Vol. 49 No. 2 IPSJ Journal Feb. 2008

Regular Paper

Web Resource Categorization by Detecting Potential Relations

Minghua Pei,[†1] Kotaro Nakayama,[†1] Takahiro Hara[†1] and Shojiro Nishio[†1]

Since Semantic Web is increasing in size and variety of resources, it is difficult for users to find the information that they really need. Therefore, it is necessary to provide an efficient and precise method without explicit specification for the Web resources. In this paper, we proposed the novel approach of integrating four processes for Web resource categorization. The processes can extract both the explicit relations extracted from the ontologies in a traditional way and the potential relations inferred from existing ontologies by focusing on some new challenges such as extracting important class names, using WordNet relations and detecting the methods of describing the Web resources. We evaluated the effectiveness by applying the categorization method to a Semantic Web search system, and confirmed that our proposed method achieves a notable improvement in categorizing the valuable Web resources based on incomplete ontologies.

1. Introduction

Recently, the Semantic Web shows an increase in size and variety of resources. The large spate of unstructured, heterogeneous resources on the current Internet makes it difficult for users to find the information they really need. A new concept, Semantic Web, appeared to make the Web machine readable and understandable by appending semantic information to the Web pages. Thus Web users can make use of Web resources more efficiently. Much work have been done on generating Semantic Web resources[5)], ranking Semantic Web documents[4)], and processing approximate query for Semantic Web search such as Corese[1)]. However, almost all these conventional methods are based on well-structured ontologies. As the WWW changes dynamically, it's impossible to structure a perfect ontology to include the relations of all the Web resources. In Semantic Web, a class is a type of Web resource. A class has some properties and it can be a subclass of another class. We define a group of classes as a category which contains some base classes and all of their offspring. Thus, it cannot be matched by the ontology-based search engines for a Web resource which is not defined as a class in the structured ontologies. For example, a query like "Find me a person who is making a study of the Semantic Web". If there is a Web resource describing a Ph.D. student who has a research interest in Semantic Web but there are not any relations defined for its resource type (e.g. original class "Phd student") in the ontology, then the ontology-based search engines cannot match it as a person due to lack of relation. Therefore, detecting the potential relations with existing ontologies becomes an important and emerging issue to use Web resources efficiently.

For this purpose, we proposed a Web resource categorization method which extracts potential relations among the Web resources. Here, we define a group of classes as a category, which contains some base classes and all of their offspring. The proposed method extracts the relation from not only a class hierarchy in ontologies, but also the description of Web resources such as class name and property pattern. We attempt to detect related and potentially related Web resources by analyzing the existing ontologies and dictionaries. We focus on some factors from the concept structure, the word relations in the dictionary and the description method of the Web resources. We defined an algorithm based on these factors to perform the categorization. A big advantage of our proposed method is that it can break through the limitation of the incomplete ontologies. In other words, the Web resources related to the categories can be categorized by using both the explicit relations and the detected relations. As a result, it provides a greater possibility of semantically matching the user queries.

We also evaluated our proposed method by implementing it on a Semantic Web search en-

†1 Graduate School of Information Science and Technology, Osaka University

gine. The main contribution of our research is that it can categorize many valuable Web resources having no definitions in ontologies and decrease the retrieval cost. As an example of the categorization result, only 17% of Web resources of our experiment data can be categorized based on the explicit relations defined in ontologies, but 58% of Web resources can be categorized by using our proposed method. As an example of decreasing the retrieval cost, in searching a person, the search engine needs to check only about 15% of Web resources categorized to Person instead of the whole database.

The rest of the paper is organized as follows: Section 2 describes our proposed categorization method. Section 3 presents an experiment for evaluating our proposed method. A discussion on related work is given in Section 4. Conclusions and future work are presented in Section 5.

2. Categorization Algorithm

2.1 Approach

Since problems such as lack of relations and definitions always exist in real world Semantic Web data, the central idea of our approach is to categorize the resource types by using not only well-defined relations but also some potential semantic relations for incomplete ontologies. We take into account four factors as follows.

The first factor is the relations among classes defined in Web ontologies. According to the definition, all extended classes based on relations such as "rdf:subClassOf", "owl:equivalentClass" and "owl:sameAs" should be in the same category. The second is the relations between class name and category. Sometimes, a class can be identified from another class, which has the same class name in different namespaces. For example, a class with the name "student" always belongs to the category Person. If we get these word relations, we can extend the category by the classes whose names are in those related words. The third is the word relations defined in dictionaries (we used WordNet[12]). For the classes having not enough information to categorize them, we can do so by using the general meaning of the class name and the relations of the words in dictionaries. The last factor is the relations between the property patterns and the category. In RDFs, a number of properties are used to describe a Web resource. The associations between properties and category are helpful for providing a deeper categorization.

We defined four processes based on the factors mentioned above, as shown in **Fig. 1**, to perform the categorization. In the following subsections, we describe the processes in detail.

Before executing the four processes, we defined several typical categories that are frequently used in searching queries, such as "Person", "Animal", "Event" and "Location". Let A denote the set of these categories specified for categorization, and a denotes a category in A. This denotation is used in each process. A measure of the relation between class c and category a is defined as relation strength and denoted as $r_{c,a}$. We defined some class names to get base classes for each category, and the relation strength of initial classes to the category equals 1.

