Journal of Southeast University (English Edition) Vol. 25, No. 4, pp. 468 – 472 Dec. 2009 ISSN 1003—7985

Task-oriented web service discovery algorithm using semantic similarity for adaptive service composition

Wen Junhao[1,2] Jiang Zhuo[1] Tu Liyun[2] He Pan[1]

([1] College of Computer Science, Chongqing University, Chongqing 400030, China)
([2] College of Software Engineering, Chongqing University, Chongqing 400030, China)

Abstract: In order to achieve adaptive and efficient service composition, a task-oriented algorithm for discovering services is proposed. The traditional process of service composition is divided into semantic discovery and functional matching and makes tasks be operation objects. Semantic similarity is used to discover services matching a specific task and then generate a corresponding task-oriented web service composition (TWC) graph. Moreover, an algorithm for the new service is designed to update the TWC. The approach is applied to the composition model, in which the TWC is searched to obtain an optimal path and the final service composition is output. Also, the model can implement real-time updating with changing environments. Experimental results demonstrate the feasibility and effectiveness of the algorithm and indicate that the maximum searching radius can be set to 2 to achieve an equilibrium point of quality and quantity.

Key words: web service; service composition; service discovery; ontology; semantic similarity

Web service composition is the process of discovering, combining and executing existing services to establish reusable and versatile interoperability application[1]. As the key technology, service discovery directly affects the quality of service composition. But the variability of requests, as well as the uncertainty and instability of the external services[2], forbid precise predictions from service providers about the outcomes of service composition. On account of the deficiency of service discovery, service composition cannot achieve self-adaptation and self-optimization when faced with changing environments.

Researches dealing with service discovery in composition have been on the way. A self-healing service composition is proposed, which is realized by process description, monitoring and recovery strategies[3]. Transforming the composite services dependability maintenance problem to an adaptive control problem can present a dependable and adaptive approach[4]. A service composition with high quality and good performance can be realized by the optimized genetic algorithm[5-6] based on the QoS. Besides, taking ontology and semantic similarity as the basic tool to discover services on the web has become a consensus among many researchers[7-10]. However, one problem, that of taking all of the available services as operation objects takes too much time and cannot deal with changes in the environment quickly enough, still needs to be solved.

The approach proposed in this paper divides the process of service composition into semantic discovery and functional matching and takes tasks as operation objects. An algorithm is designed to make use of semantic features to discover services matching a given task and generate corresponding TWC graphs. After that, the TWC is searched to obtain the final service composition.

1 Ontology-Based Semantic Similarity

Ontology, as the basis of the semantic web, is an approach of conceptualization and modeling of domain knowledge which can be used to describe the semantic information of data. There are several language describing ontologies, such as simple HTML ontology extension (SHOE), XML-based ontology-exchange language (XOL), RDF, RDF-S and the OWL standard with recommendations from W3C. Based on OWL rules, the definition of a domain ontology is proposed as follows[11].

Definition 1 Domain ontology is defined as

$$\mathrm{Ont} = \langle C,\ R,\ H,\ \mathrm{rel},\ A \rangle$$

where C is the set of concepts, including the interpretation of the ontology, the definition of properties and the method of the ontology; R is the set of relations; H is the conceptual level of the ontology in the ontological tree structure; rel represents the relationships between concepts; A represents ontology axioms.

In this way, the similarity between the entities is converted to the semantic similarity between concepts. In order to calculate the similarity more accurately, the ontology weight associated with the parent node is introduced in the corresponding concept hierarchy tree (CHT).

Definition 2 For child nodes Ont_i belonging to Ont, $i = 1, 2, \ldots, n$ and R_i should include at least the following information,

$$R_i = \langle \mathrm{parentOnt},\ \mathrm{parW}_i,\ \mathrm{parW}_i^* \rangle$$

where parentOnt is the parent node; parW_i is the weight of all the child-nodes belonging to parentOnt, which reflects the similarity degree between Ont_i and parentOnt, and $\sum_{i=1}^{n} \mathrm{parW}_i = 1$. We can normalize the weight by $\mathrm{parW}_i^* = \mathrm{parW}_i / \max(\mathrm{parW}_i)$.

