A Web Service Discovery Approach Based on Common Topic Groups Extraction

Jian Wang, *Member, IEEE*, Panpan Gao, Yutao Ma, *Member, IEEE*, Keqing He, *Senior Member, IEEE*, and Patrick C.K. Hung, *Member, IEEE*

***Abstract*—Web services have attracted much attention from distributed application designers and developers because of their roles in abstraction and interoperability among heterogeneous software systems, and a growing number of distributed software applications have been published as Web services on the Internet. Faced with the increasing numbers of Web services and service users, researchers in the services computing field have attempted to address a challenging issue, i.e. how to quickly find the suitable ones according to user queries. Many previous studies have been reported towards this direction. In this paper, a novel Web service discovery approach based on mining and matching common topic groups is presented. The proposed approach mines common topic groups from the service-topic distribution matrix generated by topic modeling, and the extracted common topic groups can then be leveraged to match user queries to relevant Web services, so as to make a better trade-off between the accuracy of service discovery and the number of candidate Web services. Experiment results conducted on two publicly-available data sets demonstrate that, compared with several widely used approaches, the proposed approach can maintain the performance of service discovery at an elevated level by greatly decreasing the number of candidate Web services, thus leading to faster response time.**

***Index Terms*—Common topic group, Web service discovery, topic model, query response time.**

I. Introduction

With the rapid development of the service-oriented computing and cloud computing, an increasing number of distributed software applications have been composed by reusing the capabilities offered by existing Web services (also known as Web APIs), which can be discovered and orchestrated collectively to deliver the desired functionality [1]. The number of publicly-available Web services is thus increasing continuously on the Internet. For example, Programmableweb[1] has published over 16,000 Web services by January 1, 2017, almost increased to three times as compared with three years ago. Many well-known IT companies have also launched their own Web service market places, e.g. Amazon's Web Service Marketplace[2], Microsoft's Azure Marketplace[3], and Baidu's API store[4]. Meanwhile, the number of service users is also dramatically increased during the past decade since these online services can be accessible to ubiquitous mobile devices. As a result, how to quickly find suitable Web services for users becomes an important issue in the services computing field.

Nowadays, most UDDI (Universal Description Discovery and Integration) registries are no longer publicly-available on the Internet. Conversely, Web service search engines or online Web service directories become increasingly popular. However, Web service search engines that rely on keyword matching always suffer from a lack of sufficient keywords in Web service descriptions or from using synonyms or variations of predefined keywords [2]. Many semantic Web service discovery approaches [3, 4, 33] that can retrieve appropriate Web services have been proposed towards this issue in recent years, but semantically annotating attributes of Web services using domain ontologies is still a time-consuming task, which brings lots of difficulties in applying these approaches in practice. Therefore, there is an urgent need for an effective Web service discovery approach.

Web service discovery normally refers to retrieving suitable Web services that can meet both functional requirements as well as nonfunctional (also known as quality of service) requirements of users. This paper aims to find Web services relevant to users' functional requirements as quickly as possible. Generally speaking, the functionality of a Web service is embodied in its description document. Therefore, the problem of Web service discovery considered in this paper is formally defined as follows. Suppose that $S = \{s_1, s_2, \ldots, s_n\}$ is a set of Web services published in a repository, where $s_i (i \in [1, n])$ is a Web service which can be characterized by its attributes including name, description, operation, input, and output. Given a user query sq, how to retrieve the most relevant Web services to sq from S as quickly as possible?

This work was supported by the National Basic Research Program of China (973 Program) under Grant No. 2014CB340404, the National Key Research and Development Program of China under Grant No. 2016YFB0800400, the National Natural Science Foundation of China under Grant Nos. 61272111, 61373037, and 61672387, 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).

J. Wang, P. Gao, Y. Ma, and K. He are with the State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China (e-mails: {jianwang, g_panpan, ytma, hekeqing}@whu.edu.cn). P. Hung is with the Faculty of Business and IT, University of Ontario Institute of Technology, Oshawa, ON, Canada (email: patrick.hung@uoit.ca). Y. Ma is the corresponding author.

An early version of this paper has been published in the Proceedings of the 9th Asia-Pacific Services Computing Conference.

[1] http://www.programmableweb.com/
[2] https://aws.amazon.com/marketplace/
[3] https://azuremarketplace.microsoft.com/
[4] http://apistore.baidu.com/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

An intuitive solution to this problem is to compare and rank the similarity between *sq* and each Web service s_i. Considering the fact that the number of Web services registered in such a large-scale repository is usually very large and might continually increases over time, it will be computationally expensive to compare a user query to each of these Web services. Many efforts have been made towards this issue. For example, several clustering technologies that can cluster similar Web services into various groups have been widely used in Web service discovery [2, 13, 26, 31, 36]. By organizing Web service descriptions into clusters in advance, service clustering can contribute to improving the efficiency of Web service discovery to a large extent. Topic models such as LDA (Latent Dirichlet Allocation) [7], which have been widely used among these clustering technologies, can be used to extract unobserved groups that explain why some parts of the documents are similar and capture the underlying domain semantics. Since the numbers of topics and Web services remain very large, the performance of Web service discovery still has much room to improve, though topic models can effectively support Web service discovery.

To this end, this paper presents a new concept *Common Topic Group* (*CTG*), which is used to further organize clustered Web services, to improve the performance of Web service discovery. The basic idea of CTG is that the similarity between two Web services will be higher, if they share more similar distribution probabilities over multiple topics. A ctg denotes the Web services that share the same probability grade under each topic within a specified scope. For example, Fig. 1 shows the distribution of four Web services (s_1-s_4) over five topics (t_1-t_5). According to Fig. 1(a) where the topic distribution probabilities have four grades (0 and g_1-g_3), Web services s_1 and s_3 can be grouped together to form a ctg because they have the same probability grade over topics t_1, t_3 and t_5. If the topic distribution probabilities are mapped into two grades, i.e. 0 and 1, s_4 can be merged into the group that consists of s_1 and s_3 (see Fig. 1(b)).

	t_1	t_2	t_3	t_4	t_5
s_1	g_1	0	g_3	0	g_2
s_2	0	0	0	g_2	g_1
s_3	g_1	g_1	g_3	0	g_2
s_4	g_2	0	g_1	0	g_1

(a) topic distribution with four grades

	t_1	t_2	t_3	t_4	t_5
s_1	1	0	1	0	1
s_2	0	0	0	1	1
s_3	1	1	1	0	1
s_4	1	0	1	0	1

(b) topic distribution with two grades

Fig. 1. Topic distribution of Web services: An example.

Based on CTG, all the Web services published in a service repository can be organized firstly according to the common topic groups they share. Given a service query with the topic distribution probability $sq = (g_1, 0, g_3, 0, g_2)$, s_1 and s_3 will then be retrieved as relevant Web services based on CTG matching, and s_4 will also be regarded as a relevant Web service with a relative lower rank using the ctg that is generated in terms of two grades. In this example, it is unnecessary to further compute the similarity between each Web service and the query. Only if the number of Web services retrieved based on CTG matching is less than the expected number, shall we compute the similarities between the remaining Web services and the query based on the service-topic distribution and topic-word distribution generated by topic models. Therefore, we aim to reduce the search space of candidate Web services using CTG matching as more as possible, so as to improve the response time of Web service discovery.

In short, the main contributions of this research are described as follows.

- A new concept *Common Topic Group* is presented to organize Web services based on topic models. An algorithm of grouping Web services that share similar distribution probabilities over multiple topics is also presented, and the size of a ctg in this algorithm can be adjusted flexibly.
- A Web service discovery approach using CTG matching is proposed. First, the topic distribution of a given user query is estimated based on topic models; second, ctgs related to the topics of the query are ranked; third, Web services relevant to those highly-ranked ctgs are selected and sorted.
- We conduct experiments on two real-world data sets to evaluate the proposed approach. Experimental results show that the proposed approach can greatly decrease the number of candidate Web services, and thus reduces the response time of Web service discovery while keeping its accuracy at an elevated level.

The remainder of this paper is organized as follows. Section II discusses the related work. In Section III, we present the problem definition of common topic groups and give an overview of the proposed approach. Section IV presents the algorithms to mine common topic groups in detail. Section V introduces the approach to CTG-based Web service matching and ranking. The experimental results and analysis are given in Section VI. Finally, Section VII summarizes this paper and puts forward our future work.

II. Related Work

This paper is closely related to the field of Web service discovery, which has been a hot topic in the services computing field during the past decade. Many approaches in this field considered both functional and non-functional attributes of Web services. Because our work focuses mainly on Web service discovery based on functional requirements, we only discuss the related work in this direction.

Earlier studies in this area usually exploited the structure of WSDL (Web Service Description Language) documents to perform profile-based service matching and discovery. Due to the limitations of keyword-based service matching such as low recall and low accuracy, many progresses have been made to improve the discovery results. Explicit semantics-based service discovery approaches and machine learning-based service discovery approaches are the two major directions among these efforts.

A. Explicit Semantics-Based Service Discovery

Explicit semantics-based approaches incorporate with Semantic Web technologies to improve the performance of Web service matchmaking. Various user-defined dictionaries,

corpora, or ontologies were leveraged to provide semantics for Web service descriptions. For example, Klusch *et al.* [4] provided a hybrid semantic matchmaker named SAWSDL-MX for Web services described in WSDL, which integrated semantic and structural similarities between Web services in service matchmaking. Similarly, OWLS-MX [10] and WSMO-MX [11] were presented by integrating syntactic concept similarity with logic-based reasoning into the discovery of services represented in OWL-S (Web Ontology Language for Services) and WSMO (Web Service Modeling Ontology), respectively. Roman *et al.* [34] presented WSMO-Lite, a lightweight ontology of Web service semantics, to semantically annotate RESTful Web services. The experimental results in these works showed that the use of ontologies in Web service discovery could greatly improve the quality of Web service discovery.