2.2 Process 1: Categorize Offspring

In *Process* 1, we categorize the classes existing in the ontology based on the hierarchical structure of classes, which is a traditional method. If a class has a relation to a category, its subclasses ("rdf:subClassOf") also have a relation, which is considered a little bit weaker than that of their super class, to the category. The equivalent classes ("owl:equivalentClass" and "owl:sameAs") have equivalent relation strength to the same category.

Process 1 is performed by using a recursive function $RER(c_0, a, r_{c_0,a})$. Let c_0 be a class in category a with the relation strength as $r_{c_0,a}$. If class c, which is an equivalent class or a subclass of class c_0 in category a, hasn't been categorized to category a, $RER(c_0, a, r_{c_0,a})$ can categorize class c to category a with relation strength as $r_{c,a}$. By recursively calling itself, $RER(c_0, a, r_{c_0,a})$ categorizes all the offspring classes of c_0. Here, a coefficient k is given for the degressive strength of relation in the hierarchical structure. We consider that the classes having an equivalent relation, such as "http://pervasive.semanticweb.org/ont/dev/person#Document" and "http://xmlns.com/foaf/0.1#Document", should have same relation strength to the category, thus we define k as 1 for a class in the same level. Then we conducted some preliminary experiments to specify a value of k which is lower than 1 for a lower level class. While checking the number of categorized classes for values of k such as 0.90, 0.91, ..., 0.99, we found out that the growth of the number changes near the value of k equal to 0.95. The number increases greatly while k is

```
Algorithm $Process\ 1(A)$
1:  Foreach ($category\ a \in A$)
2:    Foreach ($class\ c_0 \in a$)
3:      $RER(c_0, a, r_{c_0,a})$;
Function $RER(c_0, a, r_{c_0,a})$
1:  Foreach ($class\ c \in c_0.Children$)
2:    $r_{c,a} = k \times r_{c_0,a}$;
3:    If ($c.Contain(a) = false$)
4:      $a.Add(c, r_{c,a})$;
5:      $RER(c, a, r_{c,a})$;
6:  Foreach ($class\ c \in c.Equivalents$)
7:    $r_{c,a} = r_{c_0,a}$;
8:    If ($c.Contain(a) = false$)
9:      $a.Add(c, r_{c,a})$;
10:     $RER(c, a, r_{c,a})$;

Algorithm $Process\ 2(A, minW)$
1:  Foreach ($category\ a \in A$)
2:    Foreach ($classname$
              $cName \in getName(a)$)
3:      $l = getW(cName, a)$;
4:      If ($l > minW$)
5:        Foreach ($class\ c$
                  $\in getClass(cName)$)
6:          If ($c.Contain(a) = false$)
7:            $r_{c,a} = avg(getR(cName, a))$
                $\times 0.95$;
8:            $a.Add(c, r_{c,a})$;
Function $getW(i, j)$
1:  $W_{i,j} = tf_{i,j} \times log\frac{N}{df_i}$;
2:  $W'_{i,j} = \frac{W_{i,j}}{max(W_{i,j})}$;
3:  Return $W'_{i,j}$;

Algorithm $Process\ 3(a, minW)$
1:  Foreach ($classname$
              $wd_0 \in getName(a)$)
2:    $B = wd_0.Children$;
3:    Foreach ($classname\ wd \in B$)
4:      Foreach ($class\ c \in getClass(wd)$)
5:        If ($c.Contain(a) = false$)
6:          $r_{c,a} = avg(getR(wd_0, a)) \times 0.95$;
7:          $a.Add(c, r_{c,a})$;

Algorithm $Process\ 4(A, minf, minW)$
1:  Foreach ($category\ a \in A$)
2:    $P_a = getPattern(a, minsup, minconf)$;
3:      Foreach ($propertyset\ p \in P_a$)
4:        Foreach ($class\ c \in getType(p)$)
5:          If ($c.Contain(a) = false$)
6:            $r_{c,a} = avg(getR_p(p, a)) \times \frac{\frac{N_r}{N}+1}{2}$;
7:            $a.Add(c, r_{c,a})$;
Function $getPattern(a, minsup, minconf)$
1:  $initialize\ P_1$;
2:  For ($j = 2; j < maxLen; j++;$)
3:    $P_j = apriori(minsup, P_{j-1})$;
4:    Foreach ($propertyset\ p \in P_j$)
5:      $conf = \frac{count(p,a)}{count(p)}$;
6:      If ($conf > minconf$)
7:        $P_a.Add(p)$;
8:        $P_j.Remove(p)$;
9:  Return $P_a$;
```

Fig. 1 Algorithm of 4 processes.

not larger than 0.95, and the number keeps increasing but at a slower pace while k is larger than 0.95. Since too high values of k decreased the precision of categorization in some preliminary experiments, we specify k as 1 for the same level and as 0.95 for a lower level class. The relation strength of class c to category a, $r_{c,a}$ can be calculated by Eq. (1).

$$r_{c,a} = k \times r_{c_0,a}, \tag{1}$$

$$k = \begin{cases} 1 & \text{equivalentClass relation,} \\ 0.95 & \text{subClass relation.} \end{cases} \tag{2}$$

2.3 Process 2: Extend Category Based on Characteristic Class Name

By performing *Process* 1, we get some categorized classes for each category. Then we take notice of the class names that represent the characters of a category. We consider that the classes having the characteristic class names of category a can also be categorized into a.

