Hybrid Semantic Service Matchmaking Method Based on a Random Forest

Wei Jiang
the College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China.

Junyu Lin
the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China.

Huiqiang Wang
the College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China.

Shichen Zou
Huawei Nanjing Research Institute, Nanjing 210012, China.

Follow this and additional works at: https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology

Part of the Computer Sciences Commons, and the Electrical and Computer Engineering Commons

Recommended Citation
Wei Jiang, Junyu Lin, Huiqiang Wang et al. Hybrid Semantic Service Matchmaking Method Based on a Random Forest. Tsinghua Science and Technology 2020, 25(6): 798-812.
Hybrid Semantic Service Matchmaking Method Based on a Random Forest

Wei Jiang, Junyu Lin*, Huiqiang Wang, and Shichen Zou

Abstract: Semantic Service Matchmaking (SSM) can be leveraged for mining the most suitable service to accommodate a diversity of user demands. However, existing research on SSM mostly involves logical or non-logical matching, leading to unavoidable false-positive and false-negative problems. Combining different types of SSM methods is an effective way to improve this situation, but the adaptive combination of different service matching methods is still a difficult issue. To conquer this difficulty, a hybrid SSM method, which is based on a random forest and combines the advantages of existing SSM methods, is proposed in this paper. The result of each SSM method is treated as a multi-dimensional feature vector input for the random forest, converting the service matching into a two classification problem. Therefore, our method avoids the flaws found in manual threshold setting. Experimental results show that the proposed method achieves an outstanding performance.

Key words: Semantic Service Matchmaking (SSM); random forest; logic-based service matchmaking; false-positive; false-negative

1 Introduction

Semantic Service Matchmaking (SSM) is in a prominent place in service computing by discovering services that satisfy user needs from a large number of services, improving service system efficiency and user satisfaction. However, with the popularity of cloud computing technology and service crowdsourcing[1], the service computing mode is undergoing profound changes. Services that meet user needs are no longer just from existing local or internet services, but are more likely from remote clouds in a service composition manner[2].

Dynamic service construction in cloud computing environments is gradually replacing the traditional software system, becoming the most cost-effective way for users to meet their own IT requirements[3]. However, in a remote virtualized cloud computing environment, available services may have heterogeneous functions, heterogeneous preconditions, and input/output descriptions, while still meeting users’ needs.

Facing these new challenges, completing SSM with traditional strict ontology-based methods may be impossible. At present, there are mainly two kinds of service matchmaking methods: logical SSM and non-logical SSM. The former performs well in precision through rigorous logical reasoning, but it is less ideal in recall due to its strictness, especially when the preconditions and input/output concepts of semantic services are heterogeneous. By contrast, the latter is mainly matched by non-logical reasoning methods, such as text similarity[4], XML/RDF graph matching[5], and data mining[6], yet the precision is not as good as that in logic-based methods when dealing with heterogeneous functions of services. In fact, the services provided by the logic- or non-logic-based matching services...
can barely fully satisfy the specific requirements of users because different users have different perceptions regarding the same services. In other words, good results cannot be easily achieved with the use of a single type of a service matching method.

Combining different types of SSM methods to conquer the two intrinsic deficiencies mentioned above is a recently emerging solution, e.g., OWLS-MX3\cite{7} and LOG4SWS\cite{8}. For these hybrid SSM methods, the service matching process can be used to classify the matching results of different methods as a success or failure. Moreover, a comprehensive service matching degree can be obtained by aggregating different kinds of matching values with preassigned weights, whereas a successful match can be achieved when the aggregated matching exceeds its threshold. Nevertheless, the weight and threshold settings are hardly justified up to now. In Refs. [7] and [8], thresholds need to be manually preset.

Motivated by this challenge, in this paper, a new hybrid SSM method is designed based on a random forest. Our biggest innovation is in avoiding flaws in the manual setting of the threshold, since it often lacks enough evidence to support the selected threshold and the decision-making process is very complicated due to collect a massive amount of evidence. First, four common SSM methods, including logic-based methods and non-logic-based methods, are used as basic matching methods to make full use of their advantages. Then, the obtained results of the SSM methods are treated as eigenvalues to carry on feature classification through a random forest method. Finally, a comprehensive service matching degree is generated after the classification without manually setting the threshold.

The main contributions of this study are summarized as follows:

1. By introducing the random forest method, we propose a hybrid SSM method based on feature learning classification, in which the random forest classification is used to determine the matching degree between services and user demands, instead of the manual threshold setting.

2. We propose an approximate logic-based matching method to address the false-negative problem caused by a strict logic, in which the logical concepts contraction and abduction\cite{9} are used to relax the concept ontology constraints. On this basis, users may discover some of the available heterogeneous services in the cloud computing system, which are difficult to identify through traditional logical methods.

3. The false-positive issue of service matching is further mitigated by introducing the precondition and effect matching method and the state-space prune, which can improve the recall rate.

The paper is structured as follows: The background of the study is given in Section 2. In Section 3, a random forest-based SSM method is proposed to avoid the flaws in the threshold division. Next, an SSM method based on an approximate logic, which aims to overcome the shortcoming of the strict logical matching, is introduced in Section 4. While non-logical matching methods, such as service description text similarity matching and ontology structure similarity matching, are discussed in Section 5. Then, an experiment simulation analysis based on the existing public dataset is presented in Section 6. Our work is summarized in the last section.

2 Background

2.1 SSM

The target of SSM is to satisfy user needs by finding suitable services from a pool of services, e.g., the cloud computing system and edge computing platform. Existing SSM methods can generally be categorized into two types of service matching: logic-based and non-logic-based. Logic-based service matching focuses on logical reasoning and deduction from the semantic service description information, e.g., SPARQLent\cite{9}. SPARQLent adopts the Resource Description Framework (RDF) entailment rule to conduct service matchmaking and use SPARQL (that is SPARQL Protocol and RDF Query Language) to describe preconditions, effects, and input/output concepts of semantic services, where SPARQL is an concept of recursion. As such, the logic-based method can achieve a high precision ratio due to its strict logical reasoning, but has a low recall ratio and high complexity caused by the excessive strictness. By contrast, precision is sacrificed in the non-logic-based methods. However, they enable an easier implication of service matchmaking primarily based on text similarity, XML/RDF graph matching, and data mining. Some hybrid SSM methods have also been proposed to make full use of the advantages of basic SSM methods, e.g., OWLS-MX3\cite{7} and LOG4SWS\cite{8}. For these methods, the service matching process can be used to classify the matching results of all basic methods as a success or failure. However, the weight and threshold of the classification are manually set at present. Manually
setting thresholds is often very complicated and may lack evidence.

