} \tag{13}$$

Note that there can be multiple domain core nouns contained in the *sgn* part of a service goal, e.g., <*share*, *music playlist*, *null*, *null*> (*music* and *playlist* are two core nouns in the *Music* domain). Intuitively, two domain service goals should be generated from *SG*(*share*) in Table 6:

- <*share*, *{online, sheet}*, *music*, *{playlist, notation, experience}*, *{on Twitter, with user, with group for collaboration, with people for collaboration}*, *null*>
- <*share*, *{music, Spotify}*, *playlist*, *null*, *{with friend}*, *null*>

According to the merging rules described above, different domain service goals are generated from *SG*(*share*) because the merging order of three service goals may be different, as depicted in Fig. 7. To obtain the desired results, i.e., Fig. 7b, service goals with fewer domain core nouns contained in the *sgn* parts should be processed first. Following this principle, the service goals in *SG*(*sgv*) are sorted in ascending order by the number of words contained in their *sgn* parts. By doing this, during the merging process of service goals in *SG*(*sgv*), MR-3 will never be activated. The detailed domain service goal generation (*DSGG*) approach is given in Algorithm 3.

5.3 Two Improved Algorithms

Through observing the domain service goals constructed for the ten domains selected in Section 4.4 using *DSGG*, many domain service goals are not well-separated, that is, the domain core nouns contained in the *dsgn* parts are not separated from their modifiers. For example, for the goal <*purchase*, *null*, *mobile music*, *null*, *null*, *null*> in the *Music* domain, the core noun *music* in *dsgn* is not separated from its adjective modifier *mobile*. More specifically, an imperfectly separated domain service goal *dsg* can be roughly characterized by $|W(dsg.dsgn)| \geq 2 \wedge dsg.LNS = null \wedge dsg.RNS = null$, where $W(^*)$ denotes the set of words contained in phrase *. Since the domain service goals will be leveraged by domain analysts to further build richer domain knowledge, it might be a heavy burden to manually examine too many imperfectly separated goals. To mitigate this issue, we further propose two alternative solutions to reduce the number of imperfectly separated domain service goals.

Algorithm 3. DSGG

Input: service goal list *SGL* of a domain
Output: domain service goals *DSG*

```
1   DSG ← ∅;
2   SGC ← divideBySgv(SGL);
3   For each SG(sgv) ∈ SGC
4       DSG(sgv) ← ∅;
5       Sort service goals in SG(sgv) in ascending order by
        the number of words contained in their sgn parts.
6       For each sg ∈ SG(sgv)
7         boolean b ← false;
8         For each dsg ∈ DSG(sgv)
9             If sg.sgn equals dsg.dsgn then
10              update dsg by MR-1; b ← true;
11            If sg.sgn subsumes dsg.dsgn then
12              update dsg by MR-2; b ← true;
13        If !b then // MR-4
14            ndsg ← generateDsg(sg);
15            DSG(sgv).add(ndsg);
16      DSG.addAll(DSG(sgv));
17  Return DSG;
```

5.3.1 DCN-Based DSGG

As explained above, the issue of imperfectly separated domain service goals is that the domain core nouns contained in the *dsgn* parts are not separated from their modifiers. Thus, to obtain well-separated domain service goals, a set of domain core nouns can be selected from the ranked domain keyword list, as described in Section 4.4. Afterwards, domain service goals can be generated from *SG*(*sgv*) as follows. For each domain core noun *dcn*, the set of service goals in *SG*(*sgv*) for which the *sgn* part contains *dcn* are obtained using $SG(dcn) = \{sg \mid sg \in SG(sgv) \wedge sg.sgn = l_{sg.sgn-dcn} \circ dcn \circ r_{sg.sgn-dcn}\}$. A domain service goal *dsg* is then generated from *SG*(*dcn*):

$$\begin{aligned} dsg.dsgv = sgv,\ \ dsg.dsgn = dcn,\\ dsg.LNS = \bigcup_{sg\in SG(dcn)} l_{sg.sgn-dcn},\ \ dsg.RNS = \bigcup_{sg\in SG(dcn)} r_{sg.sgn-dcn},\\ dsg.vPREP = \bigcup_{sg\in SG(dcn)} sg.vsgp,\ \ dsg.nPREP = \bigcup_{sg\in SG(dcn)} sg.nsgp,\\ f(dsg) = \sum_{sg\in SG(dcn)} f(sg). \end{aligned} \tag{14}$$

More details of this improved *DSGG* approach based on domain core nouns, referred to as *DCN-based DSGG*, are given in Algorithm 4.