In *Process* 2, we categorize the classes not existing in the ontology based on important class names. This category extension consists of two steps, that is, extraction of important class names for each category and categorization of the classes having these class names as well as the offspring classes of the newly categorized classes. The algorithm is described concretely as follows.

For each category a, all the class names are extracted by function $getName(a)$. Then we compute how much a class name $cName$ can represent the character of category a by function $getW(cName, a)$ based on tfidf[7]. Since tfidf in text retrieval can extract the words that are particularly important to a document, we attempt to find out the important class names of each category by using tfidf. The weight of class name c_i in category a_j is denoted as $W_{i,j}$, which is calculated by Eq. (3).

$$W_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right), \tag{3}$$

$tf_{i,j}$ = number of occurrences of i in j,
df_i = number of categories containing i,
N = total number of categories in A.

Since $W_{i,j}$ is largely affected by the number of classes in the category, we normalize $W_{i,j}$ in

$[0,1]$, and $W'_{i,j}$ denotes the normalized value which can be calculated by Eq. (4).

$$W'_{i,j} = \frac{W_{i,j}}{\max(W_{i,j})}. \quad (4)$$

We consider the class names having $W'_{i,j}$ larger than a threshold $minW$ as the important class names for the category. Here, we also conducted some preliminary experiments to determine a threshold $minW$ for $W'_{i,j}$. We specified $minW$ as 0.01, 0.02, ..., 0.1. When $minW$ was larger than 0.04, the number of categorized classes became much smaller. When $minW$ was smaller than 0.01, the number became much larger, however, we found out that it also became time consuming. Thus, we try to set the threshold $minW$ between 0.02 and 0.04. In our experiment, we specified it as 0.03 for performing *Process* 2 efficiently.

For each important class name *cName*, we extract all the classes having class name *cName* in category a from all the classes by function $getClass(cName)$. From these classes, we extend category a with each class c which denotes a class having a class name as *cName* but not existing in category a. For a set of classes having class name *cName* and existing in category a, $getR(cName, a)$ is a function to get the relation strengths for the classes having the class name *cName* to category a, and $avg(getR(cName, a))$ denotes the average. Then, the relation strength of class c to category a can be calculated by Eq. (5).

$$r_{c,a} = avg(getR(cName, a)) \times W'_{i,j}. \quad (5)$$

As the second step, we extend the categories with all the offspring classes of the newly categorized classes by executing *Process* 1 again.

2.4 Process 3: Detect Potential Relation from Dictionary

Process 3 categorizes the classes not existing in the ontologies based on the superior-inferior relations among the words in dictionaries. We used an "is-a" relationship between superior and inferior words in WordNet. The algorithm is described concretely as follows.

We get all the class names in category a by function $getName(a)$ where we pruned the class names that exist in more than one category. For each class name $wd_0 \in getName(a)$, we extract all the offspring class names from WordNet and store them in B. For each class name wd in B, we get all the classes whose class name is wd. If these classes do not exist in category a, they are categorized into a.

There exist many multisense words that are not in common use. For example, the word "fish" has four senses in WordNet. The first sense is a kind of aquatic vertebrate. The second sense is a kind of food. The third sense is a person who is born while the sun is in Pisces. The fourth sense is a sign of the zodiac. In general, the first sense denotes the most common meaning. Therefore, we consider that using the word relation between the first senses has the highest possibility of getting the potential subclasses correctly. Thus, in our experiments, we use the relation between the first word senses of nouns in WordNet.

Since an "is-a" relationship can be considered as an extension to a lower level, we compute relation strength based on that of classes having a superior class name. $r_{c,a}$ denotes the relation strength of class c having class name wd to category a. wd_0 denotes the superior of wd, and $getR(wd_0, a)$ computes the relation strength between classes having class name as the superior of wd to category a. $avg(getR(wd_0, a))$ denotes their average. Then, $r_{c,a}$ can be calculated by Eq. (6).

$$r_{c,a} = avg(getR(wd_0, a)) \times 0.95. \quad (6)$$

Then, we execute *Process* 2 again to extend categories using all the offspring classes with the root names having high tfidf values.

2.5 Process 4: Detect Relation by Property Pattern Analysis

In *Process* 4, we perform a deeper categorization for the classes not existing in the ontology based on the description of the Web resources. Due to the complexity of *Process* 4, we structured this subsection as follows. First, we explain about the description of Web resources then define several conceptions for property set, support, confidence, and property pattern. Next, we explain the most important function in *Process* 4 for pattern discovery. Finally, we explain the main algorithm of *Process* 4.

2.5.1 Definitions

In the Semantic Web, one resource is usually described by several properties. For example, a Web resource "AKT:Employee" [10] is described by properties "has-room-number", "has-web-address", "family-name" and "given-name". It is considered that if these properties were used together only for describing a person, then the other resources described by these four properties to have a high possibility of being a similar class to a person.

