Knowledge Sharing in Virtual Community Based on RDF Triple Publication and Retrieving To Process SPQRQL Query

Yun Zhao
School of Information Engineering, Zhejiang Agriculture and Forestry University, Hangzhou 311300, China
Email: shilyze@gmail.com

Huayou Si
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
Email: sihy@hdu.edu.cn

Qing Lang
School of Information Engineering, Zhejiang Agriculture and Forestry University, Hangzhou 311300, China
Email: langqing@zjfu.edu.cn

Abstract—**Web ontologies, which usually appear as RDF graphs, are used to represent knowledge in web. In recent years, with the wide application of Semantic Web, large numbers of such web ontologies have appeared on the Internet, especially, in some virtual knowledge communities. These web ontologies are usually distributed in different sites and provide an amount of knowledge to query. But, it has become a pressing issue that, given a semantic query, how to efficiently gather the related knowledge from these web ontologies located in different sites to process it. To address this issue, in this paper, we propose a P2P-based approach to publish RDF triples in sharable web ontologies and freely sharing them in an open distributed environment. Given a query of SPARQL (Simple Protocol and RDF Query Language), this approach can automatically gather published RDF triples related to the query to process the query. As a knowledge sharing approach, our approach can directly share RDF triples coming from different web ontologies on different nodes. It overcomes limitations of those approaches which focus on how to locate related ontologies for a query. We also conducted three experiments to evaluate the effectiveness and the efficiency of our approach. The experimental results demonstrated that it is effective and efficient.**

Index Terms—**Knowledge Sharing, RDF triple, RDF graph, SPARQL Query, Structured P2P**

I. Introduction

Ontology is formal, explicit specification of a shared conceptualization to represent domain knowledge [1]. In Semantic Web, ontologies are used to represent knowledge in web. The ontologies are usually referred to as web ontologies. These ontologies are based on RDF (Resource Description Framework) so as to be called RDF graphs. In recent years, with the wide application of Semantic Web, large numbers of web ontologies have been developed and appeared in different sites on Internet. For example, we can search out tens of thousands of web ontologies by using Swoogle [2], a web ontology search engine. These ontologies possess a large quantity of knowledge to query. They can serve as sources of knowledge for web-based question answering system, recommendation system, etc [3, 4]. Especially in some virtual knowledge communities [5, 6], numbers of OWL ontologies are also created. In such a community, each member usually creates one or more ontologies to represent his/her own knowledge of a given domain. These ontologies also possess a large quantity of knowledge to be shared and leveraged by members in the community for their own purposes.

But, it has become a pressing issue that, given a semantic query, how to efficiently locate the ontologies, from which some solutions can be reasoned out for the query, within a virtual community. In recent years, the issue has been given a great deal of attention in practice as well as in research.

Most of current approaches are based on Client–Server (C/S) structure. In these approaches, all knowledge resources, such as ontologies, are gathered and stored in some centralized knowledge servers. Users can query and utilize knowledge under some sort of centralized control. These approaches have been considered inappropriate and ineffective to share knowledge [4, 7] and are not suitable for the autonomous and dynamic characteristics of knowledge sharing [8, 9]. So, knowledge sharing in a decentralized network, especially supported by peer-to-peer (P2P) technology, is introduced. Given a query, these approaches either route it to the nodes with related ontologies to construct solutions for it, or locate the related ontologies and then send the query to them to construct solutions respectively. In these approaches, knowledge sharing is essentially ontology sharing. Knowledge sharing is based on ontology, not on the knowledge itself in ontology. So, these approaches do not run well in some cases, such as, though we cannot construct any solutions for a query from ontology A or ontology B respectively, yet we can from knowledge together in ontology A and B; or we can construct more solutions from knowledge together in ontology A and B than from ontology A and B respectively. These are common cases, especially in a virtual community where some ontologies usually possess some correlative knowledge.

To address the issue and overcome the limitations of current approaches, we propose a distributed approach to directly publish and share RDF triples in web ontology and implement it based on Chord [10], a structured P2P protocol. In our approach, if a node has sharable web ontologies, it can publish all the RDF triples in these ontologies. If a node receives a query of SPARQL (Simple Protocol and RDF Query Language) [11], it can efficiently retrieves the web ontology related to the query and creates a temporary ontology to process it.

As matter of fact, in this paper RDF triple is viewed as a minimum and independent unit of knowledge. Knowledge sharing in our approach is based on each RDF triple in web ontology rather than an entire ontology. If necessary, related RDF triples in different ontologies locating in different nodes can be gathered to process a query. Our approach overcomes limitations of those approaches which focus on how to locate related ontologies for a query. It enables user to automatically share knowledge for his/her SPARQL query processing in open distributed environment. We also conducted three experiments to evaluate the effectiveness and the efficiency of our approach. The experimental results demonstrated that our approach is effective and efficient.

The rest of the paper proceeds as follows. Section 2 discusses related basic ideas and outlines our approach. Section 3 presents the algorithm of RDF triple publication. Section 4 presents the algorithms of RDF triple retrieving. Section 5 addresses our experiments to evaluate our approach. Section 6 presents related work. Section 7 draws a conclusion.

II. Basic Idea and Overview of Our Approach

In this section, first we introduce P2P networks and their application in our approach. Then we discuss OWL ontology, SPARQL query, and related basic ideas for knowledge publication and retrieving. Finally, we present overview of our approach.

A. P2P Networks and Their Application

Peer-to-peer (P2P) networks are distributed systems, which consists of large numbers of autonomous nodes (also called peers) and allows the sharable resources of each node to be accessed by others in an open distributed environment. P2P systems usually do not need any hierarchical organization or centralized control. They overcome the deficiencies of centralized registration system and possess the properties, such as fault-tolerance, self-organization, and scalability [24]. According to different resource lookup mechanisms, P2P networks can be classified into two categories: Structured and Unstructured. Unstructured P2P networks organize nodes into a random graph and use flooding or random walks on the graph to query sharable resources provided by some nodes. In most cases, the routing styles are inefficient in large-scale network. Structured P2P networks usually organize the nodes into an orderly graph in a systematic way. For any sharable resource on any node, they can assign a given node responsibility for it. Thus, structured P2P networks can achieve very efficient lookup mechanism so that they can provide very good scalability.

