Identify the Semantic Meaning of Service Rules with Natural Language Processing

Xinfeng Ye
Department of Computer Science
Auckland University
Auckland, New Zealand
xinfeng@cs.auckland.ac.nz

***Abstract*—In cloud manufacturing, manufacturing resources are services that can be looked up and accessed over the Internet. Manufacturing ontologies are used to store the service information. Manufacturers use service rules to control the access to their resources. The rules are normally written in natural language. Thus, they need to be converted to semantic rules that can be understood by the search engine of the manufacturing ontologies. Our previous work investigated converting service rules to semantic rules automatically. However, the scheme is not flexible enough. This paper proposed an improvement to the scheme in our previous work. The proposed scheme allows a wider range of service rules to be converted to semantic rules accurately.**

Keywords- ontology, natural language processing

I. INTRODUCTION

In cloud manufacturing, the service providers encapsulate their manufacturing resources into consumable services that can be looked up and accessed over the Internet. Customers query the cloud manufacturing platform for services. Service providers usually specify some service rules that control the access to their services. The service rules are normally written in natural language by manufacturing engineers. As many cloud manufacturing platforms are powered by semantic web techniques [9, 10], queries are handled by inference engines over RDF graphs. Thus, the service rules have to be converted to semantic rules that can be understood by inference engines. In our previous work [3], we developed a scheme that automatically converts service rules to semantic rules. However, the scheme is not flexible enough to handle a variety of service rules. To address the issue in [3], this paper proposed an improvement to [3].

The main contributions of the paper are: (a) it classifies the typical service rules into six patterns, and (b) it showed that, by using the classification and natural language processing techniques, the service providers, client, and resources in service rules can be identified more accurately.

The rest of this paper is organized as follow. §II introduces the technologies used in the paper and our previous work. §III, and §IV present the details of the techniques for identifying the service providers, the client, and the semantics of service rules. §V discusses generating semantic rules. The evaluation of the scheme and the related work are described in §VI and §VII respectively. Conclusions are given in §VIII.

II. RELEVANT TECHNIQUES AND MOTIVATIONS

A. RDF and Jena Rules

Resource Description Framework (RDF) is the method for modeling data in semantic web [17]. The core concepts in RDF include node, link, triple, and graph. An RDF node represents any information. An RDF link describes the relations between two nodes [14]. An RDF link with its two ends (nodes) forms an RDF triple. An RDF triple has the form (*subject*, *property*, *object*). *subject* and *object* are RDF nodes. *property* represents the relationship between *subject* and *object*. A triple can be read as "the *subject* has a *property* with the value being the *object*". A set of RDF triples form an RDF graph. An ontology defines a set of concepts and the relations between the concepts. RDF graphs are used to store the data of ontologies. Semantic rules specify the constraints on data or the relationships between data.

```
// conclusion: ?customer can access ?resource
[(?client  mc:hasAccessTo  ?resource) ←
    // premise: The name of ?provider is 'Company X'.
    (?provider  mc:name  'Company X'),
    // premise: The type of ?resource is
    // mc:machine.
    (?resource  rdf:type  mc:machine),
    // premise: ?client is located in Auckland.
    (?client  mc:hasAddress  'Auckland')]
```

Figure 1 A Jena Rule Example

Apache Jena framework has been widely used in semantic web applications [1]. Jena rules are the built-in semantic rules of the framework. Jena rules work on RDF graphs. A Jena rule consists of two parts, i.e. *head* and *body*. The head and the body consist of triples of the form (*?subject ?property ?object*). The "*?*" prefix means the entity is a variable. The meaning of a triple is the same as an RDF triple. A head has one triple called *conclusion* while the body can have one or several triples called *premises*. All premises must match the triples in the RDF graph and each variable in the premises must be bound to an item. When all the premises match the triples in the RDF graph, the conclusion is asserted and the variables in the conclusion have the values that are bound in the premises. For example, Figure 1 shows the Jena rule that

corresponds to service rule "Company X shares machines with Auckland-based companies.". The meaning of the conclusion and premises are given in the highlighted text. The labels with prefixes "*mc*" and "*rdf*" have been defined in the ontology in [9]. If all the premises can be matched with the triples in the RDF graph of the ontology, the conclusion is asserted. That is, the company bound to *?client* can access the resource bound to *?resource*.