Algorithm 4. DCN-based DSGG

Input: service goal list *SGL* of a domain, and a set of domain core nouns *DCN*
Output: domain service goals *DSG*

```
1  DSG ← ∅;
2  SGC ← divideBySgv(SGL);
3  For each SG(sgv) ∈ SGC
4      DSG(sgv) ← ∅;
5      For each dcn ∈ DCN
6          SG(dcn) ← ∅;
7          For each sg ∈ SG(sgv)
8              If sg.sgn.contains(dcn) then
9                  SG(dcn).add(sg);
10         dsg ← generateDsg(SG(dcn));
11         DSG.add(dsg);
12 Return DSG;
```

5.3.2 Two-Round DSGG

We note that many domain core nouns contained in the *dsgn* parts of imperfectly separated domain service goals have occurred as *dsgn* of some well-separated domain service goals. A well-separated domain service goal *dsg* is roughly characterized by $dsg.LNS \neq null \vee dsg.RNS \neq null$. For example, in the *Music* domain, the core noun *music* contained in the *dsgn* part of <*purchase*, *null*, *mobile music*, *null*, *null*, *null*> occurs as the *dsgn* part of <*share*, {*online*, *sheet*}, *music*, {*experience*, *notation*, *playlist*, *file*}, {*on Twitter*, ...}, *null*>. Therefore, many imperfectly separated goals can be refined using the *dsgn* parts of well-separated goals. Inspired by this idea, we develop a new *DSGG* approach which contains two rounds of processing:

- *The first round*: A set of domain service goals are generated using Algorithm 3. The *dsgn* parts of well-separated goals are collected, resulting in a noun (phrase) set.
- *The second round*: A new set of domain service goals are generated based on the collected noun (phrase) set using Algorithm 4.

More details of this improved approach, referred to as *Two-Round DSGG*, are given in Algorithm 5.

Algorithm 5. Two-Round DSGG

Input: service goal list *SGL* of a domain
Output: domain service goals *DSG*

```
1  DSGN ← ∅;
2  DSG ← generateDSG(SGL); // invoke Algorithm 3
3  For each dsg ∈ DSG
4      If dsg.LNS ≠ null || dsg.RNS ≠ null then
5          DSGN.add(dsg.dsgn);
6  DSG ← generateDSG(SGL, DSGN); // invoke Algorithm 4
7  Return DSG;
```

5.4 Evaluation of Domain Service Goal Construction

We performed the *DSGG* approach on the ten domains used in Section 4.4. In the step of synonymous verb replacement, a set of candidate synonymous verb groups (*CSVG*) were built for each domain using Algorithm 2 and refined by the three subjects who had evaluated the service goals of the domain. More specifically, the subjects refined *CSVG* independently and then discussed debatable parts together to reach consensus on the final set of synonymous verb groups.

Table 7 presents several domain service goals of the *Music* domain, along with their frequencies. As can be seen from the table, the *dsgv* and *dsgn* parts represent the commonality of domain-specific service functionalities, while the variability is reflected by *LNS*, *RNS*, *vPREP*, and *nPREP*. The frequencies of domain service goals could be used for different analysis tasks. For example, the goals with high frequencies may be used to produce representative domain functionalities, while the goals with low frequencies may contribute to identifying functionalities that have competitive advantage when developing new services since they have not been sufficiently developed or concerned by existing services.

5.4.1 Questionnaire

We conducted a questionnaire to evaluate the domain service goals. In the stage of preparation, we held a meeting with the 15 subjects who had participated in the evaluation of service goal extraction again. We explained to the subjects the definition of domain service goals, as well as the process of the experiment: browse the domain service goals (each subject was assigned the same domains as in the previous experiment) and rate the degree of agreement to the statements listed in Table 8 on a scale of 1-7 (where "7" indicates full agreement). The subjects were asked to perform the experiment within one week.

The scores received from 15 subjects are summarized in Table 8. The columns "1-3", "4", and "5-7" present the number of scores in the range of 1-3, the number of score of 4, and the number of scores in the range of 5-7, respectively. The column "average" presents the average of all 15 scores on each statement. We can see that the average scores on statements 1, 4, 5, and 6 are higher than 5.0, indicating that domain service goals can contribute to: 1) organizing service goals in a domain, 2) understanding domain functionalities, 3) distinguishing the commonality and variability of domain functionalities, and 4) defining service requests for service search.