For example, as a typical structured P2P technique, Chord [10] uses consistent hashing [13] to assign each node a key in system. And then, based on the order of the keys, it organizes the nodes into an orderly ring, where the node with maximum key connects with the node with minimal key. For a sharable resource *r* with a property *p* of any node in Chord, first Chord uses the same hashing to assign property *p* a key *k*. Then it locates a node *N*, which key is greater than *k* and closest to *k* than all the others. Finally, it saves the property *p* and the resource *r* as a pair <*p*, *r*> on the node *N*. Usually *r* is URI of the resource. This process is called resources publication.

Thus, given a property *p* of desired resources, according to the key *k* of property *p*, Chord can efficiently locate the node *N* assumed responsibility for key *k*. Then, it takes out all the pairs which involve in the property *p* on the node *N* to further find out the corresponding resources. The process is called resources discovery. In fact, Chord's lookup mechanism is very effective. It can find resource using only log(n) messages, where n is the number of nodes in the system.

If a node joins Chord, it will be inserted into the orderly ring according to its key and undertake part of burdens of the node next to it. If a node leaves, it turns over what it undertakes to the node next to it. In case of node failure, each resource is usually published on several nodes. Chord is provably robust in the face of frequent node failures and re-joins [16]. It can provide very good scalability and failure resilience.

Because of Chord with these strengths of resource publication and retrieving as a structured P2P network, we apply it to our approach to organize virtual community for knowledge sharing.

B. RDF Graph and Basic Idea for Triples Publication

Resource Description Framework (RDF) is a framework for representing information in the Web, which is designed to represent information in a minimally constraining, flexible way. The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate (also called a property) and an object. This can be illustrated by a node and directed-arc diagram, in which each triple is represented as a node-arc-node link. The nodes of an RDF graph are its subjects and objects. The direction of each arc always point toward the object. So, a collection of such triples is called an RDF graph. Each triple represents a statement of a relationship between the things denoted by the nodes that it links. The assertion of an RDF triple says that some relationship, indicated by the predicate, holds between the things denoted by subject and object of the triple. Therefore, a RDF triple can be regarded as minimum knowledge unit, which usually present a whole relationship between things. The assertion of an RDF graph amounts to asserting all the triples in it. The meaning of an RDF graph is the conjunction of the statements corresponding to all the triples it contains

RDF properties are thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. They also represent relationships between resources. RDF however, provides no mechanisms for describing these properties, nor does it provide any mechanisms for describing the relationships between these properties and other resources. That is the role of the RDF vocabulary description language, RDF Schema (RDF-S), which is a semantic extension of RDF. It specifies mechanisms that may be used to name and describe properties and the classes of resource they describe. RDF Schema provides built-in resources and properties for describing groups of related resources and the relationships between these resources in domain being described. So, they are used to determine characteristics of other resources, such as the domains and ranges of properties.

In addition, as ontology language based on RDF, OWL (Web Ontology Language) is used to formally define meaning to facilitate machine interpretability of Web content. It is supported by RDF and RDF-S by providing additional vocabulary along with a formal semantics. So, web ontologies, which are in compliance with OWL or RDFS, are all based on RDF data model. They are also viewed as RDF graph, which consists of the following four different syntactic ingredients:

- **Entities**, such as classes, properties, and individuals, are identified by IRI references. They form the primitive terms of ontology to express the basic notions in domain. For example, the class *http://www.example.com/onto.owl#Person* can represent the set of all people. It can be abbreviated as *a: Person*, where a: denotes the name space: *http://www.example.com/onto.owl#*.
- **Literals** are used to identify values such as numbers or dates by means of a lexical representation. A literal may be the object of a RDF triple, but not the subject or the predicate.
- **Language vocabularies** are provided by RDF Schema or OWL to name and describe entities in domain being described. They are also identified by IRI references, such as *http://www.w3.org/1999/02/22-rdf-syntax-ns#type*.
- **Anonymous entity** (also called Blank RDF node) is not an IRI reference or a literal. In the RDF abstract syntax, an Anonymous entity is just a unique RDF node that can be used in one or more RDF statements, but has no intrinsic name.

An IRI reference used as a predicate identifies a relationship between the things represented by the RDF nodes it connects. A convention used by some linear representations of an RDF graph to allow several statements (i.e., RDF triples) to reference the same unidentified resource is to use a blank RDF node identifier, which is a local identifier that can be distinguished from all IRIs and literals. When graphs are merged, their blank RDF nodes must be kept distinct if meaning is to be preserved.

As mentioned above, web ontology can be viewed as a collection of RDF triples which compose of entities, literal, language vocabularies, and blank RDF node, represented as a subject, a predicate, and an object. Therefore, for each triple, we can take each entity appearing in it as an index of the triple. Then, based on each index of a triple, we publish and retrieve it on structured P2P network. This is our idea to publish triples and share them. We discuss it in detail in subsection 2.4.

C. SPARQL Query and Basic Idea for Knowledge Retrieving

As W3C Recommendation, SPARQL [11] is a query language for RDF graphs. since web ontologies in compliance with OWL or RDFS [12] are all based on RDF data model, they are essentially a RDF graphs and can be queried by SPARQL. Each SPARQL query has a graph pattern which consists of one or more pattern-clause. Each pattern-clause usually includes several entities, which appear in it. For example, given a graph pattern in Figure 1, all the entities in it can constitute a collection {*dc:book*, *dc:title*, *dc:creator*, *dc:corporation*}.

As matter of fact, a graph pattern can be automatically converted into a semantically equivalent triple pattern. For example, the result of such conversion is shown from Figure 1 to Figure 2. The triple in a triple pattern is like RDF triple except that each of the subject, predicate and object may be a variable.

```
{ [ a dc:book ] dc:title "Semantic Web"; dc:creator ?y.
?y a dc:corporation.
}
```

Figure 1. Graph Pattern of a SPARQL Query

```
    {
        _:b0    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
dc:book .
        _:b0 dc:title "Semantic Web" .
        _:b0 dc:creator ?y .
        ?y      <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
dc:corporation
    }
```

Figure 2. Triple Pattern of Graph Pattern in Figure 1

Graph pattern is a core component of SPARQL query. It is used to match a sub-graph of the RDF graph being queried when RDF terms (included entities, literals, and language vocabularies) from that sub-graph may be substituted for the variables in the graph pattern, and the result, i.e., the solution of the query, is RDF graph equivalent to the sub-graph.

So, given a SPARQL query, if a RDF graph can be queried out results for the query, RDF terms appearing in its graph pattern must be appear in the RDF graph. Accordingly, given a query, these triples related to the RDF terms, which appear in the graph pattern of the query, are usually useful to process the query.