B. Natural Language Processing (NLP)

An English sentence consists of a *complete subject* and a *complete predicate*. The complete subject tells whom or what the sentence is about. The complete predicate tells what the subject is or does. A sentence might have an *object* that is a noun in the complete predicate. The object receives the action of the verb in the predicate. A sentence might also has an *infinitive phrase* that acts as an adverb to modify the verb in the predicate. For example, consider sentence "Company A allows Company B to use Company A's machine". In this sentence, "Company A" is the subject, "allows Company B to use Company A's machine" is the complete predicate, "to use Company A's machine" is the infinitive phrase which is used as an adverb of the verb "allows".

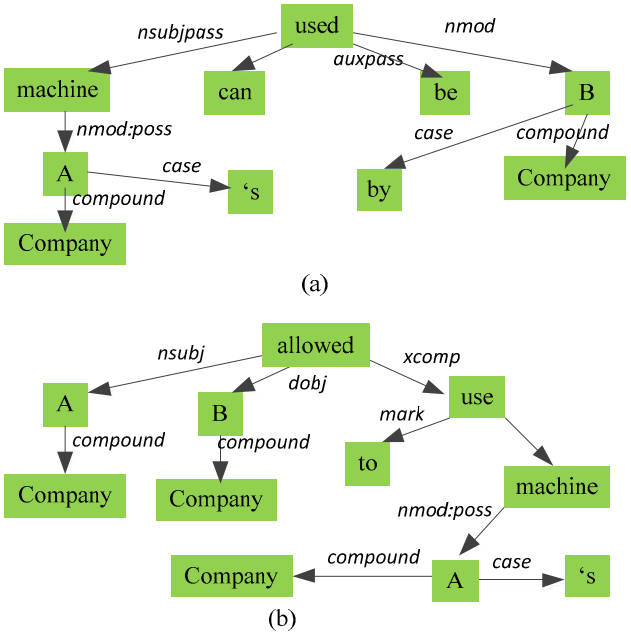

Figure 2 Dependency Tree

Natural language processing (NLP) is a range of techniques that enable computers to analyse and to understand human languages [2, 4]. Universal dependency has been widely used in NLP tasks [5]. It labels the dependencies (i.e., grammatical relations) between individual words [7]. Dependencies are triplets of the form: *dependency_relation*(*governor, dependent*) [16]. The dependencies in a sentence can be represented as a tree. In the tree, an arrow is drawn from the governor to its dependent. The dependency relation is given as the label of the arrow. For example, the dependency tree of service rule "Company A's machine can be used by Company B" is shown in Figure 2(a). In the dependency tree, the action verb "used" is the root of the tree. The Stanford Parser [7] can be used to discover the dependencies in a sentence.

C. Existing Service Rule Convertor and Its Shortcoming

In our previous work [3], we developed a scheme that automatically converts service rules written in natural language to Jena rules. The scheme analyzed typical service rules and classified the semantic meanings contained in service rules into several semantic categories. Each category is identified by some keywords in natural language. Table 1 shows a subset of the identified categories; and, a subset of the keywords for each category.

Table 1 Semantic Categories and their Keywords

ID	Category	Keywords
1	?client ***has access*** to ?resource	share, allow
2	?resource ***is available***	available
3	the ***identity*** of ?provider or ? client	name, identity
4	the ***location*** requirement of ?provider or ?client	based, located, country, city
5	?provider ***owns*** ?resource	machine, tool

The "Keywords" column contains the keywords that are used to match with the words in the service rule. If the keywords appear in a service rule, it means that the service rule contains the semantic meaning of the corresponding semantic category. In the table, ?entity denotes a variable that will be bound to an item when the inference engine matches the triples in a Jena rule with the triples in the RDF graph of the ontology.

Service rules are used to specify the conditions under which resources can be used. The outcome of the service rules (i.e. the conclusion in the corresponding Jena rules) always states that clients can access some resources or resources are available. There are normally some constraints (e.g. Auckland-based, etc.) on the service providers, the clients and the resources in the service rules. The conclusion of the Jena rules can only be asserted if these constraints are satisfied. Thus, these constraints are mapped to the premises of the Jena rules.