The main limitation of these explicit semantics-based approaches is that, due to heavy dependencies on domain ontologies as well as the complexity in the service discovery process, it is very difficult for them to deal with large Web service data sets [12]. Until now, many researchers have investigated approaches to extracting or learning ontologies from existing Web service descriptions automatically. For example, Sabou *et al.* [28] presented a framework for ontology learning from Web service descriptions, and the framework used two kinds of linguistic knowledge, including Part-of-Speech information and word dependency relations; Lee *et al.* [29] attempted to generate ontologies from WADL (Web Application Description Language) by grouping parameter names of Web services into semantic concepts and capturing relations between these concepts; and Segev *et al.* [30] described an ontology bootstrapping process for Web services, which exploited both WSDL descriptions and textual descriptions. Despite these efforts have been conducted in this direction, it is still hard to construct appropriate domain ontologies that can provide sufficient semantics (in particular semantic relations and restrictions among domain concepts) for Web services.

B. Machine Learning-Based Service Discovery

Machine learning techniques such as classification and clustering algorithms have been widely applied in Web service discovery. These works can also be viewed as implicit semantics-based approaches. For example, Elgazzar *et al.* [22] extracted features (including the service name, content, types, messages, and ports) from WSDL documents, and then clustered those Web services that share similar features into various groups; Liu *et al.* [13] presented an active learning framework, which combined LDA-based topic models with a SVM (Support Vector Machine) classifier, to classify large-scale services; Chen *et al.* [14, 21] proposed a Web service clustering approach by combining WSDL documents with service tags through augmented LDA; Zhang *et al.* [37] utilized a network routing mechanism to facilitate service discovery, where services annotated by OWL-S were organized into a network based on semantic clustering; Aznag *et al.* [15] extracted topics from Web service descriptions and organized hierarchical clusters to search Web services based on correlated topic models; and Cassar *et al.* [31] learned latent factors from the corpus of Web service descriptions using Probabilistic Latent Semantic Analysis (PLSA) and LDA, and then grouped available Web services according to their latent factors.

C. Difference between Our Work and Related Work

Many other efforts have also been made to improve the performance of Web service discovery. For example, Wang *et al.* [23, 38] proposed a Web service discovery approach by extracting domain-specific service goals from textual service descriptions to match with users' intentional requests; Bi-partite graphs were also utilized to compute the similarity between semantic Web services [20] in the process of Web service discovery; Chen *et al.* [35] proposed a framework to construct a global social service network based on complex network theories, as well as an approach to improving the quality of service discovery based on the global social service network; and Pantazoglou *et al.* [1] proposed a fuzzy-based query evaluation mechanism in the service matchmaking process where several existing similarity metrics were combined and employed.

Inspired by our previous work on context-aware role mining [5, 23], in this paper we introduce the concept of common topic group for Web service organization, and propose a new method for service matchmaking using the extracted common topic groups, which aims at decreasing the number of candidate Web services while keeping the high quality of Web service discovery. Generally speaking, our work belongs to the machine learning-based approaches, which extends existing topic models by grouping together all the Web services that share similar distribution probabilities over multiple topics. Several context-aware role mining approaches [5, 18, 19] have been proposed in recent years. Compared with these work, this paper improves the mining process based on our previous work [5], which can reduce the time of mining common topic groups.

This paper is a great extension of our previous work done in [39]. More specifically, the Web service matching and ranking part is revised by introducing the structure of inverted indexes of the extracted common topic groups, which can improve the efficiency of CTG matchmaking. A new strategy of ranking common topic groups is also proposed in this paper. Moreover, the evaluation part has been significantly extended with additional experiments on two publicly-available data sets.

III. Problem Definition And Solution Overview

First, the definition of common topic group mining is presented. Second, an overview of the proposed solution to Web service discovery based on the extraction of common topic groups is outlined.

A. Problem Definition

Common topic groups are mined from service-topic distribution matrices, which are the output of topic modeling for Web service descriptions. To extract common topic groups, the probabilities in the matrices should be mapped into different grades in advance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

The preliminary definitions related to common topic groups are introduced firstly.

Definition 1 (*Tile*). A *tile* is a sub-matrix of the service-topic distribution matrix, where all cells are non-zero and the cells in the same column have the same probability grade. The area of a tile is defined as the number of cells within the tile.

Definition 2 (*Common Topic Group*). A *common topic group* (*ctg*) is a tile that has at least MINS (≥ 2) rows and MINT (≥ 2) columns, where the values of MINS and MINT can be set manually. Clearly the area of a ctg is no less than MINS × MINT.

Definition 3 (*Common Topic Group Mining*). Given a service- topic distribution matrix $STM = <S, T, G, STG>$, where S is a set of Web services, T is a set of topics, G is a set of probability grades, and $STG \subseteq S \times T \times (G \cup \{0\})$ is a set of service-topic-grade relations, the common topic group mining problem is to find a state $<CTG, SA, TGA>$ that is consistent with STM, ensuring that for any $<CTG', SA', TGA'>$ which is also consistent with STM, $\#CTG \leq \#CTG'$ (CTG and CTG' are two sets of common topic groups), $SA, SA' \subseteq S \times CTG$ (SA and SA' are two service-ctg assignment matrices), and $TGA, TGA' \subseteq CTG \times T \times (G \cup \{0\})$ (TGA and TGA' are two ctg-topic assignment matrices). Note that a state is consistent with STM, if every Web service in S has the same topic-grade assignment as in STM.

	t_1	t_2	t_3	t_4	t_5
s_1	g_1	g_2	g_3	0	0
s_2	0	g_1	0	g_3	g_1
s_3	g_1	g_2	g_3	g_3	g_1
s_4	g_3	g_1	0	g_2	0
s_5	g_3	g_1	0	g_3	g_1

(a) service-topic matrix

	t_1	t_2	t_3	t_4	t_5
s_1	g_1	g_2	g_3	0	0
s_2	0	g_1	0	g_3	g_1
s_3	g_1	g_2	g_3	g_3	g_1
s_4	g_3	g_1	0	g_2	0
s_5	g_3	g_1	0	g_3	g_1

(b) extracted ctgs

	t_1	t_2	t_3	t_4	t_5
ctg_1	g_1	g_2	g_3	0	0
ctg_2	0	0	0	g_3	g_1
ctg_3	g_3	g_1	0	0	0

(c) ctg-topic matrix

	ctg_1	ctg_2	ctg_3
s_1	1	0	0
s_2	0	1	0
s_3	1	1	0
s_4	0	0	1
s_5	0	1	1

(d) service-ctg matrix

Fig. 2. CTG mining: An example.

Example: Let $S = \{s_1, s_2, s_3, s_4, s_5\}$ be a set of Web services, $T = \{t_1, t_2, t_3, t_4, t_5\}$ be a set of topics, $G = \{g_1, g_2, g_3\}$ be a set of probability grades, and MINS = MINT = 2. The original service-topic distribution matrix M is shown in Fig. 2(a), where value g_* ("*" denotes 1, 2, or 3) in cell $\{i, j\}$ represents the probability grade of Web service s_i under topic t_j. Fig. 2(b) displays three tiles extracted from M, as shown in the shaded regions with dashed lines. It is impossible to find a tiling that covers M with less than three tiles; therefore, these three tiles are viewed as three ctgs. Finally, a state $<CTG, SA, TGA>$ can be obtained after CTG mining, where $CTG = \{ctg_1, ctg_2, ctg_3\}$, $ctg_1 = \{<t_1, g_1>, <t_2, g_2>, <t_3, g_3>\}$, $ctg_2 = \{<t_4, g_3>, <t_5, g_1>\}$, $ctg_3 = \{<t_1, g_3>, <t_2, g_1>\}$, SA and TGA are shown in Fig. 2(c) and Fig. 2(d), respectively.

B. Overview of Our Approach

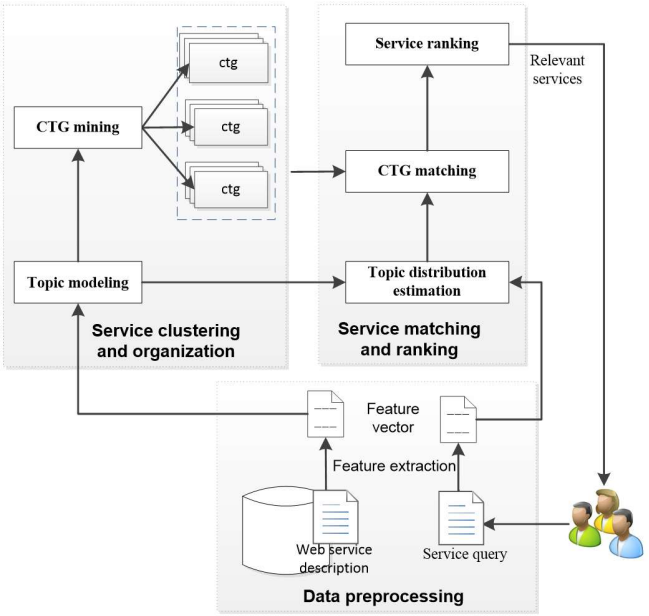


Fig. 3. Overview of our Web service discovery approach.

As shown in Fig. 3, the approach of CTG-based Web service discovery consists of three parts: data preprocessing, service clustering and organization, and service matching and ranking.

In the first part, data preprocessing is used to extract meaningful words from Web service description documents as feature words. The extracted feature words are used to construct a vector space, which serves as the input of topic modeling. More details please refer to Section VI.B.

In the second part, we take the following two steps to cluster and organize Web services available in a service repository. In the first step, existing topic modeling technologies are leveraged to discover latent topics from the vector space, and common topic groups are then extracted from the latent topics. This is one of the main contributions of this paper. LDA [7] has been viewed as the most widely used topic model, and many extensions of LDA that focus on dealing with short texts have also been proposed. For instance, the Correlated Topic Model (CTM) [16] extends LDA in modeling relations between topics by replacing the Dirichlet distribution with the logistic normal distribution; the Biterm Topic Model (BTM) [17], another latest progress in extending LDA for dealing with the short message problem, models the generation of co-occurrence patterns in the whole corpus. Because Web service descriptions after preprocessing are usually short texts, we need to test these three topic models to find the best one for the experimental data sets we used.