Thus, based on the foregoing idea of RDF triple publication and retrieving, given a SPARQL query, we can retrieve related triples for the query based on the entities, i.e., RDF terms, which appear in the query's graph pattern.

D. Overview of Our Approach

As mentioned above, we apply structured P2P protocol Chord to our approach to organize virtual community for knowledge sharing. First of all, we design two functions to publish and retrieve triples in web ontology on P2P network as follows:

- ***pubTriple(entity, triple)***, the function is used to publish a *triple* based on an *entity*. As mentioned above, Chord assigns *entity* a key and finds a P2P node based on the key. Then, saves the pair <*entity*, *triple*> on this node.
- ***retrTriple(entity)***, the function is used to get all the *triples* that are published based on *entity* by any P2P node in Chord.

In our approach, the P2P nodes in a virtual community constitute a Chord P2P network first. When a P2P node with some sharable ontologies joins, it publishes them as follows:

1. Given a sharable ontology *O* in a P2P node, changes it into a collection of serializable triples as *tplSet* according to some strategies. Here, each triple is regarded as minimum knowledge unit.
2. For each triple *tpl* in collection *tplSet*, extracts all entities appearing in triple *tpl* as an entity set *enSet*.
3. For each entity *en* in set *enSet*, publishes triple *tpl* based on entity *en* by using function *pubTriple(en, tpl)* as discussed above.

This method publishes each triple according to these entities which appear in this triple as subject, predicate, and object respectively. Thus, all P2P nodes know which P2P nodes are responsible for triples they are looking for. It guarantees to find out matched triples in the network if the triples exist.

Once a node *N* receives a SPARQL query *Q*, according to the idea as discussed in subsection 2.3, it can retrieve all related triples as temporary web ontology to process the query as follows:

1. Node *N* Parses graph pattern of query *Q* and extracts all the entities as a collection *enSet*.
2. according to each entities *en* in collection *enSet* retrieves enough related triples from virtual community for query *Q* by using function *retrTriple (en)* based on some strategies.
3. Creates a temporary web ontology *O* which holds all the triples being retrieved.
4. Reasons ontology *O* to construct solutions for the query.

The specific strategies to changes web ontology into a collection of triples and retrieve related triples for a query is discussed in detail in the following section.

III. Strategy and Algorithm of RDF Triple Publication

In a RDF graph, entities are the primitive terms of ontology to express the basic notions in domain. Entities compose of classes, properties, and individuals, which are identified by IRI references respectively. Therefore, entities in RDF graph represent the notions and terms in a domain being described and reflect the category of domain knowledge to a certain extent.

Language vocabularies, which are provided by RDF Schema or OWL, are meta-language elements and almost appear in every ontology. So, they cannot reflect the category of domain knowledge. Similarly, because literals are just used to identify values such as numbers or dates by means of a lexical representation, they cannot reflect the category of domain knowledge too.

Hence, in order to efficiently retrieve related triples as knowledge needed to process a SPARQL query, we can publish a RDF triple based on the entities appearing on it. However, there is a type of entities known as anonymous entities. When a convention is used by some linear representations of an RDF graph, they are assigned a local identifier respectively, which can be distinguished from all IRIs and literals. In fact, an anonymous entity in RDF graph represents an anonymous notion, which is used to link multiple concepts to express a complex relationship, i.e., a piece of complex knowledge. This is to say, anonymous entity in RDF graph is used to link several RDF triples. So, RDF triple should be published based on anonymous entity appearing in it so as that related triples can be retrieved.

Because anonymous entities must be kept distinct if meaning is to be preserved when graphs are merged, here we present a method to assign an anonymous entity an unique IRI reference: Given an anonymous entity, we connect IRI of the RDF graph where this anonymous entity locate, character '#', and this entity's local identifier as its IRI reference. For example, if a FDF graph, which IRI is http://www.example.com/onto.owl, have an anonymous entity, which local identifier is *_:personZhao*, according to our method, its IRI reference generated is *http://www.example.com/onto.owl#_:personZhao*. Because the IRI reference of this RDF graph is unique and the entity's local identifier is unique in this RDF graph too, this anonymous entity's IRI reference generated is unique. In this paper, for each anonymous entity in a RDF graph to be published, a unique IRI reference is generated for it.

```
Algorithm publishTriple(onto)
Input onto: ontology, i.e., RDF graph onto to be published
Output null: there is nothing to return
{
takes out all the  specified triples in ontology onto as triple set triSet;
  for( each triple tpl in triple set triSet){
takes out a subject, a predicate, and an object from triple tpl as RDF term set rsSet;
for(each RDF term rs in set rsSet){
if(rs is anonymous entity){
          Generate IRI reference rs_iri for rs;
}else if(rs is named entity){
          Extracts rs's IRI rs_iri;
      }
    publish tpl based on IRI rs_iri by using pubTriple(rs_iri, triple);
  }
}
}
```

Figure 3. Algorithm: *publishTriple*

According to discussions above, we design RDF graph publication algorithm: *publishTriple* based on function *pubTriple (entity, triple)* in subsection 2.4, shown in Figure 3. The basic idea is that, given a RDF graph, first extracts all the RDF triple specified in it; then for each entity in a triple, if it is an anonymous entity, generates an IRI reference, else takes out its IRI reference to publish the triple.

The number of accesses to P2P network is quantity of the named and anonymous entities in all the RDF triple in web ontology when it is published.

IV. Strategies and Algorithms of RDF Triple Retrieving

In this section, first we discuss the strategies and several concepts for RDF Triple Retrieving. Then present the algorithms of RDF triple retrieving.

A. Basic Idea of RDF Triple Retrieving

Given a SPARQL query, if a RDF graph can be queried out results for the query, RDF terms appearing in the graph pattern must appear in the RDF graph. Moreover, if graph pattern of the query is converted into triple pattern, each triple in the triple pattern can be matched in the RDF graph. This is to say, for each RDF term in the graph pattern and its role as one of the subject, predicate, or object in a given triple in the triple pattern, there is at least one triple specified or implied in the RDF graph, which involves in the RDF term as the same role.

Therefore, Given a SPARQL query *Q*, these triples should be retrieved from virtual community based on structured P2P network, which can match the triples in triple pattern of query *Q*, or which can reason out the triples that can match the triples in triple pattern of query *Q*.