Using the keywords for the semantic categories, the semantic meanings of the service rule can be effectively identified. For example, service rule "Company A shares machines with Auckland-based companies." means that "any company in Auckland can use Company A's machine". From the keywords, the semantic categories identified by the words in the service rule are as below:

(1) Word "shares" means that the conclusion of the Jena rule should state "*?client* has access to *?resource*" (category 1).

(2) Word "Company A" means that the premise of the Jena rule should have "the identity of *?provider* or *?client*" (category 3) and the identity should be "Company A".

(3) Word "machines" means that the premise of the Jena rule should have "?provider ***owns*** ?resource" (category 5).
(4) Words "Auckland" and "based" mean that the premise of the Jena rule needs to state "the location requirement of *?provider* or *?client*" (category 4) and the location should be Auckland-based.

It can be seen that, if all the premises can be matched by the triples in the ontology, "*?client*" in the conclusion of the Jena rule is a company in Auckland and the resource belongs to Company A due to the premises in item 4 and 3 are satisfied. Thus, the conclusion of the Jena rule and the meaning of the service rule coincide.

For a given service rule, by comparing the words in the service rules and the keywords identifying semantic categories, the semantic meanings expressed in the service rules can be recognized. That is, the semantic categories associated with the service rules can be discovered. Then, all the semantic categories associated with the service rules are mapped to the triples that form a semantic rule.

Although the scheme in [3] works well with the service rules defined in the manufacturing ontology in [9], one issue with the scheme is that it is not flexible enough to cope with a variety of ways that people specify the service rules. For example, consider the following two service rules that have the same meaning.
(a) Company A allows Company B to use Company A's machine.
(b) Company B is allowed to use Company A's machine.

Since the service providers always appear in the subject of the service rules in [9], in [3], it is assumed that the subject of a service rule always contain the service provider. Based on this assumption, the semantic meaning of service rule (a) can be correctly obtained as "?client has access to ?resource where ?client is Company B". However, the meaning of service rule (b) would be regarded as "?client has access to ?resource where ?client is Company A". Clearly, the conversion of service rule (b) is wrong as it does not identify the resource provider and the client correctly.

In order to improve the conversion accuracy when applying [3] to other manufacturing ontologies, it is necessary to analyze the locations that the words appear in the service rule and the relationship between the words. For example, in service rule (b), since "Company A's" is a possessive noun that precedes the resource (i.e. the word "machine"), it can be concluded that "Company A" must be the owner of the resource. As a result, Company B and Company A can be identified as the client and the provider of the resource respectively.

III. IDENTIFYING SERVICE PROVIDERS AND CLIENTS

The service providers and the clients can be identified by analysing the possessive noun of the resource or the absence of certain components of a sentence, e.g. the object. First, the typical patterns of the service rules are given and the method for identifying the providers and the clients in different patterns are discussed.

A. Service Rules Written in the Passive Voice

The three service rules below represent the typical service rules when they are written in the passive voice. Determining whether a sentence is in the passive voice can be carried out by checking whether the root of the dependency tree has a *passive auxiliary* relation (i.e. *auxpass*) with one of its child as shown in the dependency tree of sentence P1 in Figure 2(a).
(P1) Company A's machine can be used by Company B.
(P2) Company A's machine can be used in June.
(P3) Company A is allowed to use Company B's machine.

For P1 and P2, "Company A's machine" is the subject; "Company A" is the subject of P3; and, the complete predicate consists of the rest of the words in each sentence. In the dependency tree of a sentence in the passive voice, the subject is stored in the subtree that has the *passive nominal subject* relation (i.e. *nsubjpass*) with the root of the tree as shown in Figure 2(a). To determine whether the subject contains the resource specified in the service rule, the nodes in the *nsubjpass* subtree can be checked to see whether they have any node with the word that represents a resource, e.g. machine, tool, etc. To find out whether the subject of a service rule includes the description of the resource provider or the client, the nodes in the *nsubjpass* subtree are examined to see whether they have words that correspond to an agent that can play the role of the resource provider or client, e.g. company, group, etc. Similarly, to discover whether the complete predicate of a sentence contains the resource, the resource provider or the client, the nodes in the non-*nsubjpass* subtrees are inspected to see whether they have names that correspond to a resource or an agent. The resources are normally associated with a possessive noun (e.g. Company A's, etc.) which indicates the provider of the resource. In the dependency tree, the relation between the possessive noun and the resource is identified by relation *nmod:poss* as shown in Figure 2(a) (i.e. the relation between "machine" and "A").