The disadvantages of common SSM methods are summarized in Table 1.

2.2 False-positive and false-negative

In SSM, there may be differences between user experience and the recommendation results from service matchers, which can lead to false-positive and false-negative issues.

**False-positive.** Based on the logic analysis on the concepts of input/output between the service and user demands, the SSM is confirmed a success. In fact, it is difficult to complete user demands with the provided service, which means that matchmaking is a failure. One of the reasons for this problem is that the concept in ontology cannot accurately capture the semantics of the real world, which is reflected in the difference in ontology granularities. For instance, a user needs to inquire for the price of a hybrid sedan, and the service matchmaker returns a query service for all car prices to the user due to a semantic deduction. However, the user believes that the true semantic distance between the “hybrid sedan” and “car” is too large. Thus, the service did not meet the needs from the user’s subjective judgement. This situation was noted in the semantic service retrieval test set in OWLS-TC[10]. Another reason is that many SSM methods match the input/output of services without considering the functional information contained in the preconditions and effects of the service, which can result in the failure of the service matching.

**False-negative.** False-negative means that the semantic service matcher affirms that a service matching the user service request does not, in fact, match. The main reason for this problem is the restrictions caused by the over-strict logical matching. In practice, there are some problems in ontology based on strict logical partitioning, and the concept of similar semantics may be completely exclusive in the ontology tree[11]. For example, debit and credit cards can both make payments. When a user requests to use a debit card to pay, it is often implicit that the user can also pay using a credit card. However, debit and credit cards are divided into two completely mutually exclusive concepts in the ontology tree by the property of an overdraft account, where a service matching the user service request is considered mismatched.

2.3 Logic-based SSM and approximate logic-based SSM

SSM based on a logical method usually uses deductive reasoning on service semantics to determine whether the semantics between services and user needs is equivalent and has reasonable logic containment relations in terms of the input/output. The principles of the process are as follows: (1) All inputs of semantic services should be matched by the input of the user service requests. Only in this way can the service request provide enough input information to ensure that the service will work normally. (2) All outputs of the service request should be matched by the output of the semantic service. Only in this way can the output information of the service be satisfied with the user requirements. Usually, the semantic service input/output function parameters are given in the representation of concepts in an ontology tree, and the preconditions and effects can be described using a first-order logic rule description language, such as Semantic Web Rule Language (SWRL) and Planning Domain Definition Language (PDDL). These conditions allow semantic services to support logical reasoning, so it is feasible and effective to use logical methods to match semantic services. However, such strict methods can cause the deterioration of false-negative problems, resulting in the low recall of service matching. In addition, in remote cloud systems, services may have heterogeneous preconditions and input/output descriptions, even if they still meet user needs. Therefore, an SSM method based on an approximate logic is proposed by introducing the logical concepts contraction and abduction[12], which alleviate the false-negative problem.

### Table 1 Comparison of common SSM methods.

| SSM method       | Advantage               | Disadvantage                                      |
|------------------|-------------------------|---------------------------------------------------|
| Logic-based      | Higher precision        | Lower recall                                      |
| Non-logic-based  | Higher recall           | The precision is not good in dealing with heterogeneous functions of services. |
| Hybrid SSM       | Poss advantages of all basic methods. | The weight and threshold of the classification are manually set. |

3 Feature Learning Classification of SSM Based on a Random Forest

To realize the combination of different SSM methods and avoid the disadvantages of the traditional fixed weight, the matching results obtained by the different
types of SSM methods are used as feature vectors of service matching, and an intelligent service matching classification method is realized using a random forest method. Therefore, the SSM problem is transformed into a two-classification problem.

3.1 Random forest method

The idea of a random forest method comes from the decision tree. Because of their ease of explanation and interpretation, decision tree methods can easily handle the interaction among features, and there is no need to worry about outliers or whether the data are linearly separable. However, decision tree methods suffer from an overfitting problem. To address this problem, considering the voting theory, when multiple classifiers are combined into a single classifier, the core idea of the random forest method is to generate multiple decision trees without a high classification accuracy, which allows all trees to make decisions by voting.

When a random forest is constructed, a decision tree is set up for each training subset, and the “forest” formed by several decision trees is generated. Each decision tree does not need pruning to prevent overfitting. When using random forests for classification, $K$ samples are extracted from the original training set according to the bootstrap sampling method. Then, a $K$-decision tree combination classification model is constructed based on these samples. Next, a test sample is classified according to the multiple decision subtrees that are randomly constructed. The results of each subtree are aggregated, and the majority voting method is used to derive the final output. That is, when the independent variable $X$ is given, every decision tree in the random forest has one vote to decide the optimal classification result\cite{13}. The whole process is shown in Fig. 1, where $D$ is the training set. It is divided into $K$ small-scale training sets, such as $D_1, D_2, \ldots, D_K$ through random sampling with replacement.

A random forest usually has better classification effects (slightly better than support vector machines), it has fast and scalable characteristics and avoids the drawbacks of a large amount of parameter-tuning, for example, support vector machines. As the stipulations of an agreement of outliers, such as 0-1 normalization, have no effect on the random forest, the data do not need to be preprocessed when the random forest is used. Therefore, the use of the random forest method is advantageous for different forms of attribute values in service matching feature state vectors\cite{14}.

For the random forest training process, we first randomly select 5% of the samples from the OWLS-TC4 test set as a training set. The OWLS-TC4 test set contains a large number of evaluated matching pairs of semantic services and user requests. We first run the previous SSM method on these matching pairs and combine the matching results of different methods into a 10-dimensional feature vector $X = (x_1, x_2, \ldots, x_{10})$, where $x_1, x_2, \ldots, x_5 \in \{0, 1\}$ represent the results of the SSM based on a strict logic. Each item corresponds to a result presented in Definition 1, whose values is 0 or 1, and only one item in all the five items can be 1. $x_6, x_7 \in [-1, 1]$ represent the results obtained by the SSM based on an approximate logic, each of which corresponds to one of the results presented in Definition 4. $x_8 \in \{0, 1\}$ denotes the result obtained by the precondition and effect logical plug-in matching. $x_9 \in \{0, 1\}$ represents the result of the similarity matching of the service description text. $x_{10} \in \{0, 1\}$ represents the results of the ontology structure similarity matching. $y \in \{-1, 1\}$ represents the results of the user evaluation of the SSM with user requests. Thus, the service matching feature state vector space $X \cdot y$ is formed. Definitions 1 and 4

![Decision process of the random forest.](image-url)
will be discussed in Section 4.2.