In contrast, the average scores on statements 2 and 3 are relatively low, which indicates that domain service goals are insufficient in identifying domain functionalities that are less concerned by existing APIs and can not offer sufficient help for requirements analysts in functionality design of APIs. The feedback of subjects reveals some issues related to those two statements. For statement 2, the major issue is that there are too many goals with low frequencies in a domain, which makes it difficult to find the ones of interest. For statement 3, the major issue is that semantically relevant functionalities, e.g., *provide booking for hotel* and *support reservation cancellation*, are not connected, and these relations are important in functionality design. These insufficiencies suggest the direction of our future work, particularly in creating relations among domain service goals.

5.4.2 Comparison of Improved DSGG Approaches

As stated earlier, *DSGG* will produce many imperfectly separated domain service goals, which would be a big obstacle

TABLE 7
Several Domain Service Goals of the *Music* Domain

dsgv	LNS	dsgn	RNS	vPREP	nPREP	frequency
provide	legal, online, unlimited, streaming, latest, upcoming, . . .	music	download, event information, fashion information, lyric service, metadata, news blog, processing engine, reporting, solution, . . .	for player in car, for player in mobile, from home entertainment system to set-top box, from home entertainment system to in-car audio system, to music fan, to web developer, . . .	on subscription basis, with DRM, with ringtone	18
get	suggested	artist	audio, blog, content, information, list, news, profile, url, video	about label, on performance, on release, with similarity		14
search	beat, new, Spotify	music	catalog, collection, information, playlist	for use in slideshow, for use in video, with Spotify metadata, . . .	from blog, from music site	13
share	online, sheet	music	experience, file, notation, playlist	on Twitter, through soundrop room, with group for collaboration, with people for collaboration, with user		13
upload	sheet	music	collection, track	in cloud, to community		6
share	Spotify, music	playlist		with friend		4
purchase		mobile music				1
display		relevant lyric text				1

for domain analysts to analyze the goals. Two improved approaches, namely *DCN-based DSGG* and *Two-Round DSGG*, are proposed to reduce the number of imperfectly separated goals. We performed these approaches on each of the ten domains. For *DCN-based DSGG*, we sorted the domain core nouns selected for each domain (as described in Section 4.4) in descending order by frequency, and performed the approach using the top k domain core nouns.

The 15 subjects were asked to examine the imperfectly separated goals produced by *DSGG* in their assigned domains, and to evaluate whether they are true (*T*) or false (*F*), where *T* means that an imperfectly separated goal *dsg* is actually not well-separated, while *F* indicates that *dsg* does not need to be separated (e.g., *dsg* is wrong or meaningless). After the subjects completed the experiment independently, the three subjects of each domain discussed debatable goals, if any, to reach an agreement. Table 9 presents the numbers of true and false imperfectly separated goals, denoted by T_n and F_n, respectively, evaluated for *DSGG*.

To measure the effectiveness of the two improved approaches *DCN-based DSGG* and *Two-Round DSGG* in reducing imperfectly separated goals, we defined two metrics: true reduction ratio (*TRR*) and false reduction ratio (*FRR*), where *TRR* indicates the ratio of the reduced true imperfectly separated goals to the true ones produced by *DSGG*, and *FRR* is the ratio of the reduced false imperfectly separated goals to the false ones produced by *DSGG*. The higher *TRR* is better, while the lower *FRR* is better.

$$TRR = \frac{T_n \text{ of } DSGG - T_n \text{ of an improved approach}}{T_n \text{ of } DSGG}, \quad (15)$$

$$FRR = \frac{F_n \text{ of } DSGG - F_n \text{ of an improved approach}}{F_n \text{ of } DSGG}, \quad (16)$$

where "an improved approach" denontes *DCN-based DSGG* or *Two-Round DSGG*.