In pattern P1 and P2, the subject contains the resource as they both have the word "machine". As the word "machine" is preceded by a possessive noun in both P1 and P2 (i.e. Company A's), the possessive noun must represent the resource provider. That is, Company A is the resource provider. P1's complete predicate has an agent (i.e. Company B). As the resource is described in the subject, the agent in the predicate must be the client. Thus, it can be determined that P1 has the semantic meaning of "?client ***has access*** to ?resource" (i.e. category 1 in Table 1). P2 does not have any agent in the complete predicate. Thus, it simply specifies that a resource is available. That is, the semantic meaning of P2 is "?resource ***is available***" (i.e. category 2 in Table 1).

In pattern P3, the complete predicate of P3 has a resource (i.e. machine) and the resource has a possessive noun (i.e. Company B's). Hence, it is clear that Company B is the resource provider. As the resource provider is in the complete predicate, the agent in the subject of the sentence (i.e. Company A) must be the client; and the semantic meaning of P3 corresponds to the one in category 1 of Table 1.

B. Service Rules Written in the Active Voice

The patterns of the service rules that are written in the active voice are given below.

(P4) Company A's machine is available.

(P5) Company A can use Company B's machine.

(P6) Company A allows Company B to use Company A's machine.

P6's dependency tree is shown in Figure 2(b). In the dependency trees of the rules written in the active voice, (a) the subject of the sentence written in the active voice has the *nominal subject* relation (i.e. *nsubj*) with the root of the tree, and (b) the object of a sentence has a *direct object* relation (i.e. *dobj*) with the root of the tree.

In pattern P4, the resource (i.e. machine) appears in the subject (i.e. the *nsubj* subtree), and the possessive noun (i.e. Company A's) indicates that Company A is the resource provider. P4 does not have an object or any infinitive phrase for the root. Thus, this type of service rule simply states that the resource is available (i.e. category 2 in Table 1).

In P5, the object (i.e. the *dobj* subtree) contains the resource (i.e. machine). The possessive noun of the resource (i.e. Company B's) indicates that Company B is the resource provider. As the resource provider is in the object of the sentence, the agent appearing in the subject (i.e. Company A in the *nsubj* subtree) must be the client.

In P6, "to use Company A's machine" is an infinitive phrase that is used as an adverb of the verb "allows" to express what is allowed. An infinitive phrase has a verb (i.e. "use") and a complement which is the object of the verb (i.e. Company A's machine). In the dependency tree, an infinitive phrase has the *open clausal complement* relation (i.e. *xcomp*) with the root as shown in Figure 2(b). In P6, the infinitive phrase (i.e. the *xcomp* subtree) contains the resource (i.e. machine) and the provider of the resource (i.e. the possessive noun "Company A's"). As Company A is the provider of the resource and it coincides with the name of the agent in the subject of the sentence (i.e. the *nsubj* subtree of the root), it can be inferred that the resource provider is also described in the subject of the sentence. As a result, the agent in the object (i.e. the *dobj* subtree) must be the client. As both client and resource provider are present in P5 and P6, their semantic meanings correspond to category 1 in Table 1.

IV. Identifying Semantic Categories

According to the pattern that a service rule belongs to, the intension of the rule is ascertained (i.e. the rule either specifies that a resource is available or a client can access a resource). Thus, the semantic category for the conclusion of the Jena rule can be decided.