If a triple *T* in a graph pattern of query can match with RDF triple *T'*, out of question, the subject, predicate, and object of *T* and *T'* are corresponding to each other respectively. That is, the entity in the triple *T*, also appears on the corresponding role in triple *T'*. Thus, only these triples are possible matched with triple *T*, where an entity *e* appears if triple *T* contains an entity *e*. Moreover, the triples matched with triple *T* can be reasoned out only from such triples, which is included in the RDF unicom sub-graph of the entities appearing in triple *T*, because these triples where entity *e* appears cannot reason out from a RDF graph without entity *e*. In a similar way, if a RDF graph contains several independent unicom sub-graph and entity *e* does not appear in one of these RDF unicom sub-graphs, these triples where entity *e* appears cannot reason out from this RDF unicom sub-graph. In fact, if entity *e* is a property, it maybe appears in several independent unicom sub-graph. In our approach, if more than one independent unicom sub-graphs in a RDF graph share an entity as property, they are regarded as one RDF unicom sub-graph.

Therefore, if all the specified and potential RDF triples where entity *e* appears need to be retrieved and reasoned out, the independent unicom sub-graph where where entity *e* appears should be retrieved. This is, all the triples which locate in the unicom sub-graph should be obtained. In our approach, this independent unicom sub-graph of entity *e* is called unicom triple set, recorded as CSG_e. Given a context, CSG_e contains all the triples, where entity *e* locates, or which can reason out the triples where entity *e* locates. Graph pattern containing entity *e* of query can be matched just against these triples.

So, given a SPARQL query *Q*, according to each entity appearing in graph pattern of query *Q*, their *CSG* can be gotten respectively, all the triples in these *CSG* together are called unicom triple set of query *Q*, recorded as CSG_Q. If their biggest unicom subgraphs can be gotten in a given context, all the triples in these subgraphs are called biggest unicom triple set of query *Q*, recorded as $MCSG_Q$. In fact, if $MCSG_Q$ is taken as web ontology, it can process SPARQL query *Q* effectively.

Given a SPARQL query *Q*, if $MCSG_Q$ can be gotten in a given context, such as a given virtual community, all potential solutions in whole sharable knowledge can be reasoned out. Because $MCSG_Q$ includes all direct and indirect related triples of SPARQL query *Q*, just from which solutions of query *Q* can be reasoned out. The triples, which are not included in $MCSG_Q$, cannot contribute to construct solutions for query *Q*. Here we do not discuss it in detail.

B. Algorithms of RDF Triple Retrieving

According to the above analysis, using function *retrTriple* discussed in subsection 2.4, we design the algorithm *retrieveRldTriple* shown in figure 4 to retrieve *CSG* of a SPARQL query based on web ontology publication algorithm: *publishTriple* in figure 3. The strategy of the algorithm *retrieveRldTriple* is that: Firstly, give a SPARQL query *Q*, takes out all the entities appearing in graph pattern of query *Q* as entity set *entSet*. Then, according to each entity in set *entSet*, retrieves the published RDF triples by using function *retrTriple* discussed in subsection 2.4 as a RDF triple set *tripleSet*. Without question, at least one entity in set *entSet* appears in a RDF triple in set *tripleSet*. Moreover, triples in set *tripleSet* are centered in the entities in set *entSet* as several *CSGs* respectively. This is the first iteration. Secondly, according to the new entities introduced by the triples in set *tripleSet* (that is, the entities which appear in a triple of set *tripleSet*, but do not belong to set *entSet*), continues to retrieve RDF triples as part of set *tripleSet* (that is, further expends each *CSG*). This is the second iteration. So iteration, until there are not new entities to be introduced, $MCSG_Q$ is gotten, from which all potential can be reasoned out for query *Q*.

If we only focus on the access to P2P network, the access number is the quantity of elements in entity Set *enSet* in $MCSG_Q$ in algorithm: *retrieveRldTriple*. Though we can reason out all potential solutions for a query *Q* based its $MCSG_Q$ as mentioned above, it may be difficult to get the $MCSG_Q$ in large-scale knowledge communities because there may be a mass of related triples and need great numbers of accesses to P2P network. In practice, users sometimes do not need all possible solutions of their queries. Thus, for a query we can just retrieve a small CSG_Q according to a limited number of iterations of the algorithm: *retrieveRldTriple*. This method to retrieve a small CSG_Q for a SPARQL query usually has high performance, although it cannot guarantee to obtain all potential solutions for the query.

```
1. Algorithm retrieveRldTriple
2. Input enSet: the entities Set, each element in it appears in graph pattern of SPARQL query Q
3. Output tripleSet: $MCSG_Q$ returned
4. {
5.   initializes empty Lists levelA, levelB, current, and constru;
6.   puts elements in Set enSet into List levelA;
7.   current= levelA; constru= levelB;
8.   while(true){
9.     for(each entity en in List current){
10.      retrieves RDF triples as a Set triSet by using function retrTriple (en);
11.      puts triples in triSet into tripleSet;
12.      for(each triple triple in Set triSet){
13.        takes out each entity appearing in triple as entity Set entitySet;
14.        for(each entity ent in entity Set entitySet){
15.          if(entity ent is not belong to enSet)
16.            puts entity ent into entity Set enSet and List constru;
17.        }
18.      }
19.    }
20.    if(list constru is empty) break;
21.    if(current = =levelA){
22.      current=levelB; constru=levelA;
23.    } else{
24.      current=levelA; constru=levelB;
25.    }
26.    clear List constru;
27. }
28. return axSet;
29. }
```

Figure 4. Algorithm: *retrieveRldTriple*

However, if just a small *CSG* is retrieved for a SPARQL query rather than *MCSG* of the query, the *CSG* may contain incomplete knowledge because some triples are probably not able to express complete meaning. For example, if a *CSG* of a SPARQL query *Q* contains a triple "ex: myPage. HTML, ex: creator, _ : personSi", which introduces an anonymous entity "_ : personSi", but does not contain the triples to describe the anonymous entity "_ : personSi" in a given context, such as "_ : personSi, ex:name, Huayou Si" and "_ : personSi, ex:email, sihy@live.cn", the knowledge in *CSG* is considered to be incomplete. As matter of fact, anonymous entity usually is not an object to be queried, but just links multiple triples to express a piece of complex knowledge. Thus, if an anonymous entity loses the triples that it links, appearance of the anonymous entity makes no sense.

Therefore, if some anonymous entities exist in a *CSG* which is retrieved for a SPARQL query, some related RDF triples which describe these anonymous entities must be retrieved to add to the *CSG* so as to make the knowledge in it complete.