Here, we define property set p as a set of properties that describe the same resource. Support of property set p in category a is defined as the number of resources described by p in a divided by the number of all the categorized resources (Eq. (7)), which denotes the frequency of p occurred in a. Confidence of property set p in category a is defined as the number of resources described by p in a divided by the number of all the resources described by p (Eq. (8)), which denotes the possibility of p related to a. Then, the property pattern in category a is defined as a property set having $support_{p,a}$ and $confidence_{p,a}$ larger than the threshold *minsup* and *minconf* respectively. The number of properties in a property pattern is defined as the length of the pattern.

$$support_{p,a} = \frac{|\text{resources described by } p \text{ in } a|}{|\text{all the categorized resources}|}, \qquad (7)$$

$$confidence_{p,a} = \frac{|\text{resources described by } p \text{ in } a|}{|\text{categorized resources described by } p|}. \qquad (8)$$

2.5.2 Pattern Discovery

The pattern discovery is implemented by function *getPattern*(a, *minsup*, *minconf*), which is based on apriori algorithm[8)] to measure the cooccurrence of properties in describing the Web resources. Let p_j denote a property set whose length is j, p_{j+1} denote a property pattern whose length is $j+1$. It is easy to understand that if $p_{j+1} \supseteq p_j$ then the resources described by p_{j+1} are also described by p_j. Therefore, when p_j is a property pattern, we don't need to generate a longer property pattern p_{j+1} from p_j. Since it has been shown in our preliminary experiments that most of the available property patterns can be extracted before their length reaches 7, we set the threshold *maxLen* as 7 for property pattern length j. In the algorithm, P_j denotes a group of property set which has $support_{p,a}$ larger than *minsup*, P_a denotes the whole set of property patterns having $confidence_{p,a}$ larger than *minconf* in category a. For each step of j, we extract the property sets P_j. Then, for each property set p in P_j, if $confidence_{p,a}$ is larger than *minconf* then we add p to P_a and remove it from P_j not to generate a longer property pattern from p. Finally, we return the property patterns P_a.

2.5.3 Algorithm

Process 4 can be described as follows. For each category a we categorize the Web resources and extract the property patterns by function *getPattern*(a, *minsup*, *minconf*). For each property pattern p in P_a, we categorize the resources described by p. Then, we extract all the resource types of the resources described by p by using function *getType*(p). Then, we extend category a with class c which is extracted by *getType*(p) and does not exist in a.

We calculate the relations by considering the frequency of the property pattern in categorized Web resources. In category a, N is the count of the resources in a, and N_r is the count of resources described by property pattern p. We can get the relation strengths of classes related to property pattern p to category a by using function $getR_p(p, a)$. $avg(getR_p(p, a))$ denotes the average. Then, the relation strength of the detected class c to category a can be calculated by following Eq. (9).

$$r_{c,a} = avg(getR_p(p, a)) \times \frac{\left(\frac{N_r}{N} + 1\right)}{2}. \qquad (9)$$

Since some classes are newly categorized from the resource type, we execute *Process* 2 again to extend the category using those newly categorized classes.

2.6 Flow of Whole Categorization

The whole flow of the proposed categorization method is not sequential due to the dependency among them. Since *Process* 1 uses explicit relation among classes defined in ontologies and the ontology-based approach is standard in Semantic Web search, *Process* 1 is useful for extension for each category. That is why *Process* 1 is a basic algorithm for other processes and executed repeatedly.

Process 2 can increase the accuracy by justifying the parameter *minW*, however, it only categorizes lesser classes. Thus it is good for the categorization to use *Process* 2 together with *Process* 1. *Process* 2 provides the classes having high possibility to be related to the categories and *Process* 1 extends the categories with those classes.

Process 3 can extract the potential relations of classes from the word relations in dictionaries. It is especially effective in extending some specific categories, such as Person, Food, Restaurant, etc. Thus, we can perform *Process* 3 for those categories to keep a high quality of categorization. We also perform *Process*

3 for all the categories for further categorization. For newly categorized classes in *Process* 3, it is good for categorization to use *Process* 2 together with *Process* 1 repeatedly until the categories become stable.

Process 4 is beneficial for the categorization of Web resources. Since there always exist many resource types having not definition in the ontologies, *Process* 4 uses the cooccurrence of properties in the category to categorize the Web resources. Thus, it can effectively categorize the Web resources having no definition in the ontologies. As another contribution, some detected resource types can be used for the next categorization.

As described above, we determine the categorization flow based on these features in our experiments. First, we perform *Process* 1 with removing some noises, then perform *Process* 2 to start a categorization. Next, we perform *Process* 3 for some categories, such as Person, Food, Restaurant, etc. to keep a high quality of categorization. Next, we perform *Process* 1 and *Process* 2 repeatedly till the categories become stable. Thus *Process* 1 can be performed based on well categorized classes. Next, we perform *Process* 3 then perform *Process* 1 and *Process* 2 repeatedly till the categories become stable again for further categorization. Next, we perform *Process* 4 then perform *Process* 1 and *Process* 2 repeatedly till the categories become stable. Finally, we perform *Process* 4 again to finish the categorization.