3.2 Service matching feature state space

To coordinate the matching results of the strict logical and approximate logical matching and to improve the training effect of feature data for the classifier, performing pruning on the vector spaces of the service matching feature state is necessary. The strict logical subsumption relation also satisfies the approximate logical subsumption, so an approximate logical matching is usually consistent with the strict logical matching judgment on the successful samples of service matching. Therefore, state-space trimming is mainly for the parts that are identified as failures based on the strict logical matching, and it avoids weakening the performance of the classifier by the contradiction between strict logical matching and approximate logical matching results. In accordance with the results of the matching based on the approximate logic and the real results of the Training Samples (TS), four subsets are separated from the training samples ($x_5 = 1$), which is a strict logical matching failure,

$$TS_1 = \{(x_1, \ldots, x_{10}, y) \in TS | y = 1 \land (x_6 \leq 0 \land x_7 \leq 0)\},$$

$$TS_2 = \{(x_1, \ldots, x_{10}, y) \in TS | y = 1 \land (x_6 > 0 \land x_7 > 0)\},$$

$$TS_3 = \{(x_1, \ldots, x_{10}, y) \in TS | y = -1 \land (x_6 \leq 0 \land x_7 \leq 0)\},$$

$$TS_4 = \{(x_1, \ldots, x_{10}, y) \in TS | y = -1 \land (x_6 > 0 \land x_7 > 0)\}$$

In subset $TS_1$, $y = 1$ means that the samples are matched by the user, but neither strict logical matching nor approximate logical matching correctly matches the result, which generates a false-negative. This will affect the training results of the classifiers, so it is necessary to directly prune $TS_1$ from the state space. In subsets $TS_2$ and $TS_3$, the approximate logical matching is used to obtain the correct matching result, and their eigenvalues should be rewarded. Of note, in $TS_2$, when the approximate logical matching results are motivated, the error results (mismatches) of the strict logical matching need to be modified ($x_5 = 0$). In subset $TS_4$, where $y = -1$, mismatched services and requests are treated as a matching success, but they do not affect the correctness of the strict logical matching. Therefore, the inverse weight adjustment of the value in the state space is needed to alleviate the influence of the false-positive on the training results. The adjustment method is as follows:

$$TS_2: \begin{cases} \text{IF } x_6 \geq x_7, \\ \text{THEN } x_6 = \omega_1 x_6, x_7 = 0; \\ \text{ELSE } x_6 = 0, x_7 = \omega_2 x_7, \end{cases}$$

$$TS_3: \begin{cases} \text{IF } x_6 \geq x_7, \\ \text{THEN } x_6 = \omega_3 x_6, x_7 = 0; \\ \text{ELSE } x_6 = 0, x_7 = \omega_4 x_7, \end{cases}$$

$$TS_4: \begin{cases} \text{IF } x_6 \geq x_7, \\ \text{THEN } x_6 = (1 - \omega_1) x_6, x_7 = 0; \\ \text{ELSE } x_6 = 0, x_7 = (1 - \omega_2) x_7 \end{cases} \tag{2}$$

where $\omega_1$–$\omega_6$ are weights which will be discussed in the following.

In the selection of incentive weights, inspired by Ref. [16] and based on the Bayesian posterior probability, we adjust the positive weights of the state-space value with the help of inference to the best explanation method. First, hypothesis $H_1$ represents the approximate logical matching and correctly identifies the services and requests that can be successfully matched ($i = 1, 2$ represents the two results of approximate logical matching), and explanation $E$ represents the services and requests that are actually successful matched in the training sets. The posterior probability is used to analyze the consistency between the hypothesis and explanation, which is used as the incentive weight. Accordingly, the training set’s state-space adjustment weight is set as

$$\omega_1 = P(H_1 | E) = \frac{P(E | H_1)P(H_1)}{P(E)} \tag{3}$$

where $P(E | H_1)$ represents the frequency of the matched sample that is identified as the approximate plug-in matching by the approximate logical matching. Similarly, $\omega_2 = P(H_2 | E)$, $\omega_3 = P(\neg H_1 | \neg E)$, and $\omega_4 = P(\neg H_2 | \neg E)$.

3.3 Four basic SSM methods

Four basic SSM methods are introduced in Sections 4 and 5. These methods include logic-based and non-logic-based methods, which have different advantages and can complement one another. The advantages of these methods are summarized in Table 2.

4 SSM based on the logical method

4.1 Strict logical matching method

Matching based on a strict logic is the most common SSM method at present, and most of the methods follow the principle ($\forall C \in I_S, \exists C' \in I_R : C \sqsubseteq C'$) and ($\forall D' \in
Hybrid Semantic Service Matchmaking Method Based on a Random Forest

Let $O_R, \exists D \in O_S : D \subseteq D'$, in which $C, C', D,$ and $D'$ are logical concepts in ontology. However, in some cases, such a principle can lead to the deterioration of the false-positive problem, such as the partial lack of services or request input and output concepts, as shown in Fig. 2.

There are two groups of customer service requests and semantic services in Fig. 2, where a user needs a service to buy books. However, by matching a criterion, a dating service that has different output with the same input is provided to the user, but this service is not related to the user request. Similarly, the matcher may also incorrectly match a service that has no input but has the same output as the service request and erroneously matches the user. The main reason for such a situation is that for all $C \in I_S, \exists C' \in I_R$ has established a subjective relation between the input of the semantic service concept and the request; therefore, the lack of definitions of service semantics, even key concepts in the service input, is wrongly tolerated, which also occurs on the output. To improve this situation, it is necessary to ensure that the mapping between input and output concepts is injective to guarantee that the key concept mapping is not missing.

The input/output of the semantic service and user request constituent a bipartite graph, in which the concepts are the nodes and the weights of the concepts in the ontology tree are the degrees of difference. Thus, the injective mapping between the concepts of the input and output is the maximum matching of the bipartite graph, and the matching points in the maximal allocation must completely cover all the inputs (outputs) of the semantic service. There are many mature algorithms for solving bipartite graph matching, which will not be discussed in detail here. Finally, the two nodes of the matching edge in the maximal matching are regarded as a concept pair, which will be added to the concept mapping set according to whether it is an input or output.

In addition, because it is inconvenient for the matcher to report the matching degree between services and requirements with a single matching criterion, according to the semantic service input/output bipartite graph maximum matching mentioned above, the matching results are defined as Exact, Plug-in, Subsumes, Subsumed-by, and Logical fail.