The subject and the predicate of the service rule might include words that specify the constraints on the client/provider to which the service rule applies. These constraints are represented as the premises in the Jena rules. For example, in service rule "Auckland-based companies only share resources with NZ-based companies", the subject (i.e. "Auckland-based companies") indicates that the rule only applies to the service providers that are in Auckland. This constraint on the service provider is converted to a premise in the Jena rule. This ensures that the resulting Jena rule only applies to the resources supplied by the providers in Auckland. This is because the conclusion of a Jena rule is asserted only if all the premises of the Jena rule can be matched with the triples in the RDF graph of the ontology.

The words of the service rule are compared with the keywords in Table 1 to reveal the semantic categories of the constraints in the rule. If a word matches a keyword, the matched semantic category, the word and the agent described by the word (i.e. whether the word associates with the resource provider or the client) are recorded.

Algorithm *classify* describes the steps in classifying the information in a service rule into semantic categories. The algorithm returns a set of tuples. Each tuple consists of three elements, i.e. (*agent, category, word*). *category* is the semantic category identified by *word* in the service rule; and *agent* indicates whether the identified category is for the service provider or the client.

The pattern of a given service rule is determined first (line 3 to 27); and, once the rule's pattern is known, the semantic meaning of the conclusion part of the Jena rule can be decided as described in §III. As the conclusion of the Jena does not need to be associated with any agent (i.e. service provider/client), the agent element in the resulting tuple is set to "-". *subjHas* and *predHas* are used to record whether the provider/client is in the subject or the predicate of the service rule respectively. For the premises of the Jena rule, each node (i.e. each word in the service rule) in the dependency tree is checked to determine the semantic category that it corresponds to (line 28 to 33).

	classify(*tree*)
1	**input**: *tree* is the dependency tree of a service rule
2	**output**: the set of semantic categories that exist in a service rule
	// *SC* records the set of semantic categories found
	// in a service rule
3	$SC \leftarrow \emptyset$
4	let *root* be the root of *tree*
	// determine the pattern of the service rule, and find
	// the semantic category for the conclusion of
	// the Jena rule
5	**if** the service rule is in the passive voice **then**
	// try to match pattern P1
6	**if** the subject contains the resource **and** the predicate contains an agent **then**
7	$SC \leftarrow SC \cup \{(-, 1, root)\}$
8	*predHas* = "client"; *subjHas* = "provider"
	// try to match pattern P2
9	**else if** the subject contains resource **then**
10	$SC \leftarrow SC \cup \{(-, 2, root)\}$
11	*subjHas* = "provider"; *predHas* = null
	// try to match pattern P3
12	**else if** the subject contains an agent **and**

```
                the predicate contains a resource then
13              SC ⟵ SC ∪ {(−, 1, root)}
14              subjHas = "client";  predHas = "provider"
15          end-if
16      else // the service rule is in the active voice
            // try to match pattern P4
17          if the sentence does not have an object
                and any infinitive phrase then
18              SC ⟵ SC ∪ {(−, 2, root)}
19              subjHas = "provider";  predHas = null
            // try to match pattern P5
20          else if the object of the sentence is a resource
                    quantified by a possessive noun then
21              SC ⟵ SC ∪ {(−, 1, root)}
22              subjHas = "client";  predHas = "provider"
23          else // pattern P6
24              SC ⟵ SC ∪ {(−, 1, root)}
25              subjHas = "provider";  predHas = "client"
26          end-if
27      end-if
        // find the semantic categories corresponding to
        // the words of the service rule
28      for each node in tree do
            let category be the semantic category in Table 1
                that node corresponds to
            // for nodes in the subject of a service rule
29          if node is in the nsubj or nsubjpass subtree then
30              SC ⟵ SC ∪ (subjHas, category, node)
            // for nodes in the predicate of the service rule
31          else SC ⟵ SC ∪ (preHas, category, node)
32          end-if
33      end-for-each
34      return SC
```

Here is an example that shows how the semantic categories in service rule "Company A is allowed to use Company B's machine." are identified by the algorithm.

(1) The rule corresponds to pattern P3. Thus, tuple *(-, 1, allowed*) is recorded as the semantic category for the conclusion for the Jena rule. The pattern also means that *subjHas* is set to "client" and *predHas* is given "provider".