Here, we denote the process flow as *Process* 1-2-3-4, when it is performed from *Process* 1 to *Process* 4 in sequence. Since the first step must be *Process* 1, we tried other process flows starting from *Process* 1 to check the number of properly categorized Web resources. We get the result of 39% from *Process* 1-3-2-4, 43% from *Process* 1-4-2-3 and *Process* 1-2-4-3, 40% from *Process* 1-4-3-2, and 35% from *Process* 1-3-4-2. Therefore, we use the process flow *Process* 1-2-3-4 because of its highest amount of properly categorized Web resources compared to others.

3. Evaluation

In this research, we conducted two experiments to evaluate our proposed method. The first is a Semantic Web resource categorization. The second is an implementation of categorization method to a Semantic Web search system to examine the effectiveness of our proposed method.

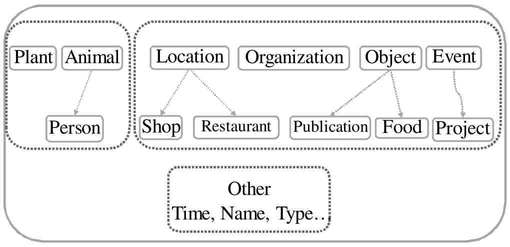

Fig. 2 Categories.

3.1 Preliminary

First, we defined the basic categories used in our experiments based on some frequent class names. Specifically, we counted the appearances of class names, then we chose several class names and defined them as the categories shown in **Fig. 2**. Some categories were defined as sub-categories. A class, which can be categorized to a sub-category, should not be categorized into its super-category to avoid the influence of correlation among categories.

We used two measures *Precision* and *Recall* for the evaluations. For a given user and query, assume that the user marked all the Web resources as being relevant. Let A denote a set of the relevant resources marked by the users, and B denote a set of results output by system. *Precision* is the ratio of the intersection of A and B to the results B, and *Recall* is the ratio of the intersection of A and B to the relevant resources A.

$$Precision = \frac{|A \cap B|}{|B|}, \qquad (10)$$

$$Recall = \frac{|A \cap B|}{|A|}. \qquad (11)$$

In our experiments, we manually made a set of relevant recourses for the categories, and we call it relevant resources in the following part of paper. The approach for deciding how to make the relevant resources is whether the resources seemed to be wanted when a user specifies the category. As we mentioned above, we did not categorize a class, which can be categorized to a sub-category, to its super-category in our proposed method. When a user specifies a category having a sub-category, we considered that the Web resources in the sub-category are also what the user wants. Thus, the Web resources in the sub-category are also marked as the Web resources in its super-category.

3.2 Data Sources

In our experiments, we used the Google API[9] to collect URLs for RDF with the most generic extensions that are likely to be Semantic Web data sources based on generic keywords. Since Semantic Web data sources typically use special file extensions such as rdf, owl and xml, the crawler gathered files with such extensions. We collected 54,679 valid URLs for RDF and parsed them, then we extracted a total of 1,437,717 Semantic Web resources. In Ref. 3), Eberhard reported 1,479 valid RDF files out of nearly 3,000,000 Web pages; in Ref. 6), OntoKhoj reported 418 ontologies out of 2,018,412 Web pages after a 48-hour crawling process. The number of URLs we used in the experiments is quite large scale and practical compared to these existing researches. Furthermore, these URLs are collected by using words (including some "bad" keywords that cannot get any URLs) randomly extracted from WordNet as the keywords without specifying a domain. Therefore, the data set we collected can be considered as having generality in the Semantic Web. About 150,000 triples were extracted for relations of classes and stored in the ontologies. About 111,000 classes were collected but 10% of them did not exist in the ontologies.

We extracted about 5 resources on average for each resource type from Web resources as experimental data. There were about 15,000 resources, which have 3,000 resource types. An interesting fact is that 75% of the resource types did not exist in the ontology. This means that it is impossible to get the most related results by an ontology-based search system 2),11).

3.3 Resource Categorization Result

We performed the Semantic Web resource categorization by using our proposed method. To make the influence of each process clear, we measured the accuracy of the result of categorization after executing each process. However, since as mentioned before, the categorization flow is not sequential, we defined the method of extracting results for processing as follows.

First, we executed *Process* 1 only once. Then, we evaluated the result of categorization as the accuracy of *Process* 1. After that, we executed *Process* 1 and 2 until the categories became stable. Then the result was evaluated as the accuracy of *Process* 2. The accuracy of *Process* 3 was measured after performing *Process* 3 once and *Process* 1 and 2 repeatedly until the categorization became stable. The accuracy of *Process* 4 was also measured in the same way with *Process* 3.

While we calculated *Precision* and *Recall* by Eqs. (10) and (11) for the categorization results, A was the relevant resources that we manually made, B was the results of the categorization.

3.3.1 Overview of the Result

In the preliminary experiment, only 17% of Web resources can be categorized based on explicit relation in ontologies, but 58% of Web resources can be properly categorized by our proposed method. Furthermore, the data target of categorization in our research is a general Semantic Web resource, thus the Web resources having less valuable information for users need not be categorized. In other words, 100% is not necessary for the search system. In our experiment, 18% of Web resources have no valuable information among the 42% of Web resources that have not been categorized. Therefore, we can improve the categorization in the other 24% of Web resources. We consider that 58% for categorization based on the incomplete ontologies means a big improvement and we also believe that the result can be improved in future work. Since there was 75% of the Web resources whose resource types did not exist in the ontologies, we believe that it is a large improvement in categorization.