**Definition 1** Let $S$ and $R$ be the semantic service and user service request, respectively, and $I_S, I_R, O_S,$ and $O_R$ be the collections of semantic concepts of the inputs and outputs of semantic service $S$ and user service request $R$, which are defined on the same ontology tree. Considering the strict logic of SSM, we divide the matching degree of $S$ and $R$ into the following situations:

1. **Exact**
   
   $$(\forall C \in I_S, \exists C' \in I_R : (C, C') \in BCM(I_S, I_R)$$
   
   $$\land C \equiv C' \land (\forall D' \in O_R, \exists D \in O_S : (D, D')$$
   
   $$\in BCM(O_S, O_R) \land D \equiv D')$$

2. **Plug-in**
   
   $$(\forall C \in I_S, \exists C' \in I_R : (C, C') \in BCM(I_S, I_R)$$
   
   $$\land C \supseteq C' \land (\forall D' \in O_R, \exists D \in O_S : (D, D')$$
   
   $$\in BCM(O_S, O_R) \land D \subseteq D')$$

3. **Subsumes**
   
   $$(\forall C \in I_S, \exists C' \in I_R : (C, C') \in BCM(I_S, I_R)$$
   
   $$\land C \supseteq C' \land (\forall D' \in O_R, \exists D \in O_S : (D, D')$$
   
   $$\in BCM(O_S, O_R) \land D \subseteq D')$$

---

*Table 2 Comparison of the basic SSM methods*

| SSM method       | Advantage                              | Disadvantage                      |
|------------------|----------------------------------------|-----------------------------------|
| Strict logic-based | Higher precision.                       | Strict logical matching leads to a lower recall |
| Approximate logic-based | There is a greater likelihood of discovering potential matching services. | Lower precision |
| Text similarity  | Fast and easy to extend                | The ambiguity of the language will exacerbate the false-positive phenomenon. |
| Ontology structure similarity | Fast | Precision depends on the optimization of the ontology tree. |

---

Fig. 2 False-positive problem in service matching.
(4) Subsumed-by
\[(\forall C \in I_S, \exists C' \in I_R : (C, C') \in BCM(I_S, I_R) \land C \sqsupseteq C') \land (\forall D' \in O_R, \exists D \in O_S : (D, D') \in BCM(O_S, O_R) \land D \sqsubseteq D')\]

(5) Logical fail. When there is no matching relationship between S and R, the strict logical match between S and R is considered a logical failure.

where BCM stands for bipartite concept mapping set.

Both the Plug-in and Subsumes matching are concepts of the semantic service output, which comprise the concept of a user service request output. The difference is that the concept of the service output in Plug-in is a direct sub-node of the user request output concept in an ontology tree, so the output of the service is closer to the expected output of the user in the true semantic distance. In addition, Subsumed-by matching is achieved by relaxing restrictions on some outputs because more generalized outputs may partially meet user needs, but the matching degree is obviously inferior to that of Plug-in and Subsumes. This generalization must be limited on the direct parent node in the service request output concept, which prevents a false-positive caused by excessive generalization. Moreover, this generalization cannot be performed on the input because the input concept generalization of user requests for a service may not satisfy the input constraints, leading to a service failure. In summary, the matching degree of the five results is sorted according to the degree of semantic similarity: Exact > Plug-in > Subsumes > Subsumed-by > Logically fail.

4.2 Approximate logical matching method

An SSM method based on strict logical reasons judges the subsumption relations between input and output ontology concepts with formal methods. It is overly strict when describing user needs with strong fuzziness and may lead to the excessive exclusion of compatible services and result in a false-negative. An example from OWLS-TC4 is demonstrated in Fig. 3.

In Fig. 3, the user proposes a service request R, hoping to buy a book with a debit card and receive confirmation information after the purchase is successful. An e-commerce service S can sell all kinds of documents (including books) to users with credit cards and provides users with receipts and pricing information. With the method described in Section 4.1, the result of Logical fail will be returned. However, in OWLS-TC4, the user believes that the service and request are matched; thus, a false-negative occurs. Here, although books and documents belong to different branches of objects in the ontology tree and are not completely logical exclusive, credit and debit cards form a strict logical exclusion because of the property of whether or not they can overdraw. Klusch and Kapahnke[15] found that if the strict logical matching method was applied in OWLS-TC4, approximately 45% of the services in each service request set would be incorrectly categorized as mismatches. In fact, if the user could give up the concepts that cause the logical subsumption failure and only keep part of the concepts for approximately calculating subsumption relations, the approximate logical matching degree between the semantic service and user request could be determined and the erroneous judgement of a strict logical matching result could be compensated.

Inspired by Ref. [12], the logical concepts contraction constriction and abduction are introduced when calculating an approximate logical matching degree.
The compatible parts Concept Compatible (CC) and the incompatible part Concept Incompatible (CI) of concepts $C$ and $D$ can be found by a logical concept constriction, which abandons some property constraints of the concepts. The structural abduction collection SAC ($C, D$) of $C$ relative to $D$ can be obtained by performing abduction on the compatible part CC, and the approximate concept $C'$ of $C$ relative to $D$ can be obtained from the mapping $\sigma[\overrightarrow{H}, \overrightarrow{H}](CC^h)$, where $CC^h$ is an abduction concept that will be discussed latter. By computing the information quantity differences among $C$, $C'$, and $D$, the approximate quantity of information can be calculated, based on which the approximate logical matching degree between the service and request can be computed.

**Definition 2 (Logical concepts constriction and abduction)** The constriction of concept $C$ relative to concept $D$ is

$$LCC (C, D) = (CI, CC)$$

where $CC$ represents the compatible part of $C$ relative to $D$ and $CI$ denotes the incompatible part of $C$ relative to $D$. Abduction concept $CC^h$ is derived from $C$ by the following expansion:

$$CC^h = h_0 \cap \text{rew}(k)$$

where $\text{rew}(A) = A$, $\text{rew}(\neg A) = \neg A$, $\text{rew}(C \cap D) = \text{rew}(D) \cap \text{rew}(D)$, and $\text{rew}(\exists R.C) = \exists R. (h_i \cap \text{rew}(C))$. $A$ is the atomic concept after expanding CC in the ontology tree. The structural abduction of $CC$ relative to $D$ is expressed as

$$SAC (C, D) = H = (H_0, \ldots, H_n)$$

where $\sigma[\overrightarrow{H}, \overrightarrow{H}](CC^h) \subseteq D$ and $\sigma[\overrightarrow{H}, \overrightarrow{H}](CC^h) \subseteq D$. The approximate concept of $C$ relative to $D$ is denoted as

$$C' = \sigma[\overrightarrow{H}, \overrightarrow{H}](CC^h)$$

where $\sigma[\overrightarrow{H}, \overrightarrow{H}]=\{h_0 \mapsto H_0, \ldots, h_n \mapsto H_n\}$. The definition of the approximate logical concept subsumption can be obtained from the Definitions 1 and 2 and calculation method above.