To match with the input of CTG mining, the probabilities in the service-topic distribution matrix generated by topic modeling should be mapped into different grades firstly. Intuitively, the basic mapping is to map the distribution probabilities into two grades. That is to say, the probabilities that exceed a certain threshold are viewed as "1", otherwise

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

"0". In this way, a service-topic distribution matrix is transformed into a binary matrix. Furthermore, to extract more accurate common topic groups at a finer granularity, the values belonging to the "1" grade can be further classified into multiple sub-grades, which will generate a multiple-valued matrix. Our CTG mining approach takes the two types of matrices as input, based on which ctgs at different granularities can be obtained. An inverted index is then used to store the extracted ctgs and the topics of Web services, aiming at reducing the user query time. More details of CTG mining are presented in Section IV.

In the third part, when a user query is submitted to a Web service search engine, the query will be preprocessed using the same steps as for service descriptions first. The topic distribution of the query is estimated, and the extracted common topic groups are then leveraged to retrieve relevant Web services for the user query. More details please refer to Section V.

IV. CTG MINING

In this section, we apply the context-aware role mining method proposed in [5] to common topic groups mining, and improve its core algorithm to enhance the efficiency in mining ctgs. In [5], we adopted the idea of tiling databases [9] in context-aware role mining. Following the principle of context-aware role mining, the CTG mining problem is reduced to the minimum tiling problem, which aims at finding a tiling of which the area equals to the total number of non-zero cells in the matrix and consists of the minimum number of tiles.

Next, we present the details of CTG mining algorithms. The overall procedure of CTGs mining is shown in Algorithm 1. Given an input service-topic matrix M_{st}, Line 1 calls Algorithm 2 to find a minimum tile set. According to the tiles mined from M_{st}, Lines 3-9 output new ctgs, the ctg-topic matrix M_{ct}, and the service-ctg matrix M_{sc}.

Algorithm 1 Minimum Common Topic Groups (CTGs) Mining

```
Input:  service-topic matrix M_st
Output: ctg-topic matrix M_ct and service-ctg matrix M_sc
1  Tiles = ExtractTileSet(M_st);
2  for each tile tile∈Tiles do
3      create a new common topic group ctg;
4      extract topic set T and grade set D from tile;
5      for each topic t∈T and each grade d∈D do
6          add assignment {ctg, t, d} to M_ct;
7      extract Web service set S from tile;
8      for each Web service s∈S do
9          add assignment {s, ctg} to M_sc;
10 return M_ct and M_sc;
```

To extract tiles from M_{st}, first of all, we group together those Web services that share the same probability grades over each topic by transforming the service-topic matrix M_{st} into a grade-topic matrix M_{gt} (see Line 2 in Algorithm 2). The transformation result of the matrix in Fig. 2(a) is shown in Table I. For each cell in the grade-topic matrix, its coordinate (<grade, topic>) and Web services associated with the cell are recorded. For example, for cell c_{11} in Table I, c_{11}.grade = $\{g_1\}$, c_{11}.topic = $\{t_1\}$, and c_{11}.services = $\{s_1, s_3\}$. Note that because the number of Web services is usually far larger than that of probability grades, the size of the matrix to be analyzed can be greatly reduced by the matrix transformation.

TABLE I
NEW GRADE-TOPIC MATRIX TRANSFORMED FROM THE MATRIX IN Fig. 2(a)

	t_1	t_2	t_3	t_4	t_5
g_1	$\{s_1, s_3\}$	$\{s_4, s_5\}$	/	/	$\{s_2, s_3, s_5\}$
g_2	/	$\{s_1, s_3\}$	/	$\{s_4\}$	/
g_3	$\{s_4, s_5\}$	/	$\{s_1, s_3\}$	$\{s_2, s_3, s_5\}$	/

Recall that, in Subsection III.A, MINS (≥ 2) and MINT (≥ 2) denote the minimal numbers of Web services and topics, respectively, which are used to form a ctg. Because it is impossible for a cell in the grade-topic matrix which contains less than MINS Web services to constitute a ctg, only the cells that contain more than MINS Web services (see Line 3 in Algorithm 2) are extracted and stored in array *Cell* (*nc* denotes the size of *Cell*). Next, we merge the elements in *Cell* to generate tiles.

For each element in *Cell*, we compare it with its subsequent cells to find qualified tiles (see Lines 5-19 in Algorithm 2). Here a qualified tile represents a candidate ctg, which indicates that it contains at least MINS Web services and MINT topics. We use arrays *commonServiceSet* and *cTileSet* to store the intersection of Web services contained in any two cells and the candidate tile set generated from each cell, respectively. Because each Web service has exactly one probability grade under each topic, we don't need to compare the cells with the same topic (see Line 9 in Algorithm 2). For any two cells, only if the size of the intersection between their related Web services exceeds MINS, can they be considered to constitute a tile.

Algorithm 2 ExtractTileSet

```
Input:  service-topic matrix M_st
Output: set of extracted tiles TileSet
1  TileSet = {};
2  Transform M_st to a grade-topic matrix M_gt ;
3  Cell ← {c_ij ∈ M_gt | #c_ij.services ≥ MINS};
4  nc =#Cell;
5  for i = 1 to nc do
6      commonServiceSet = {};
7      cTileSet = {};
8      for j = i+1 to nc do
9          if Cell[i].topic ≠ Cell[j].topic then
10             commonServiceSet = Cell[i].services ∩ Cell[j].services;
11             tile = {};
12             tile.services = {};
13             if commonServiceSet.size() ≥ MINS then
14                 tile = tile ∪ <Cell[j].grade, Cell[j].topic>;
15                 tile.services = tile.services ∪ commonServiceSet;
16                 CompareAndUpdate(tile, cTileSet);
17     for each tile t in cTileSet do
18         t = t ∪ <Cell[i].grade, Cell[i].topic>;
19     TileSet= TileSet ∪ cTileSet;
20 RemoveDuplication(TileSet);
21 return TileSet;
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

A new tile constituted by any two cells in array *Cell* should be compared with existing tile set *cTileSet* to check whether *cTileSet* will be updated (see Algorithm 3). Then, the tile sets created from each cell will be merged as a whole. In the merged tile set, a few tiles may be duplicated, and some tiles are probably a subset of others. In the final step, we remove the duplicated tiles and the tiles that are the subset of others (see Line 20 in Algorithm 2).

In Algorithm 3, we update a candidate tile set *cTileSet* by comparing it with a given tile *tile*. If *cTileSet* is empty, *tile* will be directly added into *cTileSet*. Otherwise, for each tile *t* in *cTileSet*, we compare it with *tile* by considering the following cases. First, if the Web service sets contained in *t* and *tile* are identical, *t* will be updated by adding the cells in *tile* (see Lines 6-7 in Algorithm 3). Note that the variable *equalToTile* is used to record whether the Web service set included in *tile* is identical to that of any tile in *cTileset*. If so, variable *equalToTile* is assigned to 1. Second, if the Web service set contained in *t* is a proper subset of that of *tile*, *t* will be updated by adding the cells in *tile* (see Lines 9-10 in Algorithm 3). Third, contrary to the second case, *tile* will be updated by adding the cells in *t* (see Lines 11-12 in Algorithm 3). In the last case, we judge whether the size of the set of common Web services between *t* and *tile* exceeds MINS. If the size is no less than MINS, i.e., the intersection of *t* and *tile* satisfies the minimum requirements of ctg, a new tile will be created and added to *cTileSet* (see Lines 13-18 in Algorithm 3). Finally, if *equalToTile* equals to 0, i.e., there is no tile which is identical to *tile* in *cTileset*, *tile* will be added to *cTileSet*.

Algorithm 3 CompareAndUpdateTile

Input: tile *tile* and candidate tile set *cTileSet*
Output: candidate tile set *cTileSet*