In Table 9, *DCN-based DSGG* (top k) refers to the *DCN-based DSGG* using the top k domain core nouns. It can be seen that both *Two-Round DSGG* and *DCN-based DSGG* can effectively reduce the imperfectly separated goals. For *DCN-based DSGG*, it achieved higher *TRR* using more domain core nouns. The average *TRR* of *DCN-based DSGG* (top 10) is a little lower than that of *Two-Round DSGG*, while

TABLE 8
Summary of Answers to Questionnaire

	Satement	1-3	4	5-7	average
1	The domain service goals (DSG) provide an efficient way to organize the extracted service goals.	0	5	10	5.07
2	The DSG is helpful in identifying domain functionalities that are less concerned by existing Web APIs (in the PW).	4	4	7	4.27
3	The DSG can assist requirement analysts in functionality design of services.	6	3	6	3.93
4	The DSG facilitates understanding of domain functionalities.	1	2	12	5.53
5	The DSG helps in distinguishing the commonality and variability of domain functionalities.	0	0	15	5.80
6	The DSG can help users define service requests during Web service search.	0	2	13	5.60

TABLE 9
Statistics of Imperfectly Separated Domain Service Goals

Domain	DSGG		Two-Round DSGG				DCN-based DSGG (top 10)				DCN-based DSGG (top 30)				DCN-based DSGG (top 50)			
	T_n	F_n	T_n	TRR	F_n	FRR	T_n	TRR	F_n	FRR	T_n	TRR	F_n	FRR	T_n	TRR	F_n	FRR
Database	120	51	45	62.5%	32	37.26%	29	75.83%	27	47.06%	14	88.33%	19	62.75%	4	96.67%	13	74.51%
Mapping	430	133	123	71.4%	77	42.11%	227	47.21%	105	21.05%	149	65.35%	94	29.32%	95	77.91%	87	34.57%
Music	219	66	87	60.27%	49	25.76%	111	49.32%	58	12.12%	61	72.15%	50	24.24%	20	90.87%	46	30.3%
Photos	134	58	53	60.45%	44	24.14%	51	61.94%	42	27.59%	19	85.82%	35	39.66%	12	91.05%	35	39.66%
Sports	207	86	68	67.15%	54	37.21%	99	52.17%	62	27.91%	52	74.88%	53	38.37%	34	83.58%	51	40.7%
Storage	161	61	81	49.69%	46	24.59%	59	63.35%	39	36.07%	29	81.99%	33	45.9%	13	91.93%	30	50.82%
Transportation	329	121	177	46.2%	83	31.41%	191	41.95%	88	27.27%	103	68.69%	63	47.93%	65	80.24%	54	55.37%
Travel	389	62	167	57.07%	47	24.19%	192	50.64%	47	24.19%	105	73.01%	44	29.03%	66	83.03%	37	40.32%
Video	229	43	99	56.77%	32	25.58%	109	52.4%	38	11.63%	63	72.49%	34	20.93%	40	82.53%	33	23.26%
Weather	160	43	81	49.38%	27	37.21%	77	51.88%	27	37.21%	41	74.38%	21	51.16%	20	87.5%	18	58.14%
average	**237**	**72**	**98**	**58.09%**	**49**	**30.94%**	**114**	**54.67%**	**53**	**27.21%**	**63**	**75.71%**	**44**	**38.93%**	**36**	**86.53%**	**40**	**44.77%**

the average *TRR* of *DCN-based DSGG* (top 50) is higher than 85 percent and outperforms *Two-Round DSGG*. Although the *FRR* of *DCN-based DSGG* also increased as more domain core nouns were used, the number of the reduced false imperfectly separated goals is much smaller than that of the reduced true ones. This could be acceptable in practice. It should be pointed out that *Two-Round DSGG* is an automatic approach and its *TRR* performance is limited because the domain service goals produced after the second round cannot be further improved. In contrast, *DCN-based DSGG* is semi-automatic because it depends on a predefined set of domain core nouns. This will be helpful for users who want to obtain domain service goals about the topics of interest.

6 Contributions, Limitations, and Threats to Validity

6.1 Contributions

The major contribution of this paper is the proposed approach of mining domain knowledge on service goals from textual service descriptions. The NLP-based method designed for service goal extraction can be applied to mining functional goals from other text resources, e.g., requirements documents and software product reviews. The domain service goal construction approach can extract the commonality and variability among service goals in a domain. The domain service goals offer a comprehensive view of domain-specific service functionalities.

Moreover, we foresee our approach to be beneficial to service-oriented requirements analysis and service request specification.