**Definition 3 (Approximate logical concept subsumption)** Concept $C$ is an approximate logical concept subsumed by concept $D$ if and only if the approximate concept of $C$ relative to $D$ is contained by the $D$ logically,

$$\sigma[\overrightarrow{H}, \overrightarrow{H}](CC^h) \subseteq D$$

where $C' = \sigma[\overrightarrow{H}, \overrightarrow{H}](CC^h)$, $(CI, CC) = LCC(C, D)$, and $H = SAC(CC, C, D)$.

The following concepts of debit and credit cards are taken as examples to illustrate the derivation process of the approximate logical concept subsumption. Suppose that the user agrees to relax restrictions on the debit card due to the absence of an overdraft, then

$$CI, CC = LCC(\text{DebitCard, CreditCard}) = \{\neg \text{allows.Credit}^p, \text{MediumOfExchange} \land \text{IssuedBy.Bank}^p\}$$

From Definition 2, $CC^h = h_0 \cap \text{Object}^p \land \exists \text{HasValue}. (h_1 \cap \text{Value}^p) \land \text{IssuedBy}. (h_2 \land \text{Bank}^p)$. From the algorithm given in Ref. [2], we can obtain $\overrightarrow{H} = (h_0, h_1, h_2)$, $H = SAC(\text{DebitCard, CreditCard}) = (\exists \text{allows.Credit}^p, \bot, \text{Company}^p)$. Then, $\sigma[\overrightarrow{H}, \overrightarrow{H}] = \{h_0 \mapsto \text{allows.Credit}^p, h_1 \mapsto \bot, h_2 \mapsto \text{Company}^p\}$. Thus, the approximate concept of debit card is $\text{DebitCard}' = \exists \text{allows.Credit} \land \text{MediumOfExchange} \land \text{IssuedBy}. (\text{Bank} \land \text{Company})$, obviously, $\text{DebitCard}' \subseteq_{AC} \text{DebitCard}$ is valid.

To evaluate the matching degree of the semantic service based on the approximate logic, the approximate logical subsumption degree between concepts is calculated by introducing the information quantity[16] to analyze the loss of information caused by the concept contraction during an approximate concept construction. The concept approximate calculation formula based on information quantity is given as follows:

$$\text{Sim}_{\text{info}} (C, D) = \frac{2 \times \text{IC}(LCC(C, D))}{\text{IC}(C) + \text{IC}(D)}$$

The information quantity $\text{IC}(C)[17]$ is obtained by the probability of the concept $C$:

$$\text{IC}(C) = -\log P(C), \quad P(C) = \frac{n(C)}{N}$$

where $n(C)$ is the number of occurrences of all sub-concepts contained in the ontology tree of the concept $C$. $N$ is the total number of concepts in the ontology. Clearly, $\text{Sim}_{\text{info}} (C, D) \in [0, 1]$. Accordingly, the approximate logical subsumption degree can be calculated as

$$\text{ALSD}(C, D) = \text{Sim}_{\text{info}} (C', D) - (1 - \text{Sim}_{\text{info}}(C', C))$$

Because $\text{Sim}_{\text{info}} (C, D) \in [0, 1]$, we have $\text{ALSD}(C,
Because concept $C'$, which is an approximate concept derived from logical concept constriction and abduction of $C$, has a direct subsumption relation with concept $D$, the subsumption relation formed in the ontology tree must be a direct parent-child relationship. Thus, we propose two assumptions on approximate logical matching, which are approximate Plug-in matching and approximate Subsumed-by matching, respectively.

Definition 4 (SSM based on an approximate logic) Let $S$ and $R$ be the semantic service and user service request, respectively; then, $I_S$, $I_R$, $O_S$, and $O_R$ are the sets of the input and output semantic concepts of semantic service $S$ and user service request $R$ defined on the same ontology tree. SSM based on an approximate logic divides the matching degree of $S$ and $R$ into the following cases:

1. Approximate Plug-in

\[
\forall C \in I_S, \exists C' \in I_R : (C, C') \in BCM(I_S, I_R) \land C \sqsubseteq_{AC} C' \land (\forall D' \in O_R, \exists D \in O_S : (D, D') \inBCM(O_S, O_R) \land D \sqsubseteq_{AC} D')
\]

(17)

2. Approximate Subsumed-by

\[
\forall C \in I_S, \exists C' \in I_R : (C, C') \in BCM(I_S, I_R) \land C \sqsubseteq_{AC} C' \land (\forall D' \in O_R, \exists D \in O_S : (D, D') \in BCM(O_S, O_R) \land D \sqsubseteq_{AC} D')
\]

(18)

By calculating the approximate logical subsumption degree, the approximate logical matching degrees under all kinds of conditions of service approximate logical matching can be obtained,

\[
\text{MatchAL}(S, R) = 1 - \left( \frac{\sum\text{ALSD}(C, C')}{|I_S|} + \frac{\sum\text{ALSD}(D, D')}{|O_R|} \right)
\]

(19)

Because approximate logical subsumption relation is uncertain due to different concept contractions, semantic services and user requests are likely to conform to both approximate Plug-in matching and approximate Subsumed-by matching. Therefore, it is necessary to calculate both degrees of service matching under different approximate logical matching. In a case where the matching degree of service is higher and the value is positive, the matching will be considered a match between the semantic service and user request. If the results of $\text{ALSD}(C, D)$ are less than or equal to 0 in the two matching cases, the semantic service and user request are considered not matched on the approximate logic.

4.3 Precondition and effect of logical Plug-in matching

The input and output of a service cannot reflect its functional semantics, and this part is usually expressed in the form of a precondition and effect of service logic. In addition to a semantic explanation on services using ontology, SWRL, PDDL, and other languages, which present rules in a semantic way, are also used to describe service preconditions and effects.

The use of logical matching alone on the concepts of service input and output will cause a false-positive problem. Therefore, we propose the precondition and effect of logical Plug-in matching of semantic services by introducing the method used in LARKS[18].

The following principle is inherited from the software retrieval domain: logical specification plug-in matching.