```
1  if cTileSet.isEmpty() then
2  |  cTileSet = cTileSet ∪ {tile};
3  else
4  |  equalToTile = 0;
5  |  for each tile t in cTileSet do
6  |  |  if t.services = tile.services then
7  |  |  |  t = t ∪ tile;
8  |  |  |  equalToTile = 1;
9  |  |  else if t.services ⊂ tile.services then
10 |  |  |  t = t ∪ tile;
11 |  |  else if tile.services ⊂ t.services then
12 |  |  |  tile = t ∪ tile;
13 |  |  else
14 |  |  |  commonServiceSet = tile.services ∩ t.services;
15 |  |  |  if commonServiceSet.size() ≥ MINS then
16 |  |  |  |  newTile= tile ∪ t;
17 |  |  |  |  newTile.services = commonServiceSet;
18 |  |  |  |  cTileSet= cTileSet ∪ {newTile};
19 |  if equalToTile = 0 then
20 |  |  cTileSet = cTileSet ∪ {tile};
21 return cTileSet;
```

As mentioned in Subsection III.B, for each given service-topic distribution matrix, we can transform it into a binary matrix, where the probabilities are mapped to "0" and "1", and into a multiple-valued matrix, where the values belonging to the "1" grade are classified into multiple sub-grades. Each ctg extracted from a multiple-valued matrix, which has a finer granularity, is covered by the corresponding ctg extracted from a binary matrix. It is worth noting that the above-mentioned algorithms are intended to extract common topic groups from both the two types of matrices.

TABLE II
INVERTED INDEX OF A FEW STORED CTGS: AN EXAMPLE

Topic	CTGs
t_1	ctg_1, ctg_3, ctg_4, ctg_5, ctg_6, …
t_2	ctg_1, ctg_6, ctg_7, …
t_3	ctg_2, ctg_3, ctg_4, …
t_4	ctg_1, ctg_4, …
t_5	ctg_2, ctg_3, ctg_4, ctg_5, ctg_6, ctg_7, …

TABLE III
TOPIC DISTRIBUTIONS OF A USER QUERY AND A FEW STORED CTGS

	t_1	t_2	t_3	t_4	t_5
q_1	1	0	3	0	2
ctg_1	0	1	0	2	0
ctg_2	1	0	3	0	2
ctg_3	1	0	2	0	1
ctg_4	1	0	3	1	2
ctg_5	1	0	0	0	2
ctg_6	1	1	0	0	2
ctg_7	0	2	0	0	2
...	...	...	...	...	...

V. WEB SERVICE MATCHING AND RANKING BASED ON CTG

Once a user query is submitted to a service search engine built using our approach, the query will be preprocessed using the steps of data preprocessing mentioned in Subsection III.B. As a result, such a service query *sq* will be represented as a feature vector. Next, the topic distribution of the query is estimated based on a topic model generated by existing Web service descriptions. By using the Gibbs sampling approach [6], the distribution probability of *sq* under topic t_i is computed as follows.

$$P(sq|t_i) = \frac{n_i^{sq}+\alpha}{\sum_{j=1}^{T}(n_j^{sq}+\alpha)}, \quad (1)$$

where $n_i{}^{sq}$ is the number of words in *sq* assigned to topic t_i, which can be calculated based on the topic-word matrix, *T* is the number of topics, $\sum_{j=1}^{T} n_j^{sq}$ is the total number of words in *sq*, and α is the document-topic Dirichlet parameter.

After the topic distribution of a query is generated, we map the topic distribution probabilities into various grades as those of Web services using the method described in Subsection III.B. Based on this mapping, the query can be matched with the extracted ctgs to find the right candidate Web services. Recall that each ctg consists of more than MINT (≥ 2) pairs of topics and grades (i.e. $\{<t_i, g_j>\}$), and it is associated to at least MINS (≥ 2) Web services. To improve the query speed of Web services, an inverted index is leveraged to store all the extracted

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

ctgs. That is, the data structure of such an index is created to store a mapping from the content of ctgs to the ctgs. Because the topic distribution of a query is also represented in the form of pairs of topics and grades, the inverted index can be utilized in the CTG matchmaking between the query and existing ctgs. Table II shows an example of the index of a few stored ctgs.

The CTG matchmaking process aims to find the ctgs whose topic distributions are similar to that of a given user query. Table III describes the topic distributions of a user query q_1 and a few stored ctgs. Then, we use the examples provided in Table III to introduce the CTG matchmaking process.

According to the topic distribution of the user query, we identify the topics whose distribution grades are not equal to 0. For each of these topics, the corresponding ctgs can be selected from the inverted index, as shown in Table II. In this way, a number of irrelevant ctgs are filtered out, which can improve the efficiency of matchmaking. For example, ctg_1 in Table III will not be considered for query q_1.

Algorithm 4 Matching and Ranking Web Services

Input: user query *sq* and ctg-topic matrix M_{ct}
Output: ranked list of Web services *sList*