(2) "A" is the name of a company and it is in the subject of the service rule. Hence, according to the value of *subjHas* and category 3 in Table 1, tuple (*client, 3, A*) is recorded.

(3) "B" is the name of a company and it is in the predicate of the service rule. Hence, according to the value of *predHas* and category 3 in Table 1, tuple (*provider, 3, B*) is recorded.

(4) "machine" is a resource and is in the predicate of the service rule. According to category 5 in Table 1 and the value of *predHas*, tuple (*provider, 5, machine*) is recorded.

V. Converting to Semantic Rule

The semantic rule that corresponds to a service rule is generated by converting the semantic categories identified for a service rule to Jena rule conclusion or premises. Algorithm *convertToRule* gives a brief description on how to generate the semantic rule according to the semantic categories that have been identified for the service rule. Due to limit on the length of the paper, the algorithm only covers the conversion of some of the semantic categories and a limited amount of sentence structures.

```
    convertToRule(SC, tree)
1   input: tree is the dependency tree of the service rule
           SC is the set containing the identified
               semantic categories in the service rule
2   output: rule is the Jena rule that corresponds to
               the service rule
    // convert the semantic category
    // for the conclusion of the Jena rule
3   let root be the root of tree and
        (−, category, root) be a tuple in SC
4   switch category:
        // category: ?client has access to ?resource
5       case 1:
6           add "(?client  mc:hasAccessTo  ?resource)"
                to conclusion of rule
        // other categories for conclusion of Jena rule
7       case … :
8           …
9   end-switch
    // convert semantic categories for the premises
10  for each node in tree do
11  // check whether node corresponds to a semantic
    // category
12    if there exists a tuple (agent, category, node) in SC
        then
13      switch category:
14      case 3: // the identity of ?provider or ?client
15          if agent = "provider" then
16              add "(?provider  mc:name  node)"
                    to the premise of rule
17          else // i.e. agent = "client"
18              add "(?client  mc:name  node)"
                    to the premise of rule
19          end-if
20      case ...: //other semantic category
            …
21      end-switch
22    end-if  // there exists a tuple
23  end-for-each
24  return rule
```

The set of semantic categories identified by algorithm *classify* is used by *convertToRule*. First, the conclusion of the Jena rule is generated according to the semantic category for the rule conclusion (line 3 to 9). The items in triple (*?subject, ?property, ?object*) need to be represented using the terms of the ontology. For category "?client ***has access*** *to* ?resource", "***has access***" corresponds to *?property* in the triple. The ontology term for "***has access***" is *mc:hasAccessTo* in [9] (line 6). In the conclusion of the Jena rule, ?client (i.e. *?subject* in triple) and ?resource (i.e. *?object* in triple) are represented as variables that will be bound to values through the matching

of the premises of the Jena rule. Thus, there is no need to match *?subject* and *?object* in the conclusion triple with ontology terms.

The premises of the rule are converted next (line 10 to 23). Each node in the dependency tree is a word in the service rule. If the word corresponds to a semantic category (line 12), the category is converted to a Jena rule premise (line 13 to 21). The relevant items in a premise triple (*?subject*, *?property*, *?object*) needs to be replaced by appropriate ontology terms or values. For category "the ***identity*** of *?provider* or *?client*", in the ontology of [9], ontology term for the ***identity*** property is *mc:name* (line 16 and 18). The identity (i.e. *?object*) is a literal value. Hence, the value of the node is used as *?object* (line 16 and 18). The value of *agent* parameter indicates whether the word is associated with the provider or the client. Thus, it is used to determine whether *?subject* should be set as ?*provider* or *?client* (line 15 and 17).

VI. Evaluation

Experiment was carried out to evaluate the rule conversion accuracy. The service rules in [9] have been extended by describing the rules in [9] in a variety of ways. About 600 service rules were generated. 20% of the rules were randomly selected for conversion. The results show that all the rules were converted correctly to reflect the semantic meanings of the original service rules. As the experiment only tested a limited number of service rules, it is conceivable that some rules with peculiar sentence structure might be encountered when a much larger number of rules are converted and these odd rules might not be converted correctly. However, the experiments showed that the proposed scheme works well for the service rules with commonly observed sentence structures.