To be precise, the weight parW_i^* reflects the difference of similarity degree between the same parent and different chil-

Received 2009-06-30.

Biography: Wen Junhao (1969—), male, doctor, professor, jhwen@cqu.edu.cn.

Foundation items: The National Key Technology R&D Program of China during the 11th Five-Year Plan Period (No. 2007BAF23B0302), the Major Research Plan of the National Natural Science Foundation of China (No. 90818028).

Citation: Wen Junhao, Jiang Zhuo, Tu Liyun, et al. Task-oriented web service discovery algorithm using semantic similarity for adaptive service composition[J]. Journal of Southeast University (English Edition), 2009, 25(4): 468 – 472.

dren. For example, "Hard Disk" and "CD" both can be the children of "Memory". But when talking about "Memory", apparently we refer to "Hard Disk" instead of "CD", which suggests that the weight of the former is larger than that of the latter. The weight can be calculated, for example, according to frequency of use in the context, which can be obtained from the WordNet[12] as well as the Chinese WordNet supported by Southeast University.

The calculation method of similarity between concepts is proposed in Refs. [11, 13]. Based on the analysis above, a light-weight form of the method is presented as follows:

$$\text{simDegree}(C_1, C_2) = \begin{cases} \dfrac{L \cdot \text{parW}_1^* \cdot \text{parW}_2^*}{(l_1 + l_2 - 2\alpha L)\max(|l_1 - l_2|, 1)} & C_1 \neq C_2 \\ 1 & C_1 = C_2 \end{cases} \quad (1)$$

where $0 \leqslant \alpha \leqslant 1$; l_1 and l_2 are the levels of C_1 and C_2; L is the level of the lowest common parent-node of C_1 and C_2 in the concept hierarchy tree,

$$L = \begin{cases} l_1 & C_1 \text{ is parent of } C_2 \text{ or } C_1 = C_2 \\ l_2 & C_2 \text{ is parent of } C_1 \\ \text{Level of the lowest common parent} & \text{else} \end{cases} \quad (2)$$

As shown in Eq. (1), taking no account of parW, the lower of the common parent of two concepts is in the concept hierarchy tree, the more similar the two concepts are. In Fig. 1, for example, if $\alpha = 0.5$, then simDegree (C_5, C_6) = 0.5 and simDegree(C_5, C_7) = 0.2, where C_5, C_6 and C_7 are in the same level. Since the level of the common parent of C_5 and C_6 is lower than that of C_5 and C_7, the two of the former are more similar to each other.

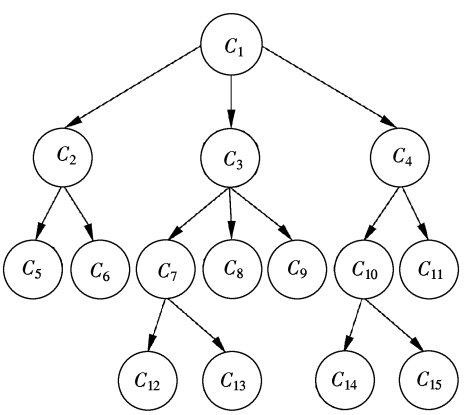

Fig. 1 Concept hierarchy tree

In order to take advantage of semantic similarity in service composition, we need to define the similarity calculation method of multiple concepts composition. If there is a combination, CA = (C_1, C_2, …, C_k) and CB = C_{k+1}, the similarity degree between them is simDegree (CA, CB) = $\prod_{i=0}^{k-1} \text{simDegree}(C_i, C_{i+1})$.

2 Task-Oriented Web Service Discovery Algorithm

2.1 Overall approach

The concept of "task" is introduced to optimize the service composition and realize the adaptive capacity. With the given tasks, the registered services are abstracted to the task layer based on the semantic similarity degree, generating the corresponding TWC graph. After that, we just need to search the generated TWC to find a QoS optimal path with functional matching as the composition result. TWC also can be updated adaptively when there are invalid services or new services.