Service-oriented requirements analysis: When developing a new service for a specific topic, it is usually not an easy task for requirements analysts to determine the service functionalities. This task can be facilitated using the domain knowledge on service goals. Particularly when the functionalities are not explicitly specified by users, this knowledge can help recommend missing or potentially interesting functionalities to analysts. In addition, the knowledge embodied in service descriptions can also be complemented with more expert-driven analysis activities of requirements analysts. On the one hand, the service goals extracted from service descriptions can be an important source of goals in goal modeling. On the other hand, the relations between the goals in service descriptions and service compositions can be leveraged by analysts in analyzing the goal composition/decomposition as well as goal dependency/conflict relations, which can further promote the service composition design. For example, the co-occurrence relation among extracted goals *search music playlist*, *search music by song*, *search music by artist*, and *search music collection*, may help analysts in modeling the decomposition relation (e.g., the means-ends realtion in the i* framework) for the goal *search music*. The extraction and analysis of these relations in goal modeling will be our future work.

Service request specification: Service discovery aims to help users find services that can be reused. While many approaches [22], [25], [26], [27] have been proposed for matching service requests with service descriptions, little attention has been paid to making an informed request. Based on the mined domain knowledge, users can get a better understanding of service functionalities of their domains of interest; and it will then be easier to specify service requests that reflect their requirements. Alternatively, relevant goals can be recommended for a submitted request, from which the user can select appropriate ones as a new request for service discovery.

6.2 Limitations

There are some limitations in the proposed approach. As for the service goal extraction approach, it can be observed from Table 5 that a number of service goals identified by the subjects fail to be extracted by the approach. Based on our analysis, the missing goals are mainly caused by the following reasons. First, the approach does not cope with the service functionalities expressed as noun phrases, e.g., *image uploading* and *order management*. Second, some useful dependencies, e.g., *xcomp* and *mark*, have not been covered by the approach, which results in some missing functionalities, such as *which hotel is quickest to access the sights*. Finally, the approach depends on the grammatical analysis of sentences produced by the Stanford Parser. For the sentences that are not parsed correctly, the service goals contained therein cannot be extracted accordingly.

As for the domain service goal construction approach, to merge semantically similar service goals, we adjust the service goals by defining a set of synonymous verb groups. However, merging some semantically similar goals still

fails, e.g., <*cancel*, *reservation*, *null*, *null*> and <*support*, *reservation cancellation*, *null*, *null*>. Although *cancel* and *cancellation* are semantically equivalent, our approach cannot merge those two service goals because the two words locate at different parts (i.e., *sgv* and *sgn*) of two goals. Moreover, the case of "a verb with multiple meanings" has not been considered in goal merging.

6.3 Threats to Validity

There are a number of internal and external validity threats that should be discussed. The *internal validity* concerns the bias and repeatability of experiment results. One major threat to the internal validity is the bias of the subjects' judgements. We recruited 15 subjects to perform the evaluation of service goals, and assigned three subjects to each domain so as to decrease the bias of subjects. For the evaluation of imperfectly separated domain service goals, the benchmark of each domain was also built by three subjects: they first evaluated the imperfectly separated goals independently and then discussed the debatable ones together to reach an agreement. Another threat to the internal validity is the subjects' familiarity with their assigned domains. To mitigate this threat, before conducting the experiments, we leveraged a set of domain-related Wikipedia pages and ranked domain keyword lists to help the subjects be familiar with the domains. Moreover, to ensure that the experiment results can be reproduced, we described the proposed approach in a step-by-step fashion, as well as some key heuristics.

The *external validity* is related to the generalizability of experiment results. One external validity threat is that the APIs used in experiments were only crawled from a single service registry, the PW. In addition, the PW relies on the input from API providers and other users for its content, which will introduce the danger of selection of domains for experimentation. In particular, in our experiments, we selected ten large and popular domains in the PW. When new APIs and new mashups are registered in the PW, the scale and popularity of a domain will change over time. Another external validity threat is the generalizability across various subject samples. Besides the student samples, we recruited two software engineers to perform the evaluation. The evaluation results presented in Table 5 indicate that there is no significant difference between the students and engineers. Although we have taken some mitigation strategies, an in-depth evaluation with more subjects from industries on multiple datasets besides the PW is still needed to obtain more reliable results.