Precision and *Recall* for the categories in the four steps are shown in **Table 1**. As we can see from *Recall* by each step, some categories keep extending in *Processes* 2 and 3. Categories Plant, Project, Restaurant and Shop have no dramatic change in *Process* 3. Most categories have not much extension in *Process* 4 except the categories Animal, Location and Object. Some categories such as Animal, Object and Publication get higher *Precision* in *Process 2*, but *Precision* of categories Person and Organization decreased. *Precision* of most categories decreased in *Process* 3. In *Process* 4, most categories have not much change in *Precision* except categories Location and Object.

As we expected, the categorization method extracted a number of potential relations. Some examples are shown as follows. Resource types "http://sw.deri.ie/2004/08/pubs/swpubs#Article" and "http://www.daml.ri.cmu.edu/ontologies/cmu-ri-employmenttypes-ont#PhD_Student" have no relations defined in the ontologies, but they were categorized to categories Publication and Person in *Process* 2. Resource types "http://xmlns.com/wordnet/1.6/demolition" and "http://www.daml.ri.cmu.edu/ontologies/cmu-ri-employmenttypesont#Director" have

Table 1 Precision and Recall in four steps.

category	*Process* 1		*Process* 2		*Process* 3		*Process* 4	
	Precision	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*
Animal	46.4%	20.1%	58.4%	32.7%	50.7%	47.2%	49.0 %	47.8%
Event	100.0%	27.9%	94.5%	39.0%	46.2%	41.1%	42.5 %	41.1%
Food	100.0%	8.7%	100.0%	26.1%	100.0%	29.6%	100.0%	29.6%
Location	96.0%	10.1%	83.4%	50.2%	83.4%	55.9%	68.4 %	57.8%
Object	18.5%	0.9%	21.1%	14.6%	15.7%	26.6%	22.1 %	57.8%
Organization	80.3%	29.8%	65.3%	55.7%	60.6%	55.7%	54.6 %	67.8%
Person	78.1%	54.5%	65.6%	72.5%	62.0%	73.0%	60.3 %	73.0%
Plant	100.0%	30.0%	71.4%	50.0%	35.7%	50.0%	35.7 %	50.0%
Project	97.6%	96.8%	96.8%	96.8%	96.8%	96.8%	96.8 %	96.8%
Publication	64.4%	24.6%	73.2%	38.1%	72.7%	38.1%	68.4 %	38.9%
Restaurant	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Shop	100.0%	37.5%	83.3%	62.5%	75.0%	75.0%	75.0 %	80.0%

neither any relations defined in the ontologies nor any classes having the same class name in the ontologies, but they were categorized to categories Event and Person in *Process* 3. In *Process* 4, Resource type "http://www.aktors.org/ontology/portal#Researcher-Fellow-In-Academia" was detected as a potentially related class to category Person because the Web resources having this resource type were described by "http://www.aktors.org/ontology/portal#given-name" and "http://www.aktors.org/ontology/portal#has-web-address", which are property patterns for the category Person.

3.3.2 Influence of Each Process

From the categorization results, we found the following facts for each process of our proposed method.

When *Process* 1 categorized all the offspring classes, there were some noises for categorization. For example, some irrelevant classes (e.g. "Document" and "Project") were defined as a subclass of "Person". We realized that *Process* 1 easily decreases the precision of categorization. In another case, some classes were defined in a very large range of meanings to bring some noise to categorization. For example, "SpatialThing" was defined in such a large range that many classes of "Person", "Animal" and "Object" were defined as its subclass. In some ontologies, "SpatialThing" was also defined as a subclass of "Person", thus many classes of Animal and Object were categorized to Person. Therefore, we remove this kind of classes when we perform the categorization.

By optimizing threshold $minW$, *Process* 2 provides good category extension. There are many classes having not enough relations defined in the ontologies to be categorized directly. *Process* 2 brings some well defined classes to the categories for a deep categorization. In our experiments, it was effective for the categories such as Location, Person and Publication.

Process 3 is useful to extract potential relations that are not in the ontologies. However, as we mentioned before, words have many senses in WordNet. Thus, the relation definitions in the dictionary are not good for categorization in some categories. We found that *Process* 3 was especially efficient for the categories having clear boundaries with other categories. Therefore, *Process* 3 can also be performed only for some categories such as Food, Animal and Publication in a preliminary step. After performing the loop of *Process* 2 and *Process* 1, we could perform *Process* 3 for all the categories.

The number of categorized classes by *Process* 4 was not large, but it can categorize some very useful classes such as "http://www.aktors.org/ontology/portal#Researcher-Fellow-In-Academia" which can be used for categorizing other Web resources.

3.4 Categorization Based Semantic Web Search

We developed a Categorization-based Semantic Web Search (CSWS) system as a case study to verify the effectiveness of the proposed method.

While we calculated *Precision* and *Recall* for search results by Eqs. (10) and (11), A was the subset of relevant resources that can match the keywords by the search system, and B was the search results.