Here, we design an algorithm: *bcCSG*, shown in figure 5, to retrieve related RDF triples for the anonymous entities in a *CSG* for query *Q*. Its strategy is: extracts all the anonymous entities in a *CSG*, which description RDF triples have not been retrieved, as a List *anoList*. Then, takes out each anonymous entity *ano* to retrieve its description RDF triples by using *retrTriple (ano)* to add to *CSG*. If some new anonymous entities are introduced by the RDF triples retrieved, puts them into List *anoList*. The algorithm ensures that each anonymous entity in CSG_Q gets all the triples it links.

```
1. Algorithm bcCSG
2. Input csgSet: $CSG_Q$ of SPAQRL query Q
3. Output csgSet: $CSG_Q$ which complete knowledge
4. {
5.   initializes empty Lists anoList;
6.   extracts all the anonymous entities without description triples in a csgSet and puts them into List anoList;
7.   for(each anonymous entity ano in List anoList){
8.     retrieves RDF triples as a Set triSet by using function retrTriple (ano);
9.     takes out each anonymous entity in triSet as entity Set anoSet;
10.    for(each anonymous entity an in Set anoSet){
11.      if(anoList does not contain an){
12.        puts an into anoList;
13.      }
14.    }
15.    puts RDF triples in Set triSet into csgSet;
16.  }
17.  return csgSet;
18. }
```

Figure 5. Algorithm: *bcCSG*

If we only focus on the access to virtual community, the access number is the quantity of elements in *csgSet* in algorithm: *bcCSG*.

V. Evaluation

We design the following three experiments to evaluate effectiveness and efficiency of our approach:

- The first experiment evaluates the ability of our approach's callback, i.e., the ability of obtaining potential solutions for a given query in a given context, such as in a virtual knowledge community.
- The second experiment reveals the specific effects on obtaining potential solutions for the query, which is based on a query's different *CSG*s according to different iterations by Algorithm: *retrieveRldTriple* in Figure 4. Usually, a *CSG* with more iterations is retrieved for a query; more solutions for the query probably can be gotten. This experiment reveals the specific relationship between them to gain a better understanding of the efficiency of our approach.
- The third experiment evaluates the efficiency of our approach by the number of the access to network when RDF graph, i.e., WEB ontology, is published, or a SPARQL query is processed.

A. Experiment Set Up

To evaluate our approach, we have implemented it. In this implementation, we apply the open source development kits: Jena [14], Pellet [15], and open-chord [16]. Jena is a Java framework to provide a programmatic environment for building Semantic Web applications. Pellet is a Java-based OWL DL reasoner, which can be used in conjunction with Jena libraries. Open Chord is a Java-based implementation of the Chord DHT [10]. It provides an interface for Java applications to take part as a node to construct a structured P2P network.

In our experiments, we use the OWL ontologies and SPARQL queries of the third party as experimental data, which locate in “pellet-2.2.2\examples\data”, i.e., the example data in the open source development kits “pellet-2.2.2”. The experimental data provides several OWL ontologies and dozens of SPARQL queries.

B. Results and Analysis

- ***Experiment 1*** is conducted to evaluate our approach’s ability of obtaining potential solutions for a given query.

In this experiment, first of all, we find out the number of the potential solutions of each query in our experimental data. First, we merge all OWL ontologies in our experimental data as a new ontology *O*. Then, for each query in our experimental data, we reason ontology *O* to find out the number of its solutions. Obviously, the number of a query’s solutions is the query’s potential solution number based on the knowledge that all these OWL ontologies in our experimental data provide. The results are shown in line Potential in table. 1.

We use our approach (*M1*) to publish all web ontologies and process all our queries in experimental data based on their $MCSG_Q$ obtained by Algorithm: *retrieveRldTriple* in figure 4. And then, record the number of each query’s solutions returned by our approach (*M1*) in line Returned in table. 1.

Figure 3 shows that, the numbers in line Returned are identical to numbers in line Potential respectively. This is, our approach (*M1*) can get all the potential solutions for a given query in a given context. It is because, for a query, although related RDF triples are divided into different P2P nodes, our approach *M1* can freely retrieve them and put them together to process the query.

- ***Experiment 2*** is conducted to explore the specific effects of a query’s different CSG according to corresponding iterations on obtaining potential solutions.

In this experiment, first we publish all OWL ontologies in our experimental data using our approach (*M1*). Then, for each query, we retrieve its *CSGs* of all possible iterations by Algorithm: *retrieveRldTriple* respectively. Next, for each query *q*, we get the numbers of query *q*’s solutions respectively based on each *CSG* of query *q*. Their results are recorded in table. 2. For example, the number in line I3 and column Q8 means the number of the returned solutions when query Q8 is processed based on a *CSG* which is obtained by Algorithm: *retrieveRldTriple* according to 3 iterations.

TABLE 1.
THE NUMBERS OF SOLUTIONS RETURNED OF EACH QUERY BASED ON DIFFERENT CSG

Iterations	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*	*Q6*	*Q7*	*Q8*	*Q9*	*Q10*	*Q11*	*Q12*
I1	0	41	0	17	0	0	0	0	1	1	1	0
I2	571	41	34	17	41	0	0	0	1	1	1	0
I3	678	41	34	17	80	0	0	2	3	3	4	0
I4	678	41	34	17	80	0	0	2	3	3	9	1
I5	678	41	34	17	80	78	39	2	3	3	9	1
I6	678	41		17	80	78	39	2	3	3	9	1
I7	678	41		17		78	39	2	3	3	9	1
I8								2	3	3	9	1
I9								2	3	3		
I10								2	3	3		
I11								2	3	3		
I12								2	3	3		
I13										3		
I14												

This table shows that, based on respective *CSG* on one iteration, 2 queries get their all potential solutions; based on respective *CSG* on three iterations, 8 queries get their all potential solutions; until based on respective *CSG* on five iteration, all queries get their all potential solutions.

In addition, we calculate the percentage of returned solutions against all the potential solutions of all the queries based on their own *CSG* on their each iterations. The results are shown in Figure 6.

The Figure 6 shows that, based on the *CSG* with one iteration by Algorithm: *retrieveRldTriple* respectively, 6.2% of solutions of all queries are gotten; based on the *CSG* on two iterations, the percentage is 71.8%; based on the *CSG* on five iterations, the percentage is 100%. The results are amazing. They mean that, given a query, it is highly possible to get most solutions in a given context just based on a small *CSG*. It is because that the semantics of an entity *et* is mainly associated with the entities (as set *enSet*), which sometimes appear in same RDF triples of entity *et*. *CSG* with *2* iterations contains all the triples. So, the results are reasonable.