7 Conclusion and Future Work

This paper presented an approach of mining domain knowledge on service goals, which aims to manage the domain-specific functionalities contained in textual service descriptions. An NLP-based method was designed to extract service goals from textual descriptions of services. An approach was further proposed to organize the extracted service goals within a domain. Experiments conducted on a real-world dataset sourced from the ProgrammableWeb demonstrated the effectiveness of the proposed approach.

In the future, we will improve the proposed approach by solving the limitations discussed in Section 6.2. For example, the service goal extraction approach will be enhanced by dealing with service functionalities expressed as noun phrases and considering more useful dependencies. The domain service goal construction approach will be refined by solving semantically similar service goals described in different forms and the polysemous verbs. In addition, we plan to establish semantic relationships among service goals by exploring the structures of mashups or other service compositions, as well as external domain knowledge bases such as Wikipedia. To support goal modeling for service development, how to supplement extracted domain service goals with an existing modeling methodology, e.g., the i* framework, will also be further investigated.

Acknowledgment

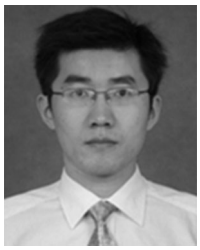
This research was supported by the National Basic Research Program of China (No. 2014CB340404), National Key Research and Development Program of China (No. 2016YFB0800401), the Wuhan Yellow Crane Talents Program for Modern Services Industry, and the Strategic Team-Building of Scientific and Technological Innovation in Hubei Province (C-type Project), and National Science Foundation of China (Nos. 61272111, 61373037 and 61672387). Jian Wang is the corresponding author.

References

[1] M. Pantazoglou and A. Tsalgatidou, "A generic query model for the unified discovery of heterogeneous services," *IEEE Trans. Serv. Comput.*, vol. 6, no. 2, pp. 201–213, 2013.
[2] M. Bano, D. Zowghi, N. Ikram, and M. Niazi, "What makes service oriented requirements engineering challenging? A qualitative study," *IET Softw.*, vol. 8, no. 4, pp. 154–160, 2014.
[3] M. Bano and N. Ikram, "Issues and challenges of requirement engineering in service oriented software development," in *Proc. Int. Conf. Softw. Eng. Adv.*, 2010, pp. 64–69.
[4] L. Chen, "WTCluster: Utilizing tags for web services clustering," in *Proc. Int. Conf. Service-Oriented Comput.*, 2011, pp. 204–218.
[5] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating web APIs on the World Wide Web," in *Proc. Eur. Conf. Web Serv.*, 2010, pp. 107–114.
[6] B. Zhao, G. J. Cai, and Z. Jin, "Semantic approach for service oriented requirements modeling," in *Proc. Int. Federation Info. Process.*, 2010, pp. 35–44.
[7] C. L. Lee and A. Liu, "A goal-driven approach for service request modeling," *Int. J. Intell. Syst.*, vol. 25, no. 8, pp. 733–756, 2010.
[8] J. Wang, K. He, B. Li, W. Liu, and R. Peng, "Meta-models of domain modeling framework for networked software," in *Proc. Int. Conf. Grid Cooperative Comput.*, 2007, pp. 878–886.
[9] J. Wang, Z. W. Feng, J. Zhang, P. C. K. Hung, K. He, and L. J. Zhang, "A Unified RGPS-Based Approach Supporting Service-Oriented Process Customization," *Web Services Foundations*. Berlin, Germany: Springer, 2014, pp. 657–682.
[10] [Online]. Available: http://nlp.stanford.edu/software/dependencies_manual.pdf
[11] X. Franch, L. Lidia, C. Carlos, and C. Daniel, "The i*, Framework for goal-oriented modeling," *Domain-Specific Conceptual Modeling*. Berlin, Germany: Springer, 2016, pp. 485–506.
[12] B. Raadt, J. Gordijn, and E. Yu, "Exploring web services from a business value perspective," in *Proc. IEEE Int. Conf. Requirements Eng.*, 2005, pp. 53–62.
[13] J. Zhang, J. Wang, P. C. K. Hung, Z. Li, N. Zhang, and K. He, "Leveraging incrementally enriched domain knowledge to enhance service categorization," *Int. J. Web Serv. Res.*, vol. 9, no. 3, pp. 43–66, 2012.
[14] K. Souza and M. Fantinato, "WS&i*-RGPS: An approach to service-oriented requirements engineering based on RGPS meta-models," in *Proc. IEEE Int. Symp. Serv. Oriented Syst. Eng.*, 2014, pp. 76–81.
[15] A. Maedche and S. Staab, "Ontology Learning for the Semantic Web," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 72–79, Mar. 2001.