2.2 Algorithm design

A registered service entity should include the most basic information of function description and interface description. Based on the domain ontology, a service can be abstracted to an entity only containing output and input. Therefore, matching of services can be abstracted to matching of concepts which are related to I/O interfaces of services. Moreover, the concept of task can be introduced to the web service[14-15].

Definition 3 Task represents specific functions and business processes in some area, and it is achieved by one or a group of web services together. $t = \langle I, O \rangle$, where I and O are the input and the output, respectively.

Definition 4 Let the services composition path be WS = ($\text{WS}_1(I_1, O_1)$, $\text{WS}_2(I_2, O_2)$, …, $\text{WS}_i(I_i, O_i)$), task be Task(I^t, O^t), similarity of the input and output threshold be S_{IO}^*, and the composition similarity threshold is S_C^*. If simDegree(I_1, I^t) $\geqslant S_{IO}^*$, simDegree(O_i, O^t) $\geqslant S_{IO}^*$, simDegree(O_k, I_{k+1}) $\geqslant S_{IO}^*$ and simDegree(WT, Task) $\geqslant S_C^*$, then WS is considered to match Task(I^t, O^t), where $i \geqslant 1$, $1 \leqslant k \leqslant i-1$.

Through the calculation of the semantic similarity, services can be matched to tasks and abstracted to the logic layer. The mapping relationship is shown in Fig. 2, where the task layer is just the generated TWC graph.

In Fig. 2, web service request, WSR(I, O), can be divided into three interdependent tasks. According to the semantic similarity of concepts, we can obtain the set of candidate services for each task through mapping all the operational services to corresponding tasks. In addition, the figure informs that, WS6 can independently accomplish Task 1, while WS1 and WS2 are required to be executed in sequence to achieve the same task; WS3 can independently accomplish both Task 2 and Task 3 simultaneously. While searching the path in the follow-up composition, we take WS3 as two different services and set the distance between them as 0.

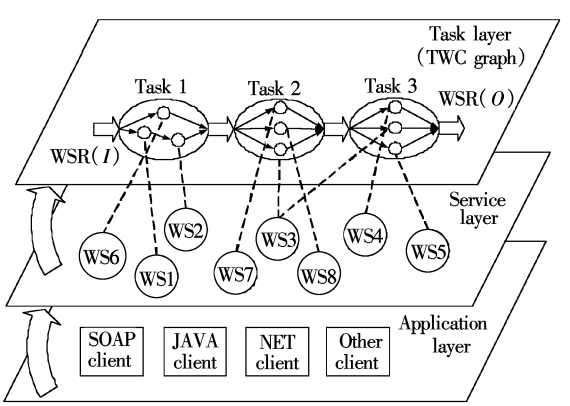

Fig. 2 Overview of mapping model

Therefore, under the premise of ascertained tasks, a global traversal search for all the matching services or service compositions is required to generate TWC. However, the similarity between the composition and the task and the value of other QoS attributes will decrease when the service composition path becomes longer. So a maximum searching radius $L_{\max}$ is set to limit the searching radius. The value of $L_{\max}$ should be based on the actual situation to make a balance between time complexity of the algorithm and the recall rate. In the following example of the algorithm, the searching radius is 2, which means that the length of the service composition path matching Task(I^{t}, O^{t}) is less than 2.

Before giving the algorithm, it is necessary to propose the definition of similarity matrix.

Definition 5 Similarity matrix of the set of services $\{\mathrm{WS}_i(I_i^{\mathrm{s}}, O_i^{\mathrm{s}})\}$ can be defined as $\boldsymbol{A} = (a_{jk})_{i\times i}$, where a_{jk} = simDegree(O_j^{s}, I_k^{s}).

Algorithm 1 Match services to task(MSTT)

Input: Similarity threshold of input and output: S_{IO}^{*}; similarity threshold of composition: S_{C}^{*}; set of available services: $\{\mathrm{WS}_i(I_i^{\mathrm{s}}, O_i^{\mathrm{s}})\}$ and Task(I^{t}, O^{t}); similarity matrix of $\{\mathrm{WS}_i(I_i^{\mathrm{s}}, O_i^{\mathrm{s}})\}$ and Task(I^{t}, O^{t}): $\boldsymbol{A}$; row and column number of tasks in $\boldsymbol{A}$: t_1, t_2.