Definition 5 (Precondition and effect of logical Plug-in matching) Semantic service and user service request are the preconditions and effects of logical Plug-in matching if and only if the precondition of the user request logically contains the precondition of the semantic service and the effect of the semantic service logically implies the effect of the user request, which is

\[
\text{MatchPE}(S, R) = (P_R \Rightarrow P_S) \land (E_S \Rightarrow E_R)
\]

(20)

The logical implication relation between a precondition and effect can be determined by the concept of $\theta$-inclusion proposed by Ref. [19], which is

\[
(\forall e \in E_R \exists e_S \in E_S : e_S \sqsubseteq_{\theta} e_R) \Rightarrow (E_S \Rightarrow E_R)
\]

(21)

5 SSM Based on a Non-Logical Method

Service signature contains not only hasInput, hasOutput, Precondition, Effect (IOPE), and other functional parameters, but also many non-functional parameters, such as serviceName, serviceCategory, qualityRating, textDescription, and the metadata of the name and location of a service provider. Comparing the degree of similarity between the semantic service and information requested by the user when performing text similarity matching on the service descriptions is intuitive. Moreover, service signature can retrieve the possible matches that are deemed mismatched by the logic-based method, which may help to alleviate the false-negative problem.

5.1 Service description text similarity matching

When using service description text similarity matching,
the service description textDescription is usually modeled by the text vector space model. To avoid the high-dimensional sparse problem caused by the increase of text amount, we first need to reduce the dimension using natural language processing methods, such as word segmentation and stop word filtering. Then, based on Term Frequency-Inverse Document Frequency (TF-IDF), the service description text similarity is calculated. Next, the weights of text keywords in the service and user request are calculated, and then the distance between feature vectors can be obtained based on the cosine theorem. The TF-IDF formula used in this paper is as follows:

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}},$$

$$IDF_{i,j} = \log \frac{|D|}{|\{j : w_i \in d_j\}|},$$

$$TF-IDF_{i,j} = TF_{i,j} \times IDF_{i,j} \tag{22}$$

After deriving the TF-IDF weights, we can calculate $Sim_{description}(S, R)$, which is the similarity of the service description texts by the cosine theorem.

Due to the lack of uniform standards, service providers and users may have different representations of the service description language, so computing the text similarity of the service description may merely exist in the low-precision problems. Considering the text matching of the service description, we introduce the expanded concept of the service input and output and propose an extended text similarity matching method for the service description.

**Definition 6 (Extended concept expression)**
An extended concept expression is the sequential splicing text that consists of all intermediate concept nodes from the root node to the current node of the ontology tree. The ontology tree in Fig. 3 is taken as an example, and according to Definition 6, the extended concept expression of CreditCard is andCreditCard andMedium OfExchange Object. When the extended concept expression is used to calculate the similarity of a text, loss of information based on similarity is introduced. The loss information of the input and output (LOI($I_S, I_R$) and LOI($O_S, O_R$)) between the services and user requests in the expanded concept expression can be obtained to measure the difference of the expanded concept expression between services and user requests, respectively,

$$LOI(A, B) = 1 - \frac{|A \cup B| - |A \cap B|}{|A| + |B|} \tag{23}$$

where $LOI(A, B) \in [0, 1]$, so the greater the value, the higher the similarity. After calculating the text similarity of the extended concept expression, the extended service description text similarity can be obtained by the following formula:

$$Sim_{ext}(S, R) = Sim_{description}(S, R) + \frac{LOI(I_S, I_R) + LOI(O_S, O_R)}{\epsilon_{Dist(C, D)}} \tag{24}$$

### 5.2 Ontology structure similarity matching
The core idea of the ontology matching method is mapping the information of semantic services and the user request to the ontology. Then, the similarity of the whole semantic is obtained to realize the SSM by calculating the similarity of the corresponding nodes among ontologies. In general, the similarity of ontology structures can be measured from the semantic distance, depth, and information quantity.

Semantic distance refers to the length of the shortest path of the two concepts to be compared in the ontology tree through its nearest common ancestor node: the greater the semantic distance, the greater the semantic difference of the concept. The similarity computation method of the semantic distance is as follows:

$$Sim_{dist}(C, D) = \frac{1}{\epsilon_{Dist(C, D)}} \tag{25}$$

where $Dist(C, D)$ represents the length of the shortest path of between concepts $C$ and $D$ bypassing their nearest common ancestor node.

Depth is also an important indicator for measuring similarity. With the increase of the depth of the Nearest Common Ancestor (NCA) nodes of the two concepts, the number of the common attributes of concepts and the semantic similarity increase. In this paper, the depth similarity is defined as the ratio of the depth of the nearest common ancestor nodes to the maximum depth of the tree:

$$Sim_{depth}(C, D) = \frac{\text{Depth(NCA}(C, D))}{\text{Depth(Tree)}} \tag{26}$$

where $\text{NCA}(C, D)$ represents the nearest common ancestor nodes of $C$ and $D$ and $\text{Depth(Tree)}$ represents the maximum depth of the ontology tree.

In addition, the quantity of information can measure the semantic information size of a concept from the perspective of probability, which helps analyze the similarity between concepts. The calculation formula of similarity based on the information quantity has been given in Section 4.2.

By integrating the semantic distance, depth, and information quantity, we can calculate the similarity of
the ontology structure between two concepts or between two concept sets $c$ and $d$, 

$$\text{Sim}_{\text{conc}}(c, d) = \text{Sim}_{\text{dist}}(c, d) + \text{Sim}_{\text{depth}}(c, d) + \text{Sim}_{\text{info}}(c, d)$$  \hspace{1cm} (27) 

$$\text{Sim}_{\text{concSet}}(C, D) = \frac{1}{|C|} \sum_{c \in C} \max_{d \in D} \text{Sim}_{\text{conc}}(c, d)$$  \hspace{1cm} (28)

Moreover, the ontology structure similarity between the semantic service and user request can be obtained as follows:

$$\text{MatchNL}(S, R) = \frac{1}{2} (\text{Sim}_{\text{concSet}}(I_S, I_R) + \text{Sim}_{\text{concSet}}(O_S, O_R))$$  \hspace{1cm} (29)

6 Simulation Experiment

To evaluate the proposed hybrid SSM method, a public service discovery test set is used to build a simulation environment for service discovery and matching. Then, the performance and effectiveness of our method are compared with those of the four basic methods through a series of classic metrics.