High *Precision* shows to what extent the results are returned correctly. However, to achieve a practical Semantic Web search engine for incomplete ontologies, it is important to return the related results that are well defined in the ontologies and the ones that have no definition in the ontologies but related to the user query. In other words, improving *Recall* of search result while not decreasing *Precision* too much is very important. Since an important fact is that increasing *Precision* often becomes a cause of decreasing *Recall*, we considered increasing *Recall* should come before that of *Precision*.

3.4.1 Evaluation Setting

Since a class with all of its offspring classes in the ontologies presents like a tree structure, we describe a class set having such a hierarchical structure as a class tree. We use a class tree based retrieval as a baseline. Users can specify a query by inputting a class name as a root of a class tree. Then the system looks into all the classes having the class name and all of their offspring classes for the search. In the category based retrieval (proposed method), users specify a category for searching, and the system will look into all the classes in that category. Then, users can input a keyword to start retrieval. The system returns all the related resources whose types are in the search range.

We chose some keywords to perform a Semantic Web search. In the category based retrieval, we performed several searches for each category by using different keywords. To make the comparison, we used the category name in our proposed method as the specified class name. We used the same keywords that are used in the category based retrieval.

As we need to examine the efficiency of the categorization method, we used the same data set described in Section 3.2, we did not consider the ranking of the result in this experiment.

3.4.2 Evaluation Results

Because the result of the class tree based retrieval is an ontology-based retrieval, the search result cannot go beyond the definitions in the ontologies, thus for some keywords, the system did not find any results for the queries which can get results by the category based retrieval. We only used the keywords that can get results by both of the methods to make the comparison in the same environment.

We used 10 keywords on average for each category and class name. Then we compared the average of *Precision* and *Recall* for these two methods. Since categories Restaurant and Shop have a few results to evaluate, we removed these results from the evaluation. **Figures 3** and **4** show the comparison for *Precision* and *Recall* of the search results. As we expected, the category based retrieval could return higher *Recall* than the class tree based retrieval. However, *Precision* of the class tree based retrieval is not always higher than that of the category based retrieval. For example, in categories Object and Person the search results showed higher *Precision* than the results in the class trees having root names as "Object" and "Person". For some categories such as Animal, Event and Location, though *Precision* of the class tree based retrieval is higher, *Recall* is much lower than the category based retrieval. Another important fact is that we only used the results for which the class tree based retrieval can return the search results. In fact, there are many cases when the search engine did not match the keyword by the class tree based retrieval, but matched the keyword by the category based retrieval. We found that our proposed method also provides search results in high *Precision* and *Recall* in such cases. Therefore, our proposed categorization method could not only provide high *Recall* but also keep high *Precision* for Semantic Web search.

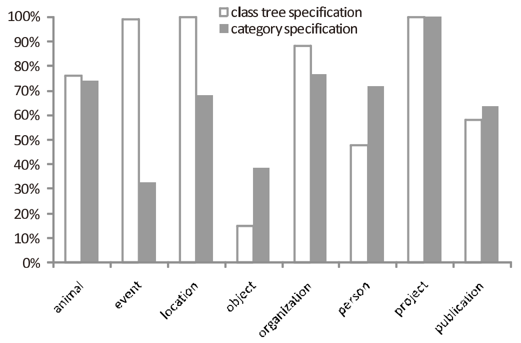

Fig. 3 Comparison for Precision.

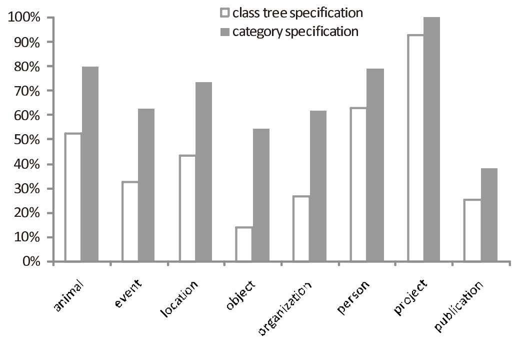

Fig. 4 Comparison for Recall.

4. Related Work

As a similarity-based Semantic Web search engine, Corese[1)] can provide an advanced query processing. Corese can search both the directly related and indirectly related Web resources. For example, assume that a user wants to retrieve organizations related to "human science". There is a member of an organization, and the member is interested in "human science". Then this organization can be considered as an organization related to "human science". Corese also contributed to improving *Recall* by using direct relation and proximate relation. Thus, for a well-structured ontology, Corese can present high *Precision* and *Recall*.

Currently, our Categorization based Semantic Web Search (CSWS) cannot find this kind of relation, because it provides a category matching method. However, our proposed method can categorize some Web resources whose types do not exist in the ontologies by detecting the probably-related resource types. When a user wants to retrieve organizations related to "human science", our CSWS could look into not only the Web resources defined as an organization but also some Web resources can be inferred as a kind of organization. For example, in our experiment data, for resource type "ns:WorkingGroup", we only know it has a parent class as "ns:Group". There is no relation between this resource type and the class organization defined in other ontologies. For such a case, this resource type is considered related to organization by Corese. However, by considering the relations between class names and categories, this resource type might be considered a kind of organization by our CSWS.

While it is impossible to construct a completed ontology, it is clear that it has a limitation for the methods based on well-structured ontologies. In contrast, however ontrary, CSWS provides high *Recall* in real world data by detecting the potential relations.