Output: Set of services matching to the task: WT.

1 Initialize WT = $\varnothing$, $k=1$, next step;
2 If $\boldsymbol{A}(t_1, k) >= S_{\mathrm{IO}}^{*}$, go to step 4; else next step;
3 $k=k+1$; if $k>=1$ && $k<=i$, go to step 2; else go to step 10;
4 If $\boldsymbol{A}(k, t_2) >= S_{\mathrm{IO}}^{*}$, next step; else go to step 6;
5 If $S=\boldsymbol{A}(t_1, k)\boldsymbol{A}(k, t_2) >= S_{\mathrm{C}}^{*}$, add ($S$, WS_k) to WT; go to step 3;
6 $j=1$; next step;
7 If $\boldsymbol{A}(k, j) >= S_{\mathrm{IO}}^{*}$, go to step 9; else next step;
8 $j=j+1$; if $j>=1$ && $j<=i$, go to step 7; else go to step 3;
9 If $\boldsymbol{A}(j, t_2) >= S_{\mathrm{IO}}^{*}$ && $S=\boldsymbol{A}(t_1, k)\boldsymbol{A}(k, j)\boldsymbol{A}(j, t_2) >= S_{\mathrm{C}}^{*}$, add wt = ($S$, WS_k, WS_j) to WT; go to step 8;
10 Return to WT

Steps 1 to 5 search for any independent services WS_k matching Task, and steps 6 to 9 for any composition of services in the space of radius 2. The MSTT can commendably handle the situation, as shown in Fig. 2, and WS1 and WS2 are required to be linked together in sequence to match Task 1. In addition, another algorithm, namely match tasks to service(MTTS), which can deal with the situation that WS3 independently accomplishes both Task 2 and Task 3 simultaneously, should be designed. The idea is basically the same as the MSTT, so we do not repeat it here.

In fact, a global search is only executed when a large number of services are invalid or when the system is updated regularly. When a new service is registered, the search algorithm should only be aiming at the new one to ensure the real-time update. The algorithm of matching a new service to task is as follows.

Algorithm 2 Match new service to task(MNSTT)

Input: S_{IO}^{*}; S_{C}^{*}; $\{\mathrm{WS}_i(I_i^{\mathrm{s}}, O_i^{\mathrm{s}})\}$; Task($I^{\mathrm{t}}$, O^{t}); $\boldsymbol{A}$; t_1, t_2; new service NWS($I_{\mathrm{n}}^{\mathrm{s}}$, $O_{\mathrm{n}}^{\mathrm{s}}$).

Output: WT.

1 If simDegree($I_{\mathrm{n}}^{\mathrm{s}}$, I^{t}) >= S_{IO}^{*}, next step; else go to step 8;
2 If simDegree($O_{\mathrm{n}}^{\mathrm{s}}$, O^{t}) >= S_{IO}^{*}, next step; else $k=1$, go to step 4;
3 If S = simDegree(I_n^{s}, I^{t}) simDegree($O_{\mathrm{n}}^{\mathrm{s}}$, O^{t}) >= S_{C}^{*}, add (S, NWS) to WT; step 12.
4 If simDegree($O_{\mathrm{n}}^{\mathrm{s}}$, I_k^{s}) >= S_{IO}^{*}, go to step 6; else next;
5 $k=k+1$; if $k>=1$ && $k<=i$, next step; else step 12.
6 If $\boldsymbol{A}(k, t_2) >= S_{\mathrm{IO}}^{*}$, next; else go to step 5;
7 If S = simDegree($I_{\mathrm{n}}^{\mathrm{s}}$, I^{t}) simDegree($O_{\mathrm{n}}^{\mathrm{s}}$, I_k^{s}) $\boldsymbol{A}(k, t_2) >= S_{\mathrm{C}}^{*}$, add wt = ($S$, NWS, WS_k) to WT; go to step 5;
8 If simDegree($O_{\mathrm{n}}^{\mathrm{s}}$, O^{t}) >= S_{IO}^{*}, $j=1$, next step; else step 12.
9 If $\boldsymbol{A}(t_1, j) >= S_{\mathrm{IO}}^{*}$, go to step 11; else next;
10 $j=j+1$; if $j>=1$ && $j<=i$, go to step 9; else step 12.
11 If $S=\boldsymbol{A}(t_1, j)$ simDegree(O_j^{s}, $I_{\mathrm{n}}^{\mathrm{s}}$) simDegree($O_{\mathrm{n}}^{\mathrm{s}}$, O^{t}) >= S_{C}^{*} && simDegree(O_j^{s}, $I_{\mathrm{n}}^{\mathrm{s}}$) >= S_{IO}^{*}, add wt = (S, WS_j, NWS) to WT; go to step 10;
12 Return to WT.