6.1 Simulation scenario setting

Because there is no standard test set for OWL-S service discovery and matching, in this study, we use the service discovery test data set OWLS-TC4 to construct the simulation experiment scenes. OWLS-TC4[21] is the fourth version of the OWL-S service test set developed by Klusch et al.[20], and the purpose of this test set is to support the performance evaluation of the SSM algorithm based on OWL-S description. OWLS-TC4 provides 1083 semantic services described in OWL-S 1.1, which encompass nine different areas: education, healthcare, food, travel, communications, economics, weapons, geography, and simulation. OWLS-TC4 also provides a set of 42 test requests for performance evaluation tests. The service part of OWL-TC4 comes from the public IBM UDDI registration center and has been improved and expanded by many organizations and research and development personnel to realize semi-automatic conversion from WSDL to OWL-S. The simulation experiment in this study selects three domain services and requests from OWLS-TC4 as the experimental data set, as shown in Table 3.

| Field     | Number of services | Number of requests |
|-----------|--------------------|--------------------|
| Tourism   | 197                | 6                  |
| Economy   | 395                | 12                 |
| Education | 286                | 6                  |

The hardware environment used in the experiment is Intel Xeon(TM) E3-1230 v2, 3.30 GHz CPU, 16 GB RAM PC. The operation system is 64-bit Windows 7 SP1, and XAMPP is configured to run the OWLS-TC4 test set. The ontology set used is based on the semantic dictionary WordNet designed by Princeton University. The service matching simulation experiment is based on the S2M2 framework, and Java is used to program the semantic service and request analysis. In addition, the random forest method adopted in this experiment is programmed by MATLAB.

Precision and recall are selected as the two core metrics for the simulation experiments. Precision refers to the ratio of true positive matches in the results obtained by the service matchmaking to all the service in the set, and recall is the ratio of true positive matches in the results obtained by the service matching method to all the matching services specified by the test set. With the selected metrics, the hybrid SSM method proposed is compared with four classic SSM methods, namely, typical logical matching, approximate logical matching, ontology structure matching, and service text matching algorithm. To eliminate the influence of experimental errors, simulation experiments are performed 30 times. The same service request is used for various service matching methods in each experiment, and the average precision and recall rate of 30 experiments under different algorithms are finally calculated.

6.2 Result analysis

6.2.1 Effectiveness of the approximate logic-based SSM method

We first analyze the effectiveness of the approximate logic-based SSM method. For brevity, in the following part of this paper, the SSM method based on the approximate logic is denoted as the Alogic-based SSM method. Because both the Alogic-based SSM method and the strict logic-based SSM method belong to the category of semantic logic reasoning, which has a higher recall than non-logic-based SSM method, only the precision is analyzed. In Fig. 4, the precision of the strict logical semantic matching, approximate logical semantic matching, and precondition and effect of logical Plug-in matching is compared in different fields. As shown in Fig. 4, the result of the strict logical semantic matching is commonly low because it is difficult for strict logical matchmaking to handle special cases, such as the absence of a service input and output. The introduction of the precondition and
effect of logical Plug-in matching solves the false-positive problem, effectively improving the precision of the service. Compared with that in the economic and educational fields, the semantic terminology in the travel field is less but is more freely expressed, so the input and output of the service cannot reflect its functional semantics. If only a logical matching on the concept of the service input and output is performed, it will lead to the false-positive problem, where the matching service function does not really satisfy the user’s requirement. In subsequent experiments, the precondition and effect of logical Plug-in matching will apply to all SSM methods by default if there is no special explanation.

6.2.2 Precision of different SSM methods

Figure 5 shows the precision performance of the proposed algorithm and three typical basic SSM algorithms in different areas of the test set. Figure 5 shows that the results obtained by the service matching method based on the service description text similarity are better than those of the other two algorithms in the precision rate, and the method based on the ontology structure similarity performs the worst. The ontology structure similarity and logic-based matching method are very dependent on the optimization degree of the ontology tree construction. These methods are likely to put the services with large distance in concept ontology into the matching service set, which leads to a false-positive, affecting the precision of service matching. Although service text matching can improve the precision based on the similarity of the service description text, it is likely to lead to a false-positive issue where the service function is not related to the user request due to language ambiguity. The false-positive can lead to subsequent difficulty for the service combination to fully meet the user’s business needs, resulting in low user satisfaction and seriously threatening the level of service credibility. By synergistically analyzing the results from various algorithms and conducting a self-learning classification, the hybrid matching method proposed in this paper can alleviate the false-positive problem in the service text matching method after incorporating the precondition and effect of logical Plug-in matching, improving the precision performance of service matching.

6.2.3 Recall of different SSM methods

Figure 6 shows the recall performance of the proposed algorithm and the other four typical algorithms in different fields. Due to the over-strict constraint from strict logical matching methods, many services that can be identified as matching in the real world are judged as mismatches by strict logical reasoning, resulting in a poor recall performance based on the strict logical matching method. By contrast, the method
based on the approximate logical matching realizes the relaxation of the ontology concept through the logical concepts contraction and abduction. As a result, more potential matching services can be found, which improves the recall performance. Moreover, methods, such as ontology structure matching and service text matching, can alleviate the false-negative issue caused by strict logical matching to some extent. The proposed method further improves the performance of the service matching recall by synthesizing the characteristics of various algorithms and using the advantages of the random forest method in the classification learning model.

6.2.4 Analysis of response time
Response time is another key metric of service matching. A quick response is helpful for improving the user experience, especially for systems with potentially large numbers of services and requests, such as cloud computing systems. The response time of the proposed hybrid SSM based on a random forest is compared with other basic SSM methods in Fig. 7. The text similarity SSM method and ontology structure similarity method have shorter response times, whereas the logic-based and the proposed hybrid method have longer response time. The reason for the longest response time of the proposed SSM method is determined by the principle of the logical matching method, which needs more time to finish semantic reasoning. Nevertheless, the response time is still within a reasonable range.

6.2.5 Effectiveness of state-space prune
Figure 8 compares the impact of state-space prune on the average precision of service matching, where the average precision is the mean of all precision results in the test set. The state-space prune based on testimony consistency has a significant improvement on the precision of random forest self-learning classification, which illustrates the effectiveness of the state-space prune.

The above experiments show that it is easy to find a special phenomenon in which the precision of the strict logical matching method is different from that of the approximate logical matching in different fields of the test sets. In the travel field, with a low level of recall, the precision performance of the strict logic-based method is better than that of the Alogic-based matching, but the result is opposite in the economics field with a level of high recall. This result also shows that there are defects in using any single type of the service matching method, and ensuring good service matching results under all conditions is difficult. Therefore, a hybrid approach, which combines different types of SSM methods, is the better solution for service matchmaking.