```
TOPICS←topics of M_ct, CTGS←{}, SERVICES←{};
for each topic t_i in TOPICS do
    calculate the topic distribution P(sq| t_i) of sq using Eq. (1);
map the topic distribution probabilities into various grades;
for each topic t_i in TOPICS do
    if sq.t_i≠0 then
        CTGS←ctgs related to t_i in M_ct;
for each ctg ctg_i in CTGS do
    determine the rank of ctg_i relevant to sq;
for each rank rl_i (i∈[1,4]) do
    for each ctg ctg_j in rl_i do
        SERVICES←ctg_j.services;
    if #SERVICES >= k then
        break;
for each Web service service_i in SERVICES do
    calculate the similarity between service_i and sq using Eq. (2);
sList←Top k Web services in SERVICES;
return sList;
```

The selected ctgs will then be ranked by comparing their topic distributions with that of the query. We define the following four ranks.

- rl_1: If the topic distribution of a ctg matches exactly with that of the query, the ctg belongs to rl_1. For instance, ctg_2 in Table III is an example of this level. Note that here we refer to the multiple-valued distribution grades. Compared with ctg_2, ctg_3 has the same non-zero topic distribution except for two different distribution grades for t_3 and t_5. Therefore, ctg_3 can also be viewed as an instance of this level, but it is ranked lower than ctg_2.
- rl_2: If the topic distribution of a ctg covers that of the query, the ctg belongs to rl_2. For instance, ctg_4 in Table III is an example of this level.
- rl_3: If the topic distribution of a ctg is a subset of that of the query, the ctg belongs to rl_3. For instance, ctg_5 in Table III is an example of this level.
- rl_4: If the topic distribution of a ctg intersects with that of the query, the ctg belongs to rl_4. Note that the ctgs belonging to rl_4 will be ordered by the number of overlapping topics between each of them and the query. For instance, ctg_6 and ctg_7 in Table III are two examples of this level, and ctg_6 has a higher priority than ctg_7.

After these ctgs are ranked, Web services associated to them will be returned in a predefined order based on the above-mentioned four ranks. Note that not all the ordered ctgs should be considered for a specific Top-*k* service query. If the number of candidate Web services reaches *k*, the CTG matchmaking process will be completed.

Web services belonging to the first two ranks can match with the query very well and thus are ranked as preferred candidate Web services. It is unnecessary to further compute their similarities with the query. For those Web services selected according to the other two ranks, a classical LDA-based information retrieval approach [8] is leveraged to rank them. In (2), each Web service s_i is scored by the likelihood of its model generating query *sq*.

$$P(sq|s_i) = \prod_{w_j \in sq} P(w_j|s_i), \tag{2}$$

$$P(w_j|s_i) = \sum_{k=1}^{T} P(w_j|t_k) P(t_k|s_i), \tag{3}$$

where $P(w_j|t_k)$ and $P(t_k|s_i)$ are obtained from the topic-word distribution matrix and the service-topic distribution matrix, respectively. In this way, candidate Web services are ranked for each query, and the top ranked Web services will be returned to users.

The whole process of Web service matching and ranking based on CTG is presented in algorithm 4.

VI. Experiments And Result Analysis

In this section, we evaluate the performance of the proposed approach on two open-access data sets published on the Internet. All the algorithms were developed in Java, and our experiments were conducted on a PC with 2 GHz Intel Core T7300 CPU and 4GB RAM, running Windows 7 OS.

A. Data Set Description

Two publicly-available data sets were employed in our experiments. The first data set we used is a WSDL service retrieval test collection: SAWSDL-TC[5], which consists of 1,080 WSDL Web services belonging to nine domains, as well as 42 queries represented in WSDL documents. A set of graded relevance for each query is preconfigured in the data set, which can be viewed as the benchmark of relevant Web services. Three relevance levels were defined in the dataset: "1" denotes that a Web service is potentially relevant to a given query, "2" for a relevant relation, and "3" for high relevance.

The second data set we utilized is an OWL-S service retrieval test collection: OWLS-TC[6], which contains 1,083

[5] http://www.semwebcentral.org/projects/sawsdl-tc
[6] http://semwebcentral.org/projects/owls-tc/

OWL-S Web services belonging to nine domains, as well as 42 queries represented in OWL-S profile documents. Similarly, these queries are also equipped with the sets of preconfigured relevance levels that have the similar meaning as those of WSDL queries.

B. Data Preprocessing

For each data set, we performed the following steps to preprocess Web service descriptions.

- **Tokenization and stop words removal.** The NLTK[7] toolkit is leveraged to obtain the original word vectors from Web service descriptions. Function words, e.g. *a* and *the*, and WSDL specific words, e.g. *request* and *response*, will be removed**.** The built-in stop word list in NLTK and the inverse document frequency are utilized to remove these types of words.
- **Stemming and lemmatization.** Each remaining word should be reduced to its root form. Both lemmatization and stemming can achieve this purpose, but they also have their respective advantages and disadvantages. According to our experiments on some selected Web service descriptions, we find that there are slight differences between stemming and lemmatization in dealing with nouns and verbs. For example, the verb "driven" is reduced to "driven" by using a stemmer, while to "drive" in a lemmatizer; the noun "medias" is reduced to "media" by the stemmer, while to "medias" by the lemmatizer. Hence, in the data preprocessing part, we will judge whether a word is a noun or a verb using WordNet[8], and then utilize the Stemmer and the Lemmatizer in WordNet to deal with nouns and verbs, respectively.

Referring to the WSDL documents, we extracted some typical feature words that describe Web services, such as *service name*, *port type*, *message*, *documentation*, and *operation*. Note that the *types* element defined in WSDL, which consists of data type definitions that are relevant for exchanged messages, is also an important element to provide semantics of Web services. However, in the Web services of SAWSDL-TC, the definitions of *types* are usually much longer than those of other elements and many Web services share the same *types* definitions, which makes it difficult to differentiate among these Web services using topic models.

For example, two Web services *academic-degreegovernmentorganization-funding_service* and *CAD_ medical_service* share the same *types* definition, which still contains 979 terms after data preprocessing. Except the *types* definition, their service descriptions have only 22 and 30 words, respectively. Although the two Web services belong to different domains: Education and Medical, they are very likely to be grouped to the same cluster if their *types* definitions are considered in service clustering. Therefore, the elements tagged by *types* were not extracted in our experiments.

Referring to the OWL-S documents, we extracted the elements labeled with tags including *profile:serviceName*, *profile:hasInput*, *profile:hasOutput*, and *profile:textDescription*. Since most of OWL-S Web services in OWLS-TC contain only an atomic process, the elements marked with the *profile*:*has_process* tag were not extracted in our experiments.

[7] http://www.nltk.org/
[8] http://wordnet.princeton.edu/

C. Selection of Topic Models

1) Basic Topic Models

Considering that our proposed CTG-based Web service discovery approach heavily depends on topic models, we evaluated the performance of three Web service discovery approaches based on typical topic models, including LDA, CTM, and BTM, to test which one could serve as the basis of our approach towards the two data sets. Note that the procedure of Web service discovery based on these topic models is similar to our proposed approach in Section V except that the CTG matchmaking process is not involved in the process. That is to say, these approaches use Equations (2) and (3) to rank all the candidate Web services for a given service query.

2) Evaluation Metrics

We used the following metrics to evaluate the performance of Web service discovery based on these topic models.

Precision* at *k: *Precision* at *k* is one of the most intuitive metrics for evaluating the performance of search engines. For each user query in the test set, this metric represents the fraction of the first *k* retrieved Web services that also appear in the set of relevant Web services of the query. Note that each query has three types of relevant Web services. Here, all the three kinds of relevant Web services are considered in our evaluation. Because the number (*nr*) of relevant Web services associated to several queries is less than *k*, we replaced *k* with *nr* in calculating the values of this metric under these cases.

NDCG at *k*: The quality of Web service discovery is also sensitive to the positions of the top *k* retrieved Web services, and highly-relevant Web services are clearly expected to appear earlier in an ordered list of retrieved Web services. We used normalized discounted cumulative gain (NDCG), a widely used evaluation metric in Web search, to measure the ranking quality of retrieved Web services.

$$NDCG_k = \frac{DCG_k}{IDCG_k}, \tag{4}$$

where DCG_k, the discounted cumulative gain (DCG) of the top *k* retrieved Web services, is calculated using Equation (5) [32], and $IDCG_k$, the ideal DCG at *k*, is the maximum possible DCG till position *k* by sorting retrieved Web services in terms of relevance.

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i - 1}}{\log_2(i+1)}, \tag{5}$$

where rel_i is the graded relevance of the Web service at position *i* in an ordered list of retrieved Web services.