Steps 1 to 3 check whether NWS independently matches Task, and the remaining steps discover whether any composition of services matches Task and contains NWS.

2.3 Adaptive service composition model

Based on the above algorithm, we can discover a set of candidates from the available services to match a certain task. If the discovery algorithm is executed for all the given tasks, we can obtain the TWC graph, and ensure the real-time update.

As shown in Fig. 3, this approach can adaptively respond to the environmental changes. The main task of functional composition and allocation is performed by the adaptive path selection model(APSM).

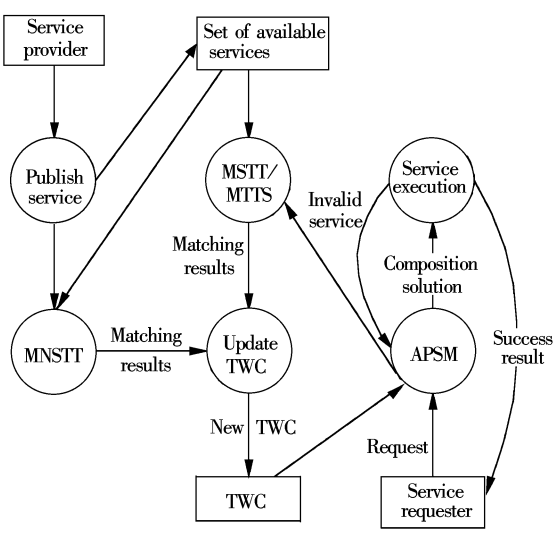

Fig. 3 Adaptive service composition model

1) After TWC is generated, the APSM calls the path search algorithm, such as the genetic algorithm[5-6], to confirm the optimal composition path. The basic method is to enumerate all the available paths and find the optimal one

with several weights of similarity and other QoS.

2) Send the composition solution to service execution. If there is information about invalid service WS_e, execute step 3).

3) Find out the task Task* according to WS_e and the corresponding set of candidates, WT, from TWC. If there are still enough available services in WT, take out an approximate optimal service instead to generate a new solution and execute step 2); if the number of candidates is less than a given minimum value, execute step 4).

4) Invoke MSTT/MTTS to update TWC, if necessary, with decreasing S_{IO}^*, S_C^* or increasing L_{max}.

3 Experiment and Analysis

We designed experiments to test the execution efficiency and the success rate of the MSTT and the MNSTT. The algorithms have been implemented in Matlab 7.8. All the experiments are conducted in a hardware configuration with an AMD Turion 64 × 2, 1.9 GHz CPU and a 2 GB RAM.

The experimental data originated from pizzas.owl version 1.5 published by Manchester University. The concept hierarchy tree is generated according to the relational description in the food domain, and $parW_i$ is calculated based on the word frequency provided in WordNet. We randomly select concepts of ontology from the generated domain ontology database as input and output of the service to obtain the test service dataset. Similarity matrix $\boldsymbol{A}$ is calculated according to definition 5.