5. Conclusions

In this paper, we proposed a categorization method for Semantic Web resources by extracting implicit relation among them. By performing the evaluation experiments, we confirmed that applying the categorization method to a search engine can provide users with better search results than a query processing based on explicit relation analysis.

We also realized that inferring the relation based on the ontologies closely depends on the integration of the ontologies. In this research, we successfully inferred the well defined relations and detected the potential relations having no explicit definitions in the ontologies. Since the real world Semantic Web dynamically changes, it is impossible to construct a perfect ontology by only one organization. Thus, the extension of incomplete ontologies by detecting potential relations becomes more and more important. By the experiments using real Semantic Web resources described in Section 3.2, our proposed method is evaluated to be effective in using such incomplete ontologies. Therefore, we believe that our approach is a promising attempt for information processing in the real Semantic Web.

As part of our future work, we plan to study in more detail to unveil the factor which is effective for total accuracy. We need to optimize the parameters to improve the accuracy of the categorization.

Acknowledgments This research was supported in part by Grant-in-Aid on Priority Areas (18049050), and by the Microsoft Research IJARC Core Project.

References

1) Corby, O., Dieng-Kuntz, R. and Faron-Zucker, C.: Querying the Semantic Web with the Corese Search Engine, *Proc. ECAI/PAIS*, pp.705–709 (2004).
2) Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V.C. and Sachs, J.: Swoogle: A Search and Metadata Engine for the Semantic Web, *Proc. ACM Conf. on Information and Knowledge Management*, pp.652–659 (2004).
3) Eberhart, A.: Survey of rdf data on the web, Technical Report, International University in Germany (2002).
4) Guha, R., McCool, R. and Miller, E.: Semantic Search, *Proc. International Conf. on WWW*, pp.700–709 (2003).
5) Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W. and Musen, M.A.: Creating Semantic Web Contents with Protege-2000, *IEEE Intelligent Systems*, Vol.16, Issue2, pp.60–71 (1997).
6) Patel, C., Supekar, K., Lee, Y. and Park, E.K.: OntoKhoj: A Semantic Web Portal for Ontology Searching, Ranking and Classification, *ACM WIDM'03*, pp.58–61 (2003).
7) Salton, G. and McGill, M.: Introduction to Modern Information Retrieval, *McGraw-Hill Book Company* (1984).
8) Srikant, R. and Agrawal, R.: Mining Generalized Association Rules, *Proc. VLDB, 1995*, pp.407–419 (1995).
9) Google SOAP Search API.

http://www.google.com/apis/
10) http://d3e.open.ac.uk/akt/2001/ref-onto.html
11) Intellidimension, Semantic Web Search. http://www.SemanticWebsearch.com/
12) WordNet for the Web. http://xmlns.com/2001/08/wordnet/

(Received May 17, 2007)
(Accepted November 6, 2007)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.4, pp.103–113.)

Minghua Pei is currently pursuing a Ph.D. in the Graduate School of Information Science and Technology, Osaka University, Japan. She received a Master's degree from Shiga University, Japan, in 2003. Her research interests include data mining and Semantic Web mining. She is a member of IPSJ.

Kotaro Nakayama received his B.I. and M.I. from Kansai University, Osaka, Japan in 2001 and 2003. While he was a bachelor student, he launched an IT company named "Kansai Information Institute" and managed the company as the president and a director for three years. While he was a master course student, he also became a lecturer of the "Internet architecture" at Doshisha Women's College. After that, he received his Ph.D. in Information Science from Osaka University in 2007. His research areas are mainly AI and the WWW. He is especially interested in knowledge extraction from huge scale WWW contents. He is a member of ACM, IEEE, JSAI, IPSJ and IEICE.

Takahiro Hara received his B.E., M.E., and D.E. in Information Systems Engineering from Osaka University in Japan, in 1995, 1997, and 2000. He is currently an Associate Professor at the Department of Multimedia Engineering of Osaka University. His research interests include distributed database systems in advanced computer networks, such as high-speed networks and mobile computing environments. Dr. Hara is a member of ACM, IEEE, IEICE, IPSJ, and DBSJ.

Shojiro Nishio received his B.E., M.E., and Dr.E. from Kyoto University, Japan, in 1975, 1977, and 1980. He was with the Department of Applied Mathematics and Physics of Kyoto University from 1980 to 1988. In October 1988, he joined the faculty of the Department of Information and Computer Sciences of Osaka University. He became a full professor at the Department of Information Systems Engineering of Osaka University in August 1992. He has been a full professor at the Department of Multimedia Engineering at the same university since April 2002. He served as the founding director of the Cybermedia Center of Osaka University from April 2000 to August 2003, and served as the dean of the Graduate School of Information Science and Technology of this university from August 2003 to August 2007. He has been serving as a trustee and vice president of Osaka University since August 2007. His current research interests include database systems and multimedia systems. Dr. Nishio has served on the Editorial Board of *IEEE Transactions on Knowledge and Data Engineering* and *ACM Transactions on Internet Technology*, and is currently involved with the editorial board of *Data and Knowledge Engineering*. He is a fellow of IEICE and IPSJ, and he is a member of eight learned societies, including ACM and IEEE.