3) Determining the Number of Topics

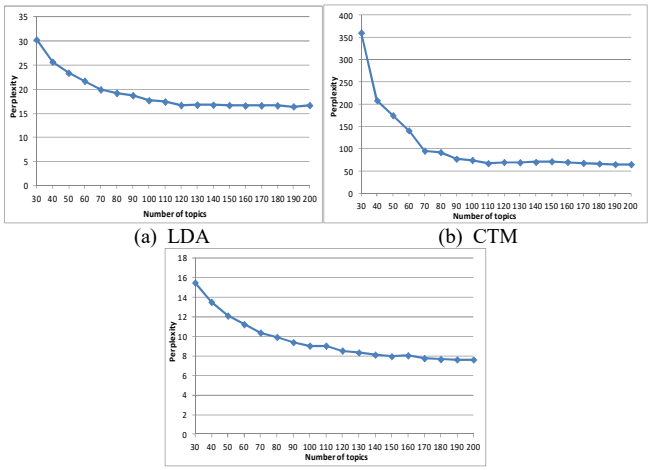
The number of topics is an important factor to influence the quality of Web service clustering. Several approaches [7, 27] have been proposed to choose the number of topics that can result in better generalization performance. According to the definition of the perplexity of a held-out test set defined in [7],

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

in this paper we adopt the perplexity to select the optimal topic number. The lower the perplexity becomes, the better the generalization performance will be. For a test set of M Web services, its perplexity is defined as below.

$$perplexity(D_{test}) = \exp\{-\frac{\sum_{i=1}^{M}\sum_{j=1}^{N_i}\log P(w_j|s_i)}{\sum_{i=1}^{M}N_i}\}, \quad (6)$$

where N_i is the total number of word tokens contained in Web service s_i and $P(w_j|s_i)$ can be calculated using Equation (3).

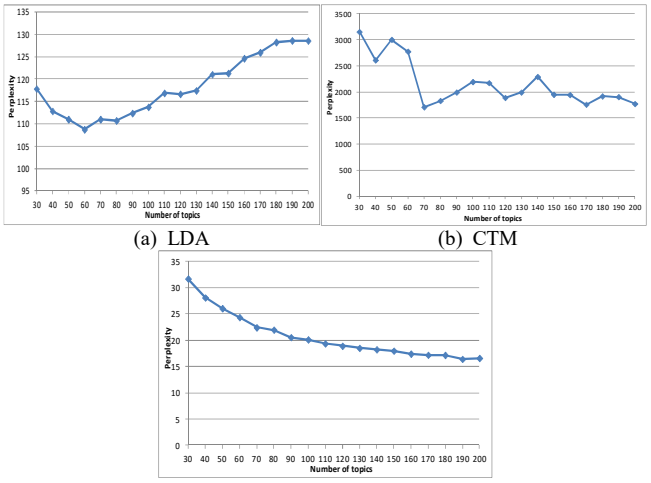


(a) LDA (b) CTM

(c) BTM

Fig. 4. Perplexity results of the three topic models on SAWSDL-TC.

Given a data set, the optimal number of topics for LDA, CTM and BTM might be different, and the best generalization performance does not necessarily lead to the best performance of Web service discovery. To compare these topic models comprehensively, in this paper we estimated the optimal interval number of topics for the three topic models. As shown in Fig. 4, for the SAWSDL-TC data set, LDA can achieve the ideal generalization performance when the number of topics ranges between 120 and 190, CTM can achieve the ideal generalization performance when the number of topics ranges in values from 110 to 190, and BTM can achieve the ideal generalization performance when the number of topics is around 190. The interval numbers of topics used for comparisons are therefore set to [110, 130] and [180, 200].



(a) LDA (b) CTM

(c) BTM

Fig. 5. Perplexity results of the three topic models on OWLS-TC.

Similarly, Fig. 5 shows the results of perplexity for the three topic models on the OWLS-TC data set. LDA, CTM, and BTM can achieve the ideal generalization performance when the numbers of topics are around 60, 70, and 190, respectively. We therefore set the interval numbers of topics used for comparisons to [50, 70] and [180, 200].

4) Results

Generally speaking, users are usually interested in the top-ranked candidate Web services. Therefore, up to 20 top-ranked Web services among all the retrieved results were taken into account in our evaluation. Tables IV and V show the results of *Precision* at k and NDCG at k using LDA, CTM, and BTM on the data sets SAWSDL-TC and OWLS-TC, respectively. The value of each cell in the two tables means the average value of the corresponding measure over all the 42 queries.

TABLE IV
COMPARISON OF THE THREE TOPIC MODELS ON THE SAWSDL-TC DATA SET

Number of topics	Top k Services	LDA		CTM		BTM	
		NDCG	Precision	NDCG	Precision	NDCG	Precision
110	k=5	0.6491	0.8357	0.6331	0.8341	0.5741	0.8353
	k=10	0.6633	0.8119	0.6294	0.7968	0.5928	0.8028
	k=15	0.6671	0.7921	0.6426	0.7810	0.6208	0.7980
	k=20	0.6839	0.7827	0.6591	0.7712	0.6461	0.7916
	Average	0.6658	0.8056	0.6411	0.7958	0.6085	0.8069
120	k=5	0.6603	0.8540	0.5691	0.8056	0.5940	0.7893
	k=10	0.6754	0.8333	0.5954	0.7952	0.6267	0.7988
	k=15	0.6765	0.8008	0.6062	0.7738	0.6499	0.7861
	k=20	0.6988	0.7963	0.6290	0.7647	0.6739	0.7896
	Average	**0.6778**	**0.8211**	0.5999	0.7848	0.6361	0.7910
130	k=5	0.6472	0.8373	0.6131	0.8294	0.6088	0.8052
	k=10	0.6592	0.8429	0.6001	0.7683	0.6297	0.8123
	k=15	0.6783	0.8222	0.6229	0.7675	0.6504	0.7956
	k=20	0.6866	0.8023	0.6527	0.7683	0.6718	0.7934
	Average	0.6678	0.8262	0.6222	0.7833	0.6402	0.8016
180	k=5	0.6726	0.8448	0.5866	0.8278	0.6332	0.8183
	k=10	0.6726	0.8162	0.5992	0.7976	0.6455	0.8123
	k=15	0.6768	0.8003	0.6166	0.7810	0.6710	0.8060
	k=20	0.6821	0.7916	0.6380	0.7593	0.6814	0.7815
	Average	0.6760	0.8132	0.6101	0.7914	0.6578	0.8045
190	k=5	0.6700	0.8586	0.5634	0.8278	0.6068	0.7825
	k=10	0.6658	0.8114	0.6062	0.8190	0.6195	0.7921
	k=15	0.6809	0.8059	0.6089	0.7857	0.6521	0.7960
	k=20	0.6894	0.7916	0.6223	0.7553	0.6709	0.7850
	Average	0.6765	0.8169	0.6002	0.7970	0.6373	0.7889
200	k=5	0.6737	0.8405	0.5805	0.8230	0.6177	0.8421
	k=10	0.6754	0.8190	0.6015	0.8214	0.6296	0.8171
	k=15	0.6803	0.8040	0.6103	0.7762	0.6548	0.8044
	k=20	0.6814	0.7912	0.6315	0.7562	0.6713	0.7937
	Average	0.6777	0.8137	0.6059	0.7942	0.6434	0.8143

As shown in Table IV, these three approaches can get similar *Precision* and NDCG results on the SAWSDL-TC data set. In most cases, the *Precision* values are around 80%, and the NDCG values are around 60%. Overall, LDA performs better than BTM and CTM. Considering both aspects of *Precision* and NDCG, LDA can achieve the best results when the number of topics is equal to 120.

As for the OWLS-TC data set, according to Table V, the results of LDA and BTM are close, while the results of CTM are lower than those of these two methods. In most cases, the *Precision* values of LDA and BTM are also around 80%, and the NDCG values are around 60% as well. Considering both

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

aspects of *Precision* and NDCG, BTM can achieve the best results when the number of topics is equal to 180.

TABLE V
COMPARISON OF THE THREE TOPIC MODELS ON THE OWLS-TC DATA SET

Number of topics	Top *k* Services	LDA		CTM		BTM	
		NDCG	Precision	NDCG	Precision	NDCG	Precision
50	*k*=5	0.5438	0.7520	0.3939	0.6091	0.4610	0.7349
	k=10	0.5766	0.7556	0.4046	0.5837	0.5031	0.7468
	k=15	0.5898	0.7476	0.4236	0.5881	0.5371	0.7405
	k=20	0.6103	0.7360	0.4533	0.5938	0.5697	0.7560
	Average	0.5801	0.7478	0.4189	0.5937	0.5178	0.7446
60	*k*=5	0.5823	0.7925	0.4136	0.5817	0.4579	0.7714
	k=10	0.6139	0.8183	0.4211	0.5837	0.5210	0.7667
	k=15	0.6300	0.7897	0.4401	0.5786	0.5565	0.7651
	k=20	0.6519	0.7844	0.4683	0.5914	0.5916	0.7705
	Average	0.6195	0.7962	0.4358	0.5839	0.5317	0.7684
70	*k*=5	0.6104	0.8222	0.4248	0.6167	0.5461	0.8063
	k=10	0.6462	0.8246	0.4516	0.6552	0.5920	0.8087
	k=15	0.6614	0.8032	0.4905	0.6722	0.6135	0.8000
	k=20	0.6722	0.7743	0.5171	0.6613	0.6463	0.8013
	Average	0.6475	0.8061	0.4710	0.6513	0.5995	0.8041
180	*k*=5	0.5303	0.7651	0.3695	0.6143	0.6338	0.8587
	k=10	0.5555	0.7532	0.3782	0.5881	0.6699	0.8476
	k=15	0.5759	0.7563	0.4092	0.6071	0.6905	0.8286
	k=20	0.5981	0.7535	0.4352	0.6066	0.7057	0.8105
	Average	0.5649	0.7570	0.3980	0.6040	**0.6750**	**0.8364**
190	*k*=5	0.6069	0.7849	0.3686	0.6238	0.6137	0.8492
	k=10	0.6186	0.7683	0.3813	0.6214	0.6527	0.8381
	k=15	0.6316	0.7524	0.4145	0.6246	0.6833	0.8159
	k=20	0.6578	0.7571	0.4399	0.6225	0.7009	0.8027
	Average	0.6287	0.7657	0.4010	0.6231	0.6627	0.8265
200	*k*=5	0.5848	0.7591	0.3786	0.6095	0.6204	0.8602
	k=10	0.5905	0.7587	0.4013	0.6333	0.6552	0.8512
	k=15	0.6121	0.7484	0.4355	0.6500	0.6687	0.8221
	k=20	0.6375	0.7461	0.4646	0.6567	0.6895	0.8114
	Average	0.6062	0.7531	0.4200	0.6374	0.6585	0.8362

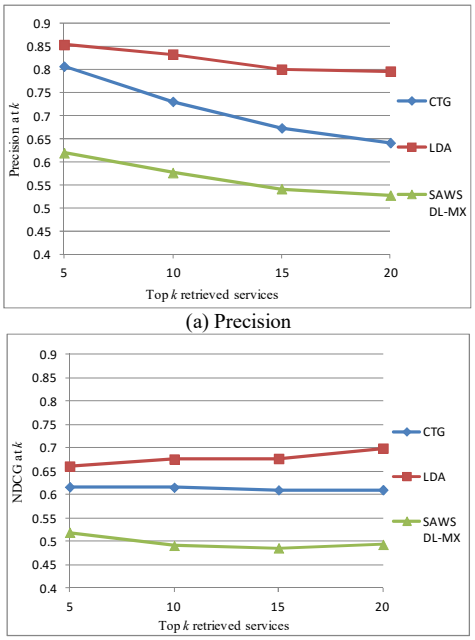
Note that because the number of some queries' relevant Web services is often less than 20, most of the *Precision* values are slightly decreased with the increase of *k*. Due to the same reason as *Precision* and the fact that there are only three kinds of graded relevance, the NDCG values are slightly increased with the increase of *k*.

D. Evaluation of CTG-Based Web Service Discovery

1) Competing Approaches

In Subsection VI.C, the results of our experiment show that LDA and BTM achieve better results on the data sets SAWSDL-TC and OWLS-TC, respectively. The respective results of these two topic models are thereby used as the input of our CTG mining approach for the two data sets, respectively. More specifically, for SAWSDL-TC, CTG mining takes as input the service-topic distribution matrix generated by LDA when the number of topics is set to 120; for OWLS-TC, CTG mining takes as input the service-topic distribution matrix generated by BTM when the number of topics is set to 180.

To demonstrate the effectiveness of our approach called CTG based Web service discovery (CTG for short), we compared CTG with topic model based Web service discovery approaches (e.g. LDA and BTM) as well as two well-known Web service discovery approaches, namely SAWSDL-MX [4] and OWLS-MX [10]. That is to say, for SAWSDL-TC, we compared CTG with LDA and SAWSDL-MX; for OWLS-TC, we compared CTG with BTM and OWLS-MX.



(a) Precision

(b) NDCG

Fig. 6. Comparison among CTG, LDA, and SAWSDL-MX on SAWSDL-TC.

2) Evaluation Metrics

Here we still use the metrics *Precision* and NDCG to evaluate the performance of CTG and other Web service discovery technologies under discussion. Our approach (CTG) differs from those topic model based Web service discovery approaches in that ours can reduce the search space greatly by using CTG matching. In this paper, LDA and BTM compute the similarity between a query and each Web service in the candidate Web service set, while CTG ranks only those Web services selected by CTG matchmaking, which belong to a subset of what LDA (or BTM) ranks. Therefore, the *Precision* and NDCG values of CTG are not likely to be better than those of LDA (or BTM).

It is noteworthy that CTG aims to minimize the number of candidate Web services while keeping the *Precision* and NDCG of Web service discovery at an elevated level. We then evaluated the differences in *Precision* and NDCG between CTG and LDA (or BTM), as well as the decreased ratio of candidate Web services using CTG, also known as compression rate. Compression rate indicates the proportion of candidate Web services obtained based on CTG matchmaking to all the candidate Web services.

$$\text{Compression rate} = \frac{|S_{CTG}|}{|S|}, \tag{7}$$

where S_{CTG} denotes those Web services selected using CTG matchmaking and S denotes all the Web services available in a service repository.

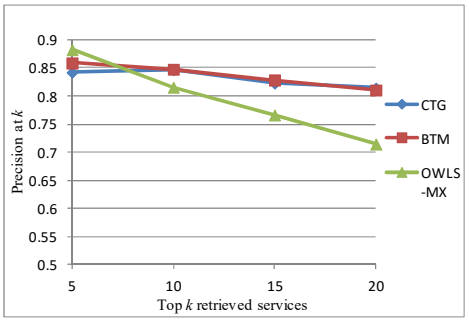
Moreover, we compared CTG with other topic model based Web service discovery methods with respect to query response time.