7 Conclusion
A hybrid SSM method is proposed in this paper based on random forest feature classification, addressing false-positives and false-negatives in SSM. The random forest method uses the results obtained by different matching methods as the eigenvalues to classify, that is, converting service matching into a classification problem, which avoids manually setting the threshold and the lack of sufficient evidence. The simulation experiments show that the proposed method has certain advantages in terms of precision and recall. We will further study the equilibrium problem of recall and precision in future and add more alternative algorithms to improve the service matching ability of random forests.

Acknowledgment
The work was supported by the National Natural Science Foundation of China (Nos. 61872104 and 61502118), the National Science and Technology Major Project of China (No. 2016ZX03001023-005), and the Natural Science Foundation of Heilongjiang Province in China (No. F2016009).
Wei Jiang et al.: Hybrid Semantic Service Matchmaking Method Based on a Random Forest

References

[1] Y. Liu, F. Du, J. Sun, Y. Jiang, J. He, and T. Zhu, A crowdsourcing-based topic model for service matchmaking in internet of things, Future Generation Computer Systems, vol. 87, pp. 186–197, 2018.

[2] A. Afzal, B. Shafiq, S. Shamail, A. Elahraf, J. Vaidya, and N. R. Adam, Assemble: Attribute, structure and semantics based service mapping approach for collaborative business process development, IEEE Transactions on Services Computing, vol. 10, no. 2, pp. 1–14, 2018.

[3] J. U. Hassan, P. Wen, P. Wang, Q. Zhang, F. Saleem, and M. U. Nisar, Dynamic model for service composition and optimal selection in cloud manufacturing environment, in Proceedings of Recent Advances in Intelligent Manufacturing, Chongqing, China, 2018, pp. 50–60.

[4] B. D. Martin, A. Esposito, and G. Cretella, Semantic representation of pattern services and services with automated reasoning to support cloud application portability, IEEE Transactions on Cloud Computing, vol. 5, no. 4, pp. 765–779, 2017.

[5] I. Dongo, Y. Cardinale, F. Alkhalil, and R. Chbeir, Semantic web datatype inference: Towards better RDF matching, in Proceedings of International Conference on Web Information Systems Engineering, Moscow, Russia, 2017, pp. 57–74.

[6] J. Leng and P. Jiang, Mining and matching relationships from interaction contexts in a social manufacturing paradigm, IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, no. 2, pp. 276–288, 2017.

[7] M. Klusch and P. Kapahnke, Adaptive signature-based semantic selection of services with OWLS-MX3, Multiagent and Grid Systems, vol. 8, no.1, pp. 69–82, 2012.

[8] U. Lampe and S. Schulte, Self-adaptive semantic matchmaking using COV4SWS, KOM and LOG4SWS. KOM, in Semantic Web Services, Berlin, Germany: Springer Science and Business Media Press, 2012, pp. 141–157.

[9] M. L. Sbodio, D. Martin, and C. Moulin, Discovering semantic web services using SPARQL and intelligent agents, Web Semantics: Science, Services and Agents on the World Wide Web, vol. 8, no. 4, pp. 310–328, 2010.

[10] M. Klusch, B. Fries, M. Khalid, and K. P. Sycara, OWLS-MX: Hybrid OWL-S service matchmaking, in Proceedings of AAAI Fall Symposium: Agents and the Semantic Web, Washington, DC, USA, 2005, pp. 77–84.

[11] W. Wei and J. A. Gulla, Sentiment learning on product reviews via sentiment ontology tree, in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 2010, pp. 404–413.

[12] T. D. Noia, E. D. Sciascio, and F. M. Donini, A tableau-based calculus for abduction in expressive description logics: Preliminary results, in Proceedings of the 22nd International Workshop on Description Logics, Oxford, UK, 2009, pp. 1–13.

[13] C. Lindner, P. A. Bromiley, M. C. Ionita, and T. F. Cootes, Robust and accurate shape model matching using random forest regression-voting, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1862–1874, 2015.

[14] R. Caruana, N. Karampatziakis, and A. Yessenalina, An empirical evaluation of supervised learning in high dimensions, in Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 2008, pp. 96–103.

[15] M. Klusch and P. Kapahnke, The iSeM matchmaker: A flexible approach for adaptive semantic service selection, Journal of Web Semantics, vol. 15, no. 6, pp. 1–14, 2012.

[16] D. Lin, An information-theoretic definition of similarity, in Proceedings of International Conference on Machine Learning, Madison, WI, USA, 1998, pp. 296–304.

[17] H. Amiri, P. Resnik, J. Boyd-Graber, and H. Daume III, Learning text pair similarity with context-sensitive autoencoders, in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1882–1892.

[18] K. Sycara, S. Widoff, M. Klusch, and J. Lu, Dynamic matchmaking among heterogeneous software agents in cyberspace, Autonomous Agents and Multi-Agent Systems, vol. 5, no. 2, pp. 173–203, 2002.

[19] P. Idestam-Almquist, Generalization of clauses under implication, Journal of Artificial Intelligence Research, vol. 3, pp. 467–489, 1995.

[20] M. Klusch, B. Fries, and K. Sycara, OWLS-MX: A hybrid semantic web service matchmaking for OWL-S services, Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, no. 2, pp. 121–133, 2009.

[21] D. H. Glass, Inference to the best explanation: A comparison of approaches, in Proceedings of the AISB 2009 Convention, Edinburgh, UK, 2009, pp. 1–10.

Wei Jiang received the BEng degree from Harbin Institute of Technology in 1996, and the MEng degree from Heilongjiang University in 2010. Now, he is engaged in teaching and researching as an associate professor and a master supervisor at Harbin Normal University. He is a member of China Computer Federation. His current research interests involve network security and autonomic computing.

Junyu Lin received the BS, MS, and PhD degrees from Harbin Engineering University in 2002, 2005, and 2009, respectively. Now, he is an associate researcher and researching at mobile data offloading in cellular networks and optimization. He is a full stuff at Institute of Information Engineering, Chinese Academy of Sciences. His research interests involve edge computing, LTE (LTE-A) networks, and wireless channel access control.
Huiqiang Wang received the MS and PhD degrees from Harbin Engineering University (HEU) in 1985 and 2005, respectively. From 2001 to 2002, he was at the Queen’s University, Ontario, Canada, as a senior visiting scholar. Now, he is engaged in teaching and researching as a professor and a doctoral advisor at HEU. Up to now, he holds ten Chinese patents. His research interests include network security, cognitive networks, and autonomic computing.

Shichen Zou received the PhD degree from Harbin Engineering University, in China in 2018. Now he is a full staff at Huawei Nanjing Research Institute, and his research interests include network security and cognitive networks.