TABLE 2.
THE NUMBERS OF POTENTIAL SOLUTIONS AND RETURNED BY OUR APPROACH

Query	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*	*Q6*	*Q7*	*Q8*	*Q9*	*Q10*	*Q11*	*Q12*
Potential	678	41	34	17	80	78	39	2	3	3	9	1
Returned	678	41	34	17	80	78	39	2	3	3	9	1

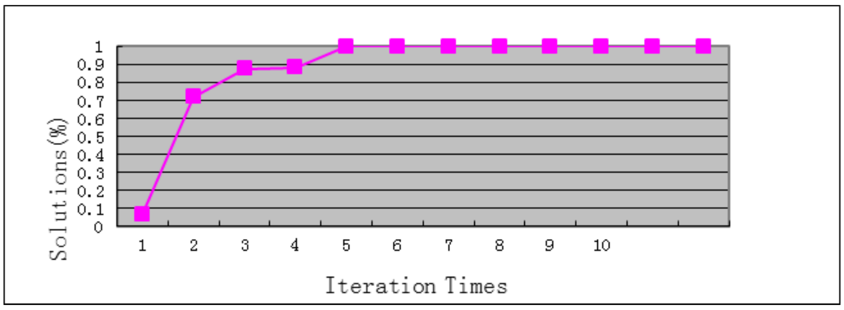

Figure 6. Percentages of Returned Solutions Based on Respective CSG According to Each Iteration

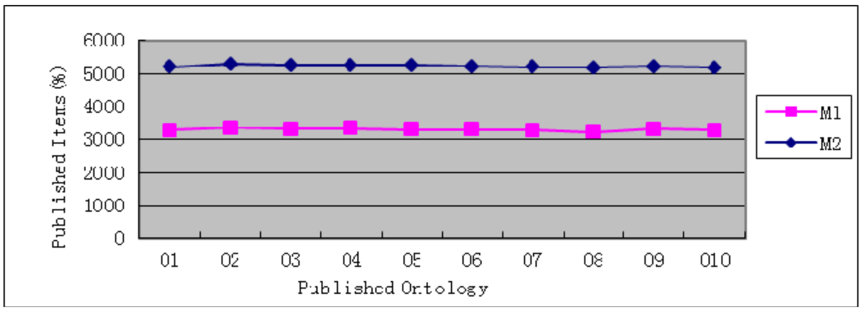

Figure 7. Number of the Accesses to Network When Each RDF graph Is Published

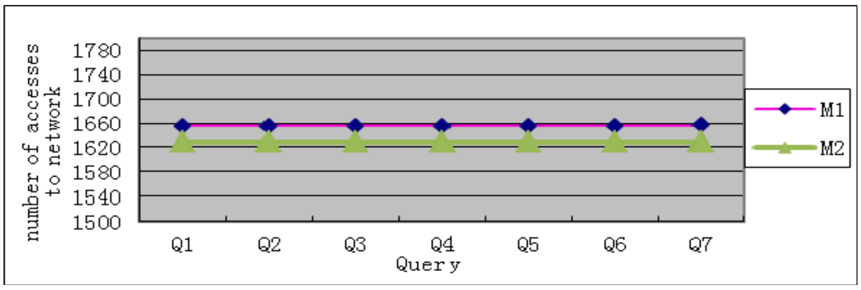

Figure 8. Number of the Accesses to Network When Each SPARQL Query Is Processed

This results show our approach is essentially efficient. For a query, usually it is unnecessary to retrieve its *MCSG* even if all solutions are needed.

- ***Experiment 3*** is conducted to evaluate the efficiency of our approach by the number of the accesses to network when a RDF graph is published, or a SPARQL query is processed.

Similar to our approach (*M1*), Si et al [39] propose a P2P-based approach (*M2*) to publish axioms in sharable web ontologies and freely share them to process SPARQL query in virtual community. Given a query of SPARQL, this approach *M2* can gather related axioms so as to reason out all potential solutions for the query. In this experiment, we compare the efficiency of the approach *M1* and *M2* to evaluate our approach *M1*.

To simulate a distributed virtual community, first we merge university0-0.owl and univ-bench.owl in our experimental data as a new ontology *O*. Then, randomly divide ontology *O* into 10 ontologies and deploy them on different nodes. Finally, we conduct our tests. This is, by using our approach *M1* and *M2* respectively we publish these ontologies and record the number of their accesses to P2P network in process of each ontology publication. These results are shown in figure 7.

The Figure 7 shows that the number of approach *M1*'s accesses to the network is more than approach *M2* when a web ontology is published. It is because that an axiom in web ontology usually represents a piece of complex knowledge, which can break down into several RDF triples. Each of RDF triple usually is published three times according to its subject, predicate, and object by our approach *M1*, while approach *M2* just publishes each axiom based on each entity appearing on it.

In this experiment, by using approach *M1* and *M2* respectively, we further process the first seven SPARQL queries in our experimental data (because the published ontologies have nothing to do with the rest of queries). Then, we record the number of their accesses to P2P network in process of each query. This experimental result is shown in figure 8.

The Figure 8 shows that the number of approach *M1*'s access to the network is slightly more than approach *M2* when a SPARQL query is processed. It is because these two approaches retrieve RDF triples or axioms just based on entities appearing on graph pattern of the query being processed and triples or axioms returned. But, approach *M1* also considers related anonymous entities. So, approach *M1* needs more accesses to the network than approach *M2*. But the difference values are not big.

From these three experiments, it can be seen that our approach is effective and efficient. If necessary, it can get all potential solutions for a query in a virtual community. If not necessary, it can get some solutions for a query within a small number of accesses to P2P network. Although it will consume more resource than approach *M2* when web ontology is published or query is processed. But, the granularity of knowledge to be published and sharing by approach *M1* is smaller than approach *M2* (because publication and retrieving of approach *M1* is RDF triples, while approach *M2* is OWL axioms). So, approach *M1* is has more flexibility than approach *M2*. In addition, approach *M2* just processes OWL ontology, while approach *M1* can handle all kinds of RDF graphs.