VII. Related Work

Kang et al. [6] used NLP to extract manufacturability rules from English sentences. Compared with [6], the scheme in this paper handles more complicated semantic categories that involve many more keywords than the ones in [6]. GE's SADL [13] enables the manufacturability rules to be accessed using an English-like language. Although users use an English-like language in SADL, they still need to know the syntax of the language. In contrast, the scheme in this paper allows users write the service rules in natural language. This makes it easier for users to use the system. WHYPER [11] and SUPOR [12] were developed to carry out privacy and security checks. Both schemes used the Stanford parser to parse the application descriptions into individual words in order to identify sensitive words. Unlike [11, 12], the scheme in this paper needs to analyze the dependencies between the words to obtain the semantic meanings of the service rule. NLP techniques have been used in use case analysis to check whether the documents conform to industry standard [15, 8]. Unlike the scheme in this paper, they do not analyze the semantic meanings of the sentences.

VIII. Conclusions

This paper proposed a scheme that can improve the conversion accuracy when converting service rules in cloud manufacturing to semantic rules that are used by the ontology hosting the cloud manufacturing platform. The scheme classified the service rules into six patterns. The Stanford parser is used to parse the words in a given service rule to discover the dependencies between the words. Using the dependency relations and the classification of the service rules, the resource provider and the client in a service rule can be identified accurately. The classification of the service rule also allows the semantic meaning of the service rules to be captured easily. Experiments showed that the proposed scheme achieves better conversion accuracy than the existing scheme.

References

[1] Apache Jena, https://jena.apache.org/. Accessed on 2016-09-11.

[2] S. Bandyopadhyay, S. K. Naskar and A. Ekbal, "Emerging Applications of Natural Language Processing: Concepts and New Research", IGI Global, Information Science Reference, 2013.

[3] X. Ye and P. Zhao, Converting Service Rules to Semantic Rules, In Procs. of 2016 IEEE International Service Computing Conference.

[4] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research," IEEE Computational Intelligence Magazine, vol. 9, no. 2, pp. 48-57, 2014.

[5] M. de Marneffe et al. Universal Stanford dependencies: A cross-linguistic typology. LREC 2014: 4585-4592

[6] S. Kang et al., Extraction of Manufacturing Rules From Unstructured Text Using a Semantic Framework. In Pros. Of ASME 2015 Intl. Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. V01BT02A033.

[7] M. C. De Marneffe, B. MacCartney, and C. D. Manning, "Generating typed dependency parses from phrase structure parses," In Proc. 5th LREC, 2006, pp. 449-454.

[8] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari, "Application of linguistic techniques for use case analysis," Requirements Engineering Journal, vol. 8, no. 3, 2003, pp. 161-170.

[9] Y. Lu, X. Xu, J. Xu, "Development of a Hybrid Manufacturing Cloud," J Manuf Syst, 2014.

[10] C. Singh, Q. Shao, Y. Lu, X. Xu, X. Ye, Tool Selection: A Cloud-Based Approach, Lecture Notes in Electrical Engineering, 2014, Volume 301, pp 237-245

[11] R. Pandita, X. S. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: Towards Automating Risk Assessment of Mobile Applications," In Procs of the 22st USENIX conference on Security symposium. 2013.

[12] J. Huang et al., SUPOR: Precise and Scalable Sensitive User Input Detection for Android Apps. In Pros. of the 24th USENIX Security Symposium, pp 977-992, 2015

[13] A. Rangarajan et al., Manufacturability analysis and design feedback system developed using semantic framework, In ASME Pros. of 18th design for manufacturing and the life cycle conference, 2013.

[14] RDF Working Group, "Resource Description Framework (RDF)," W3C, Available: http://www.w3.org/RDF/. Accessed on 2015-11-08.

[15] S. M. Sutton et al., "Text2Test: Automated inspection of natural language use cases," In Proc. 3rd ICST, 2010, pp. 155-164.

[16] M. C. De Marneffe, "Stanford typed dependencies manual," 2008.

[17] W3C, "Semantic Web," Available: http://www.w3.org/standards/semanticweb/. Accessed on 2016-09-11