Obviously, with a greater similarity threshold, the composition result of the algorithm matches the user's request better. On the other hand, in order to clearly show the differences between the experimental results, the experiment should not be too fine-grained. Therefore, we set threshold $S_{IO}^* = S_C^* = 0.9$ and execute the MSTT algorithm with different L_{max}. And the experiment selects three random groups of datasets as the test data, records the computing time and matching results, and takes the average value as the final result. The MNSTT algorithm with $L_{max} = 2$ is also tested while comparing the computing time.

Fig. 4 shows the comparison of computing time when L_{max} varies from 1 to 4. "MNSTT with radius 2" represents the cost time of the MNSTT algorithm with $L_{max} = 2$. From Fig. 4, we can see that the greater the L_{max} is, the more time the MSTT takes; while for the MNSTT with $L_{max} = 2$, the time cost is obviously less than that using the MSTT under the same conditions. For example, supposing that $L_{max} = 2$, the number of web services is 500, the time cost using the MNSTT is 6.9 ms while it is 15.6 ms using the MSTT.

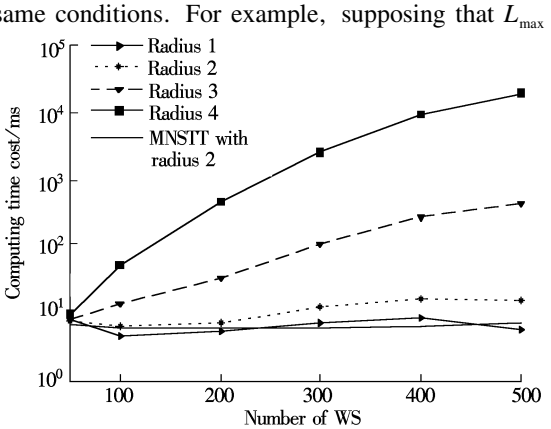

Fig. 4 Variation in computing time cost

Fig. 5 shows the quantity of the successful matching results with different L_{max} when the number of web services is from 50 to 500. The experimental results indicate that when L_{max} is set to 1, we often obtain only little or no service matches. However, with the increase in searching radius, the number of successful matches is dramatically increased.

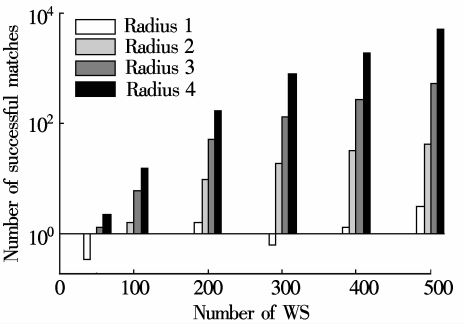

Fig. 5 Variation in successful matches

Some conclusions can be drawn from the experimental results:

1) The computing time and the number of the successful matches are proportional to the maximum searching radius L_{max}. The greater L_{max} is, the more time it costs, and the number of the successful matches increases. However, with the increase of L_{max}, the global similarity of service composition is decreased, so are the values of some QoS attributes, such as network delay. So, it is meaningless to infinitely increase L_{max} to obtain more matching results. In fact, the experimental results show that when L_{max} is set to 2, the results are satisfactory enough. Consequently, $L_{max} = 2$ can be an equilibrium point of quality and quantity.

2) Given the same service scale and searching radius, the time of the MNSTT algorithm is obviously lower than that of the MSTT algorithm. When a new service is registered, there is no need to refresh the whole TWC and we only need to search this new registered service. With the MNSTT, we not only implement real-time updating but also save time and resources.

4 Conclusion

We present a task-oriented discovery algorithm and apply it to web service composition. Compared with the current composition model, the model we proposed divides the process into two phases: simplify the operation objects and make the composition efficient and flexible. This approach achieves self-adaptation and self-optimization when faced with changing environments. The simulation results show that the algorithm is feasible and effective and can meet the actual requirements. However, this paper does not put great stress on the algorithm of the automatic division of task and the algorithm for searching the optimal path based on TWC. This will be the emphasis of our future work.