VI. Related Work

Along with the development of P2P techniques and the technical requirement of knowledge sharing in virtual community, in recent years some P2P-based approaches for ontology publication and discovery have been proposed. The current research works can be divided into three categories. The works in the first category focuses on P2P-based web ontology publication and semantic query routing; the second focuses on distributed storage and management of web ontology; and the third category mainly focuses on distributed management and sharing of knowledge in virtual community. The typical works of the first category are listed as follows:

Earliest of all, Tian et al. [19] present an ontology-based P2P lookup service (named SemanticPeer), which extends Chord protocol with express table and index table to publish resources in common ontology and private ontologies respectively. With the approach, a semantic query can be routed to a node which contains a private ontology to process it. Similarly, Gao et al. [20] publish classification information of a resource to P2P. Then, based on the classification of target resource, it can find out the peers where a kind of resource maybe locates in. In essence, these approaches must be based on common knowledge base or unified classification.

Raul Palma1 and Peter Haase [17] provide an approach (named Oyster) to exchange and re-use ontologies based on P2P network. Oyster provides an infrastructure for storing, sharing, and finding ontologies making use of the proposal for Ontology Metadata Vocabulary (OMV) [18] which describes the properties of ontology. It does not involve in the specific knowledge in ontologies when publishing and discovering them.

In addition, we [26] also propose a structured P2P-based approach to publish sharable ontologies in different sites and automatically locate the useful ontologies for a semantic query. Given a semantic query, if an ontology published can be reasoned out solutions for it, the approach is sure to locate the ontology and achieve related solutions.

However, in these approaches, knowledge sharing is incomplete because it is based on sharable ontology rather than knowledge itself. This is to say, the unit of sharing knowledge is ontology itself. For example as mentioned above, though we cannot construct any solutions for a query from ontology *A* or ontology *B* respectively, yet we can from knowledge together in ontology *A* and *B*.

As matter of fact, there are also some P2P-based approaches for ontology publication, which just focus on distributed storage and management of web ontology. The typical works are discussed as follows:

Min et al. [21] present a scalable distributed RDF repository (named RDFPeers) that stores each triple at three places by applying globally known hash functions to its subject, predicate, and object. Thus, all nodes know which node is responsible for triples they are looking for. Queries are guaranteed to find out matched triples in the network if the triples exist. However, if RDF triples are directly published on P2P, it does not support semantic retrieval. To address this question, Kohigashi et al. [22] focus on class hierarchies of RDF resources, encode them into related resource ID, and publish the ID. Then, they present a P2P information sharing method for RDF triples based on the class hierarchies. However, except for classification information, the method cannot yet support complex semantic query.

When semantic information is distributed over a structured P2P network, some related problems, such as load balance and reliability, will be encountered. To address the problems, Rizzo et al. [23] present a solution for distributed and reliable RDF storage. But, they do not care about semantic retrieval.

In general, these approaches directly publish knowledge in web ontology, so they break the limitations that the unit of sharing knowledge is ontology itself. But, except for classification information, complex semantic query processing is difficult to achieve based on these approaches.

In current research, many works focus on distributed management and sharing of knowledge in virtual community. These typical works are listed as follows:

The works [27-30] discuss interaction, behavior and key techniques for knowledge sharing in virtual community from different perspectives. Especially, the work [30] discloses characteristics of knowledge sharing in virtual community based on structured P2P techniques. But, they did not focus on specific implementation technology.

Chen et al. [8] propose an approach for knowledge sharing in community, which organizes the P2P nodes with sharable knowledge as an unstructured P2P network. If a node receives a query and has not related knowledge to process it, the P2P node sends the query to its neighbors until the query can be processed. Similar approaches are also presented in works [4, 7, 9, 31-33]. Their difference is that they adopt different strategy to construct unstructured P2P strategy. They try to connect the P2P nodes with similar knowledge as neighbors so as to improve the routing efficiency of semantic query. Just like the first category, knowledge sharing is incomplete in these approaches. In fact, unstructured P2P also limits their efficiency.

Javier et al. [34] propose a multi-agent system-based approach for knowledge sharing in community. However, due to its complexity, multi-agent system (MAS) has not been applied effectively. The current research is limited and further research work should be conducted.

In addition, in the works [35-38], distributed ontology techniques are discussed. But, they just focus on mapping, integration, and reason of distributed ontology. They do not place emphasis on knowledge publication and retrieving in ontology.

VII. Conclusion and Future Work

In this paper, we propose a structured P2P-based approach to publish and share RDF triples in web ontologies so as to automatically gather related RDF triples from different ontologies in different nodes to process SPARQL queries in an open distributed environment. It overcomes the limitation that knowledge sharing is based on entire ontology. It is particularly suitable for knowledge sharing in a virtual community, where numbers of ontologies of a given domain are created respectively by informal groups of people, and shared and leveraged for their respective purposes. We also conducted experiments to evaluate the effectiveness and efficiency of the approach. The experimental results demonstrated that it is effective and efficient.

In near future, we plan to continue our research work in the following aspects:

- Conduct experiments for our approach in a large-scale community.
- Conduct further study to investigate graph pattern of SPARQL query to find out other clues to more efficiently retrieve related RDF triples rather than the appearing entities. For example, structure of graph pattern, relationship among entities in graph pattern, and so on.
- Deal with the inconsistency among retrieved RDF triples. In a large-scale community, RDF triples related to a query maybe come from different ontologies in different nodes. Inconsistency is unavoidable.
- Study the method to map a word to existing ontological entities so as to facilitate requestors to construct their SPARQL queries automatically.

ACKNOWLEDGMENTS

The research work was supported by Scientific Starting Foundation of Hangzhou Dianzi University under Grant No. KYS055612041 and National Natural Science Foundation of China under Grant No. 61003077.

REFERENCES

[1] OWL. http://www.w3.org/TR/owl2-overview/. Retrieved March 17, 2013.

[2] Swoogle. http://swoogle.umbc.edu. Retrieved March 17, 2013.

[3] Ning Zhong, Jiming Liu, and Yiyu Yao. Advances in Web Intelligence [M]. Higher Education Press. Beijing. 2011. pp.239.

[4] Zhen, L.; Jiang, Z. & Song, H. Distributed recommender for peer-to-peer knowledge sharing Information Sciences, 2010, 180, 3546 – 3561

[5] P. Maret, M. Hammond, and J. Calmet. Virtual Knowledge Communities for Corporate knowledge Issues [C]. M.-P. Gleizes, A. Omicini, and F. Zambonelli (Eds.): ESAW 2004, LNAI 3451, pp. 33–44.

[6] Melanie Gnasa, Sascha Alda, Jasmin Grigull et al. Cremers. Towards Virtual Knowledge Communities in Peer-to-Peer Networks [C]. J. Callan et al. (Eds.): SIGIR 2003 Ws Distributed IR, LNCS 2924, pp. 143–155

[7] J.S.H. Kwok, S. Gao, Knowledge sharing community in P2P network: a study of motivational perspective, Journal of Knowledge Management 8(1) (2004) 94–102.