[16] P. Cimiano, A. Hotho, and S. Staab, "Learning concept hierarchies from text corpora using formal concept analysis," *J. Artif. Intell. Res.*, vol. 24, pp. 305–339, 2005.
[17] A. B. Rios-Alvarado, I. Lopez-Arevalo, and V. J. Sosa-Sosa, "Learning concept hierarchies from textual resources for ontologies construction," *Expert Syst. Appl.*, vol. 40, pp. 5907–5915, 2013.
[18] A. Segev and Q. Z. Sheng, "Bootstrapping ontologies for web services," *IEEE Trans. Serv. Comput.*, vol. 5, no. 1, pp. 33–44, Jan. 2012.
[19] M. Sabou, C. Wroe, C. Goble, and H. Stuckenschmidt, "Learning domain ontologies for Semantic Web service descriptions," *J. Web Semantics*, vol. 3, pp. 340–365, 2005.
[20] Y. J. Lee and C. S. Kim, "A learning ontology method for RESTful Semantic Web services," in *Proc. IEEE Int. Conf. Web Serv.*, 2011, pp. 251–258.
[21] Y. Jung, Y. Cho, Y. Park, and T. Lee, "Automatic tagging of functional-goals for goal-driven Semantic service discovery," in *Proc. Int. Conf. Semantic Comput.*, 2013, pp. 212–219.
[22] N. Zhang, K. He, J. Wang, and Z. Li, "WSGM-SD: An approach to RESTful service discovery based on weighted service goal model," *Chinese J. Electron.*, vol. 25, no. 2, pp. 256–263, 2016.
[23] C. Rolland, C. Souveyet, and C. B. Achour, "Guiding goal modeling using scenarios," *IEEE Trans. Softw. Eng.*, vol. 24, no. 12, pp. 1055–1071, Dec. 1998.
[24] M. Stevenson and M. A. Greenwood, "Comparing information extraction pattern models," in *Proc. Associat. Comput. Linguistics*, 2006, pp. 12–19.
[25] G. Cassar, P. Barnaghi, and K. Moessner, "Probabilistic matchmaking methods for automated service discovery," *IEEE Trans. Serv. Comput.*, vol. 7, no. 4, pp. 654–666, Oct.-Dec. 2013.
[26] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 260–275, Apr.-Jun. 2012.
[27] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An integrated Semantic Web service discovery and composition framework," *IEEE Trans. Serv. Comput.*, vol. 9, no. 3, pp. 537–550, Jul.-Aug. 2016.
[28] M. Robeer, G. Lucassen, M. E. M. Jan, F. Dalpiaz, and S. Brinkkemper, "Automated extraction of conceptual models from user stories via NLP," in *Proc. IEEE Int. Conf. Requirements Eng.*, 2016, pp. 196–205.
[29] S. Ghosh, D. Elenius, W. Li, L. Patrick, S. Natarajan, and S. Wilfried, "ARSENAL: Automatic requirements specification extraction from natural language," in *Proc. NASA Formal Methods*, 2016, pp. 472–476.
[30] E. Casagrande, S. Woldeamlak, W. L. Woon, H. H. Zeineldin, and D. Svetinovic, "NLP-KAOS for systems goal elicitation: Smart metering system case study," *IEEE Trans. Softw. Eng.*, vol. 40, no. 10, pp. 941–956, Oct. 2014.
[31] J. Wang, N. Zhang, C. Zeng, Z. Li, and K. He, "Towards services discovery based on service goal extraction and recommendation," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2013, pp. 65–72.
[32] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn Treebank," *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.
[33] T. H. Nguyen, J. Grundy, and M. Almorsy, "Rule-based extraction of goal-use case models from text," in *Proc. Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.*, 2015, pp. 591–601.

Neng Zhang received the BS degree from Wuhan University, China, in 2012. He is working toward the PhD degree in the State Key Laboratory of Software Engineering, Wuhan University, China. His research interests include services computing and knowledge mining.



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

Yutao Ma received the PhD degree from Wuhan University, China, in 2007. He is now an associate professor in the State Key Laboratory of Software Engineering, Wuhan University, China. His current research interests focus on software engineering and services computing. He is a member of the IEEE.