3) Results

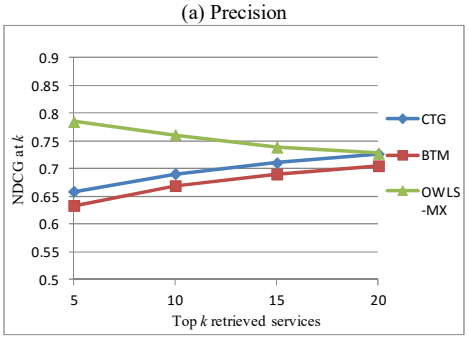
Fig. 6 shows the average *Precision* and NDCG values of CTG, LDA, and SAWSDL-MX over all the queries of SAWSDL-TC, and Fig. 7 shows the average *Precision* and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

NDCG values of CTG, BTM, and OWLS-MX over all the queries of OWLS-TC. In particular, as for SAWSDL-TC, it is clear from Fig. 6 that both CTG and LDA perform better than SAWSDL-MX, and that the performance of CTG is close to that of LDA when k = 5. According to Fig. 7, CTG and BTM perform better than OWLS-MX with respect to *Precision* in most cases, while OWLS-MX performs the best in terms of NDCG. The NDCG values of CTG and BTM increase with the increase of the number of retrieved Web services, and they are close to that of OWLS-MX when k = 20. High NDCG values of OWLS-MX indicate that semantic annotations based on domain ontologies in OWLS-MX can indeed achieve better service discovery results. However, such an ontology-based approach always suffers from slow response time, as shown in Table VIII, let alone the heavy burden on developing and maintaining complex domain ontologies.



(a) Precision



(b) NDCG

Fig. 7. Comparison among CTG, BTM, and OWLS-MX on OWLS-TC.

TABLE VI
COMPARISON BETWEEN CTG AND LDA ON SAWSDL-TC

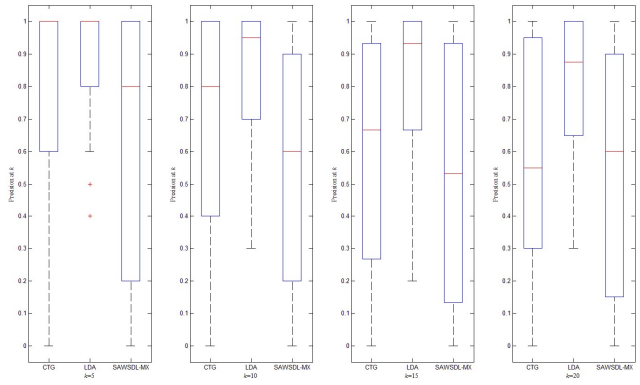
	Top 5	Top 10	Top 15	Top 20	Average
Precision ratio	94.52%	87.71%	84.14%	80.59%	86.74%
NDCG ratio	93.29%	91.08%	90.07%	87.26%	90.42%
Compression rate	7.14%	7.90%	8.85%	9.41%	8.32%

Furthermore, Table VI shows a comparison between CTG and LDA on SAWSDL-TC, and Table VII shows a comparison between CTG and BTM on OWLS-TC. We compared them by using the precision ratio of the *Precision* value of CTG to that of LDA or BTM, the NDCG ratio of the NDCG value of CTG to that of LDA or BTM, as well as the compression rate over all the queries provided in the data sets. According to Tables VI and VII, the *Precision* and NDCG values of our approach are close to those of LDA or BTM. In addition, the average compression rates for the two data sets are 8.32% and 15.28%, respectively, which suggests that 91.68% of WSDL services and 84.72% of OWL-S services are filtered out from the candidate Web service set using our approach, while the most relevant ones related to the queries are retained for ranking.

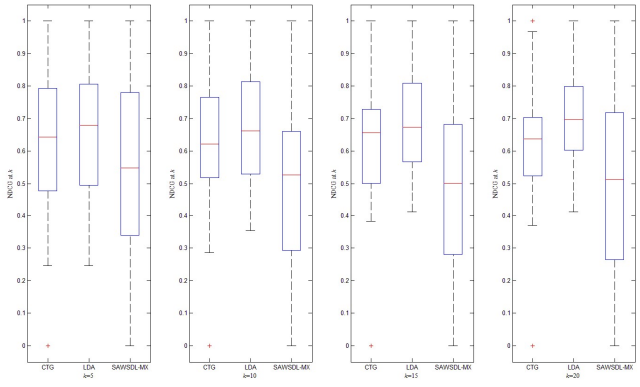
TABLE VII
COMPARISON BETWEEN CTG AND BTM ON OWLS-TC

	Top 5	Top 10	Top 15	Top 20	Average
Precision ratio	98.06%	99.91%	99.33%	100.42%	98.06%
NDCG ratio	104.04%	103.17%	102.97%	103.03%	104.04%
Compression rate	14.83%	15.37%	15.37%	15.55%	15.28%

More remarkably, all the NDCG ratio values in Table VII are over 1, which means that CTG can return more relevant Web services than BTM at a lower cost of service search. That is to say, CTG only uses, on average, 15.28% of all the candidate Web services for ranking. For example, for the query represented in *geopolitical-entity_weatherprocess_service*, five Web services returned by BTM are *GermanIcingcondition*, *GermanWeatherProcessService*, *GRWService*, *GEWeatherSystemService*, and *GEIZWeatherService*, while the corresponding Web services returned by CTG are *GermanWeatherProcessService*, *GRWService*, *GEWeatherSystemService*, *GEIZWeatherService*, and *GRWeatherProcessService*. All of these retrieved Web services are relevant to the query, but *GRWService* and *GRWeatherProcessService* are highly relevant to it. Clearly, for the top five Web services related to the query, CTG returns two highly-relevant Web services while BTM returns only one service.



(a) Precision



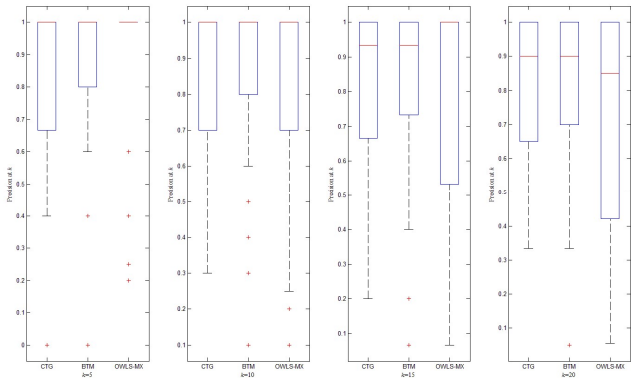
(b) NDCG

Fig. 8. Comparison among CTG, LDA, and SAWSDL-MX on SAWSDL-TC using box plots.

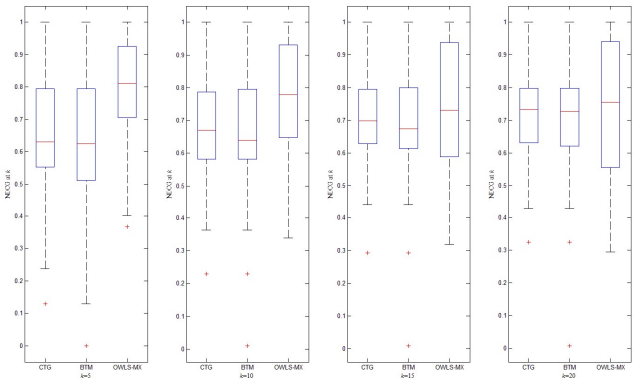
The above analysis is based on the average results over the 42 queries of the two data sets. To make a detailed comparison

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

of distributions of performance results between CTG and other competing approaches, Figs. 8 and 9 depict the standardized box plots of *Precision* and NDCG of these different approaches for SAWSDL-TC and OWLS-TC, respectively. As shown in Fig. 8(a), both the median *Precision* values of CTG and LDA are equal to 1 when $k = 5$, which suggests that CTG can return accurate results (especially the top-ranked Web services) for most of the 42 queries. This finding is supported by the evidence that the median NDCG values of CTG are also close to those of LDA under different settings of k (see Fig. 8(b)). Similarly, it can be seen from Fig. 9 that both the median *Precision* values of CTG and BTM are also equal to 1 when $k = 5$ and $k = 10$. Moreover, the median NDCG values of CTG are slightly greater than those of BTM, though both of the two methods perform worse than OWLS-MX. From the perspectives of the mean value and distribution of values, CTG, compared with the four competing approaches, can obtain the performance of Web service discovery at an elevated level.



(a) Precision



(b) NDCG

Fig. 9. Comparison among CTG, BTM, and OWLS-MX on OWLS-TC using box plots.

Table VIII(a) shows the average response time of CTG, LDA, and SAWSDL-MX on the queries in SAWSDL-TC, while Table VIII(b) shows the average response time of CTG, BTM, and OWLS-MX on the queries in OWLS-TC. It is clear that the response time of CTG is faster than the other four methods. In particular, compared with LDA and BTM, the query response time of CTG on SAWSDL-TC and OWLS-TC is, on average, decreased by about 48.81% and 29.19%, respectively, which is caused by a major reduction in the number of candidate Web services based on CTG matching (see the compression rates in Tables VI and VII).

TABLE VIII
COMPARISON OF DIFFERENT METHODS WITH RESPECT TO AVERAGE QUERY RESPONSE TIME

(a) Results on SAWSDL-TC

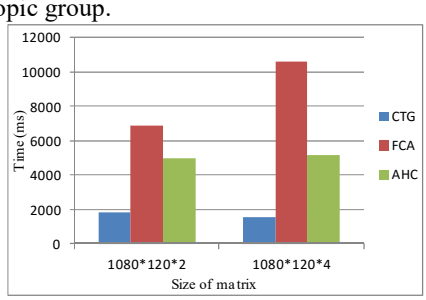
Method	Time(*ms*)
CTG	340.2
LDA	664.6
SAWSDL-MX	1496.1

(b) Results on OWLS-TC

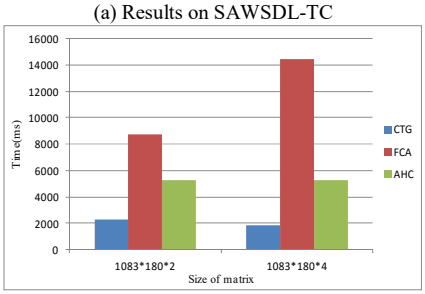
Method	Time(*ms*)
CTG	372.8
BTM	526.5
OWLS-MX	2631.9

E. Further Discussion on CTG Mining

Many widely-used clustering approaches (e.g. formal concept analysis (FCA) and agglomerative hierarchical clustering (AHC)) can also be utilized to cluster and organize Web services according to the topic distributions of their descriptions. We further conducted experiments to compare the computing time of the proposed approach with those of AHC and FCA. As for the experiment on SAWSDL-TC, we used two service-topic distribution matrices whose sizes are 1080*120*2 and 1080*120*4, respectively. Similarly, in the experiment on OWLS-TC, we used two service-topic distribution matrices, with the sizes of 1083*180*2 and 1083*180*4, respectively. According to Fig. 10, the time spent on mining ctgs by our approach is less than the computing time of both FCA and AHC. Note that the computing time of our approach decreases when the scale of the probability grades is increased from 2 to 4, because the number of tiles will decrease with the increase of the number of grades. Moreover, compared with FCA and AHC, our approach is more flexible in defining the area of a target common topic group.



(a) Results on SAWSDL-TC



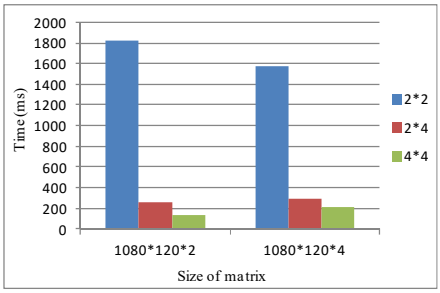
(b) Results on OWLS-TC

Fig. 10. Comparison of computing time among CTG, FCA, and AHC.

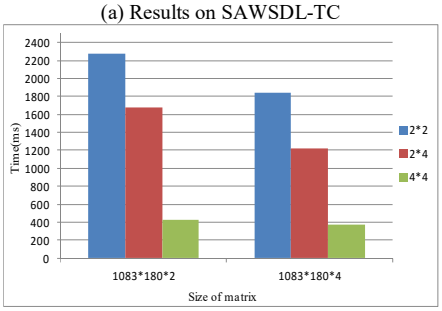
We further evaluated the effect of the change in the area of a ctg on computing time. The default area of a ctg in our approach is 4 (2*2), which means that at least two Web services and at least two topics constitute a ctg. In the contrast experiment, the minimum areas of a ctg were set to 8 (2*4, i.e., 2 Web services and 4 topics) and 16 (4*4, i.e., 4 Web services

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2712744, IEEE Access

and 4 topics), respectively. As shown in Fig. 11, the computing time of our approach is dramatically reduced with the increase of the area of a ctg, because the number of common topic groups is greatly decreased. This suggests that our method is indeed suitable for those large-scale data sets.



(a) Results on SAWSDL-TC



(b) Results on OWLS-TC

Fig. 11. Effect of the change in the area of a ctg on computing time.

F. *Threats to Validity*

Although the experiments provided us with competing results, several internal and external threats to the validity of our work should be taken into account. The *internal validity* concerns the bias and repeatability of experimental results. As noted in the explanation file of OWLS-TC, there is still no standard test collection for Web service retrieval. The test queries as well as the associated relevance sets in the two data sets (i.e. OWLS-TC and SAWSDL-TC) we used are manually defined, which may result in a bias to a certain extent. But it is worth noting that the two data sets used in the experiments are open access, which can contribute to the repeatability of our experiment results.

The *external validity* involves the generalizability of experimental results. Web service discovery is closely related to the formats of Web service descriptions. Currently, typical service description languages include WSDL, OWL-S, WADL, textual description, and so on. The experiments were conducted on Web services described in WSDL and OWL-S, since they are the only two service description languages that have publicly-available test collections for Web service retrieval. An in-depth evaluation on more data sets of other service description languages is still needed to obtain more reliable and repeatable results.

VII. Conclusions and Future Work