[8] Chen-Ya Wang, Hsin-Yi Yang, Seng-cho T. Chou. Using peer-to-peer technology for knowledge sharing in communities of practices, Decision Support Systems 45 (2008) 528–540.

[9] L.C. Jain, N.T. Nguyen, Knowledge Processing and Decision Making in Agent-based Systems, Springer-Verlag, 2009.

[10] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Transactions on Networking (TON) 11(1) (2003) 17-32.

[11] SPARQL Query Language for RDF. http://www.w3.org /TR/rdf-sparql-query/. Retrieved March 18, 2013.

[12] RDF. http://www.w3.org/RDF/. Retrieved March 17, 2013.

[13] D. Karger et al., Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. Proc. 29th Annual ACM Symp. Theory of Comp., May 1997, pp. 654–63.

[14] Jena 2. http://openjena.org/. Retrieved March 20, 2013.

[15] Pellet. http://clarkparsia.com/pellet. Retrieved March 20, 2013.

[16] Open Chord. http://open-chord.sourceforge.net/. Retrieved March 20, 2013.

[17] Raul Palma1, Peter Haase. Oyster-Sharing and Re-using Ontologies in a Peer-to-Peer Community. Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729. 2005. pp.1059–1062

[18] J. Hartmann, R. Palma, Y. Sure, M. Suarez-Figueroa, P. Haase, A. Gomez-Perez, and R. Studer. Ontology metadata vocabulary and applications. In Proc. of the Workshop on Web Semantics (SWWS'05), First IFIP WG 2.12 and WG 12.4 Agia Napa, Cyprus, 2005.

[19] Jing Tian, Yafei Dai, and Xiaoming Li. SemanticPeer: An Ontology-Based P2P Lookup Service. M. Li et al. (Eds.): GCC 2003, LNCS 3032, pp. 464–467, 2004.

[20] Gao, Q.; Qiu, Z.; Wu, Y.; Tian, J. & Dai, Y. An interest-based P2P RDF query architecture. Proceedings of the First International Conference on Semantics, Knowledge and Grid (SKG'05). 2005. pp.1-5

[21] Min Cai, Martin Frank. RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network. Proceedings of the 13th international conference on Word Wide Web (WWW'04). ACM Press. 2004. pp.650-657.

[22] Kohigashi, K.; Takahashi, K.; Harumoto, K. & Nishio, S. A Peer-to-peer information sharing method for RDF triples based on RDF schema. Lecture Notes in Computer Science, 2009, 5518 LNCS, 646-650

[23] Rizzo, G.; Di Gregorio, F.; Di Nunzio, P.; Servetti, A. & De Martin, J. C. A peer-to-peer architecture for distributed and reliable RDF storage. 1st International Conference on Networked Digital Technologies(NDT 2009). 2009. pp.94–99

[24] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A Survey and Comparison of Peer-To-Peer Overlay Network Schemes. IEEE Communications Surveys & Tutorials Second Quarter 2005, Volume 7, No.2. pp:72-93.

[25] Diego Calvanese, Giuseppe de Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati.Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family [J]. Journal of Automated Reasoning 39(3):385–429, 2007

[26] Huayou Si, Zhong Chen, Yong Deng. P2P-based Publication and Location of Web Ontology for Knowledge Sharing in Virtual Communities[C]. The 24th International Conference on Software Engineering and Knowledge Engineering (SEKE2012). Redwood City, San Francisco Bay, USA. 2012.07

[27] Kleanthous, S., Dimitrova, V.: Analyzing community knowledge sharing behavior. In: Proceedings of User Modeling, Adaptation, and Personalization UMAP2010. Springer, Hawaii (2010) , pp. 231–242

[28] Kleanthous, S., Dimitrova, V.: Detecting changes over time in a knowledge sharing community. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence I2009, Milan. IEEE Computer Society, Los Alamitos (2009) , pp.100–107

[29] Puntambekar, S.: Analyzing collaborative interactions: divergence, shared understanding and construction of knowledge. Comput. Educ. 47(3), 332–351 (2006)

[30] Cheng, R., Vassileva, J.: User motivation and persuasion strategy for peer-to-peer communities. In: Proceedings of 38th Hawaii International Conference on System Sciences Hawaii, USA, pp. 3–6 (2005)

[31] Ardissono, L., Bosio, G.: Context-dependent awareness support in open collaboration environments. User Model. User-Adapt. Interact. 22(3), 223–254 (2012)

[32] Kleanthous Loizou, S.: Intelligent support for knowledge sharing in virtual communities. Ph.D., School of Computing, University of Leeds, Leeds (2010). pp.45-46

[33] Zhang, J., Ackerman, M., Adamic, L.: Expertise networks in online communities: structure and algorithms. In: Proceedings of the International Conference on World Wide Web WWW2007, Alberta, Canada, ACM Press, New York (2007) . pp. 221–230.

[34] Javier Portillo-Rodríguez, Aurora Vizcaíno, Juan Pablo Soto et al. Fostering Knowledge Exchange in Virtual Communities by Using Agents [C]. J.M. Haake, S.F. Ochoa, A. Cechich (Eds.): CRIWG 2007, LNCS 4715, pp. 32–39, 2007.

[35] Schlicht, A. & Stuckenschmidt, H. Towards Distributed Ontology Reasoning for the Web [C]. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology(WI-IAT '08). 2008.1. pp.536 -539

[36] Lee, J.; Park, J.; Park, M. et al. An intelligent query processing for distributed ontologies [J]. Journal of Systems and Software. 2010(83). pp.85 – 95

[37] Kaneiwa, K. & Mizoguchi, R. Distributed reasoning with ontologies and rules in order-sorted logic programming [C]. Web Semantics: Science, Services and Agents on the World Wide Web, 2009, 7, 252 – 270

[38] Jiang, Y.; Tang, Y.; Wang, J. et al. Representation and reasoning of context-dependant knowledge in distributed fuzzy ontologies [J]. Expert Systems with Applications, 2010, 37, 6052 – 6060

[39] Huayou Si, Zhong Chen, Yun Zhao, Yong Deng. P2P-Based Publication and Sharing of Axioms in OWL Ontologies for SPARQL Query Processing in Distributed Environment[C]. The 14th Asia-Pacific Web Conference (APweb 2012). April 11-13, 2012