References

[1] Qiu Lirong, Shi Zhongzhi, Lin Fen, et al. Agent-based automatic composition of semantic web services [J]. *Journal*

of Computer Research and Development, 2007, **44**(4): 643 - 650. (in Chinese)
[2] Mei Lijun, Chan W K, Tse T H. An adaptive service selection approach to service composition [C]//*IEEE International Conference on Web Services*. Beijing, China, 2008: 70 - 77.
[3] Guinea S. Self-healing web service compositions [C]//*The* 27*th International Conference on Software Engineering*. Saint Louis, MO, USA, 2005: 655.
[4] Guo Huipeng, Huai Jinpeng, Deng Ting, et al. A dependable and adaptive approach to supporting web service composition [J]. *Chinese Journal of Computers*, 2008, **31**(8): 1434 - 1444. (in Chinese)
[5] Ai Lifeng, Tang Maolin. QoS-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm [C]//*Proceedings of the Fourth IEEE International Conference on eScience*. Indianapolis, IN, USA, 2008: 119 - 126.
[6] Yang Lei, Dai Yu, Zhang Bin, et al. Dynamic selection of composite web services based on a genetic algorithm optimized new structured neural network [C]//*Proceedings of the* 2005 *International Conference on Cyberworlds*. Singapore, 2005: 515 - 522.
[7] Buford J, Brown A, Kolberg M. Meta service discovery [C]//*Proceedings of the Fourth IEEE International Conference on Pervasive Computing and Communications Workshops* (*PERCOMW*'06). Pisa, Italy, 2006: 129 - 134.
[8] Feng Zaiwen, He Keqing, Li Bing, et al. Method for semantic web service discovery based on context inference [J]. *Chinese Journal of Computers*, 2008, **31**(8): 1354 - 1363. (in Chinese)
[9] Wu Chen, Chang E, Aitken A. An empirical approach for semantic web services discovery [C]//*Proceedings of the Australian Software Engineering Conference*. Perth, WA, Australia, 2008: 412 - 421.
[10] Shi Bin, Wang Haiyang, Cui Lizhen, et al. Service composition algorithm using semantic constraint to implement user personality [J]. *Journal of Southeast University*: *English Edition*, 2008, **24**(3): 365 - 368.
[11] Li Man, Wang Dazhi, Du Xiaoyong, et al. Dynamic composition of web services based on domain ontology [J]. *Chinese Journal of Computers*, 2005, **28**(4): 644 - 650. (in Chinese)
[12] Fellbaum C. *WordNet*: *an electronic lexical database* [M]. MIT Press, 1998.
[13] Wu Jian, Wu Zhaohui, Li Ying, et al. Web service discovery based on ontology and similarity of words [J]. *Chinese Journal of Computers*, 2005, **28**(4): 595 - 602. (in Chinese)
[14] Shang Zongmin, Cui Lizhen, Wang Haiyang, et al. Research on exception handling of composite services based on compensation business process graph [J]. *Chinese Journal of Computers*, 2008, **31**(8): 1478 - 1490. (in Chinese)
[15] Sirin E, Parsia B, Wu D, et al. HTN planning for web service composition using SHOP2 [J]. *Journal of Web Semantics*, 2004, **1**(4): 377 - 396.

自适应服务组合中基于语义相似度及面向任务的服务发现算法

文俊浩[1,2] 江 卓[1] 涂丽云[2] 何 盼[1]

([1]重庆大学计算机学院, 重庆 400030)
([2]重庆大学软件工程学院, 重庆 400030)

摘要: 为了实现自适应和高效的 Web 服务组合, 提出了一种面向任务的服务发现算法. 将传统的服务组合过程划分为语义上的发现和功能上的匹配, 并将任务作为操作对象, 利用语义相似度从候选服务中寻找与给定任务相匹配的服务, 并生成对应的 TWC 图, 同时针对新服务设计更新算法. 将该方法应用到服务组合模型中, 搜索 TWC 图以得到一条最优的路径作为服务组合结果输出, 并能在服务失效时做出实时的更新以完成服务请求. 实验结果证明了算法的可行性和有效性, 也表明当最大搜索半径取 2 时, 服务数量和质量之间可达到平衡.

关键词: Web 服务; 服务组合; 服务发现; 本体; 语义相似度
中图分类号: TP331