Besides an approach to Web service discovery based on common topic groups, this paper presents a few algorithms to mine common topic groups from the generated service-topic distribution matrix using topic modeling. The purpose of extracting common topic groups is to minimize the number of candidate Web services during the process of Web service discovery. The experiments conducted on two publicly-available data sets demonstrate that, compared with the Web service discovery approaches that rank all the candidate Web services according to topic models, our approach can significantly decrease the number of candidate Web services while maintaining high accuracy of Web service discovery. Thus, Web service search engines designed based on our approach will provide end users a better user experience on response time.

In the future, we plan to extend our approach in the following directions. First, we are going to leverage more domain knowledge during the process of CTG mining. For example, the domain knowledge mentioned in [24] including a must-link, which denotes that two words should belong to the same topic, and a cannot-link, which denotes that two words should not belong to the same topic, will be utilized. Second, further decreasing the computing time of CTG mining is another research direction. Third, we will try to model user preferences according to historical usage services and extracted common topic groups.

References

[1] M. Pantazoglou and A. Tsalgatidou, "A Generic Query Model for the Unified Discovery of Heterogeneous Services," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 201–213, 2013.

[2] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and Managing Web Services: Issues, Solutions, and Directions," *The VLDB Journal*, vol. 17, no. 3, pp. 537–572, 2008.

[3] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-Based Automated Service Discovery," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 260–275, 2012.

[4] M. Klusch, P. Kapahnke, and I. Zinnikus, "Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer," in *Proc. ESWC 2009*. LNCS, vol. 5554, pp. 550–564. Springer, Heidelberg, 2009.

[5] J. Wang, C. Zeng, C. He, *et al.*, "Context-Aware Role Mining for Mobile Service Recommendation," in *Proc. 27th Annual ACM Symposium on Applied Computing*, pp. 173–178. ACM Press, New York, 2012.

[6] L. Yao, D. Mimno, and A. McCallum, "Efficient Methods for Topic Model Inference on Streaming Document Collections," in *Proc. 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 937–946. ACM Press, New York, 2009.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[8] X. Wei, and W. B. Croft, "LDA-Based Document Models for Ad-hoc Retrieval," in *Proc. 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 178–185. ACM Press, New York, 2006.

[9] J. Vaidya, V. Atluri, and Q. Guo, "The role mining problem: finding a minimal descriptive set of roles," in *Proc. 12th ACM Symposium on Access Control Models and Technologies*, pp. 175–184. ACM Press, New York, 2007.

[10] M. Klusch, B. Fries, and K. Sycara, "OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services," *Journal of Web Semantics*, vol. 7, no. 2, pp. 121–133, 2009.

[11] M. Klusch, and F. Kaufer, "WSMO-MX: A Hybrid Semantic Web Service Matchmaker," *Web Intelligence and Agent Systems*, vol. 7, no. 1, pp. 23–42, 2009.

[12] K. Mohebbi, S. Ibrahim, M. Khezrian, *et al.*, "A comparative evaluation of semantic web service discovery approaches," in *Proc. 12th International Conference on Information Integration and Web-based Applications & Services*, pp. 33–39. ACM Press, New York, 2010.

[13] X. Liu, S. Agarwal, C. Ding, and Q. Yu, "A LDA-SVM Active Learning Framework For Web Service Classification," in *Proc. 2016 IEEE International Conference on Web Services*, pp. 49–56. IEEE Computer Society Press, San Francisco, 2016.
[14] L. Chen, Y. Wang, Q. Yu, *et al*., "WT-LDA: User Tagging Augmented LDA for Web Service Clustering," in *proc. ICSOC2013*. LNCS, vol. 8274, pp. 162–176. Springer, Heidelberg, 2013.
[15] M. Aznag, M. Quafafou, and Z. Jarir, "Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery," in *Proc. 2014 IEEE International Conference on Web Services*, pp. 153–160. IEEE Computer Society Press, New York, 2014.
[16] D. M. Blei, and J. D. Lafferty, "Correlated topic models," *Advances in Neural Information Processing Systems*, vol. 18, pp. 147–154, 2006.
[17] X. Cheng, X. Yan, Y. Lan, and J. Guo, "BTM: Topic Modeling over Short Texts," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 12, pp. 2928–2941, 2014.
[18] V. W. Chu, R. K. Wong, and C. H. Chi, "Online Role Mining without Over-fitting for Service Recommendation," in *Proc. 20th IEEE International Conference on Web Services*, pp. 58–65. IEEE Computer Society Press, New York, 2013.
[19] R. K. Wong, V. W. Chu, and T. Hao, "Online role mining for context-aware mobile service recommendation," *Personal and Ubiquitous Computing*, vol. 18, no. 5, pp. 1029–1046, 2014.
[20] U. Bellur and R. Kulkarni, "Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching," in *Proc. 2007 IEEE International Conference on Web Service*, pp. 86–93, IEEE Computer Society Press, New York, 2007.
[21] J. Wu, L. Chen, Z. Zheng, M. R. Lyu, and Z. Wu, "Clustering Web Services to Facilitate Service Discovery," *International Journal of Knowledge and Information Systems*, vol. 38, no. 1, pp. 207–229, 2014.
[22] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering WSDL documents to bootstrap the discovery of web services," in *Proc. 2009 IEEE International Conference on Web Services*, pp. 147–154. IEEE Computer Society Press, New York, 2009.
[23] J. Wang, N. Zhang, C. Zeng, Z. Li, and K. Q. He, "Towards Services Discovery based on Service Goal Extraction and Recommendation," in *Proc. 2013 IEEE International Conference on Services Computing*, pp. 65–72, IEEE Computer Society Press, New York, 2013.
[24] Z. Chen and B. Liu, "Mining topics in documents: standing on the shoulders of big data," in *Proc. 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1116–1125. ACM Press, New York, 2014.
[25] M. Aznag, M. Quafafou, and Z. Jarir, "Correlated Topic Model for Web Services Ranking," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 6, pp. 283–291, 2013.
[26] Q. Yu, "Place semantics into context: Service community discovery from the WSDL corpus," in *Proc. ICSOC 2011*. LNCS, vol. 7084, pp. 188–203. Springer, Heidelberg, 2011.
[27] T. L. Griffiths, and M. Steyvers, "Finding Scientific Topics," in *Proc. National Academy of Sciences of the United States of America*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.
[28] M. Sabou, C. Wroe, C. Goble, and H. Stuckenschmidt, "Learning domain ontologies for semantic Web service descriptions," *Journal of Web Semantics*, vol. 3, no. 4, pp. 340–365, 2005.
[29] Y. J. Lee and C. S. Kim, "A Learning Ontology Method for RESTful Semantic Web Services," in *Proc. 2011 IEEE International Conference on Web Services*, pp. 251–258. IEEE Computer Society Press, New York, 2011.
[30] A. Segev and Q. Z. Sheng, "Bootstrapping Ontologies for Web Services," *IEEE Transactions on Services Computing*, vol. 5, no. 1, pp. 33–44, 2012.
[31] G. Cassar, P. Barnaghi, and K. Moessner, "Probabilistic methods for service clustering," in *Proc. 4th International Workshop on Semantic Web Service Matchmaking and Resource Retrieval*, p. 1. CEUR Workshop Proceedings, Aachen, 2010.
[32] C. Burges, T. Shaked, E. Renshaw, *et al.*, "Learning to rank using gradient descent," in *Proc. 22nd International Conference on Machine Learning*, pp. 89–96. ACM, New York, 2005.
[33] P. Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An Integrated Semantic Web Service Discovery and Composition Framework," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 537–550, 2016.
[34] D. Roman, J. Kopecký, T. Vitvar, *et al.*, "WSMO-Lite and hRESTS: Lightweight semantic annotations for Web services and RESTful APIs," *Journal of Web Semantics*, vol. 31, pp. 39–58, 2015.
[35] W. Chen, I. Paik, I., and P. C. K. Hung, "Constructing a Global Social Service Network for Better Quality of Web Service Discovery," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 284–298, 2015.
[36] J. Liu, K. He, J. Wang, *et al.*, "Service Organization and Recommendation Using Multi-granularity Approach," *Knowledge-Based Systems*, vol. 73, pp. 181–198, 2015.
[37] J. Zhang, R. Shi, W. Wang, *et al.*, "A Bloom Filter-Powered Technique Supporting Scalable Semantic Service Discovery in Service Networks," in *Proc. 2016 IEEE International Conference on Web Services*, pp. 81–90. IEEE Computer Society Press, San Francisco, 2016.
[38] N. Zhang, J. Wang, and Y. Ma, "Mining Domain Knowledge on Service Goals from Textual Service Descriptions," *IEEE Transactions on Services Computing*, 2017, DOI: 10.1109/TSC.2017.2693147.
[39] J. Wang, P. Gao, Y. Ma, and K. He, "Common Topic Group Mining for Web Service Discovery," in *Proc. the 2015 Asia-Pacific Services Computing Conference* (*APSCC 2015*), LNCS 9464, pp. 92–107. Springer, Heidelberg, 2015.

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

Panpan Gao received the master degree in 2016 from Wuhan University, China. His current research interests include software engineering and services computing.

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

Keqing He received the Ph.D. degree in 1995 from Hokkaido University, Japan. He is now a Professor at the State Key Laboratory of Software Engineering, Wuhan University, China. His current research interests include software engineering and services computing.

Patrick C.K. Hung is an Associate Professor at the Faculty of Business and Information Technology in University of Ontario Institute of Technology, Canada. He is an Honorary International Chair Professor at National Taipei University of Technology in Taiwan, an Adjunct Professor at Nanjing University of Information Science & Technology in China, a Visiting Professor at University of Sao Paulo in Brazil, and an Adjunct Professor at Wuhan University. He is a founding committee member of the IEEE International Conference of Web Services (ICWS), IEEE International Conference on Services Computing (SCC), and IEEE BigData Congress (BigData Congress).