Peer-to-Peer Discovery of Semantic Associations

Matthew Perry, Maciej Janik, Cartic Ramakrishnan, Conrad Ibañez, Budak Arpinar, Amit Sheth

LSDIS Lab,
Department of Computer Science, The University of Georgia
415 Boyd Graduate Studies Research Center
Athens, GA 30602-7404
{mperry, janik, cartic, ibanez, budak, amit}@cs.uga.edu

Abstract. The Semantic Web vision promises an extension of the current Web in which all data is annotated with machine understandable metadata. The relationship-centric nature of this data has led to the definition of Semantic Associations, which are complex relationships between resources. Semantic Associations attempt to answer queries of the form "how are resource A and resource B related?" Knowing how two entities are related is a crucial question in knowledge discovery applications. Much the same way humans collaborate and interact to form new knowledge, discovery of Semantic Associations across repositories on a peer-to-peer network can allow peers to share their local knowledge to collectively make new discoveries. In this paper we propose a method for computing Semantic Associations over distributed RDF data stores in a peer-to-peer setting. We follow a hierarchical peer / super-peer network topology, and we propose a novel query planning algorithm based on a notion of knowledgebase borders and minimum distances between borders.

1 Introduction

Today's data and information management tools enable massive accumulation and storage of knowledge that is produced through scientific advancements, personal and corporate experiences, communications, interactions, and other accomplishments. The willingness and the ability to share and use this information are key factors for realizing the full potential of this knowledge scattered over many distributed data stores. By correlating these isolated islands of knowledge, individuals can gain new insights through the discovery of new relations. For example, many complex scientific problems increasingly require collaboration between teams of scientists who belong to diverse disciplines [11]. Such collaboration therefore requires a system that enables scientists to correlate knowledge across multiple knowledgebases. In this paper we present a method of discovering complex relationships between resources across distributed RDF [12] repositories to enable such collaboration in a peer-to-peer environment.

Determining how two resources are related provides insight and knowledge about the two resources. The relationship-centric organization of RDF data allows us to find answers for these questions. In [2], Anyanwu and Sheth propose a set of relationship-

based query operators for this purpose. They define a Semantic Association as a complex relationship between two resources, and introduce a set of operators, ρ, for querying Semantic Associations. One such operator is *ρ-path*. Two resources A and B are ρ-path related if a path exists from A to B in the RDF graph. Finding ρ-path associations that span many peers will allow for collective discovery of new knowledge. With the goal of efficient processing of Semantic Associations in a peer-to-peer environment, this paper makes the following contributions:

- A P2P architecture for the efficient computation of Semantic Associations
- Introduction of the concept of knowledgebase borders (knowledge contained in more than one knowledgebase).
- A query planning algorithm for *ρ-path* queries based on borders and distances between borders.

2 Semantic Associations

RDF that initially emerged as the de-facto standard for representing semantic metadata has been adopted by W3C as a standard for representing information on the Web. RDF uses XML to represent metadata. RDFS (RDF Schema) [13] provides a standard vocabulary that can be used to specify concepts and relationships between them. Relationships in RDF, known as *Properties*, are binary relationships between resources/literals that take on the roles of *Subject* and *Object* respectively. The *Subject*, *Predicate* and *Object* compose an RDF statement. This model can be represented as a *directed* labeled graph with typed edges and nodes. In this model, a directed edge labeled with the *Property* name connects the *Subject* to the *Object*.

Semantic Associations are originally defined in [2]. We give simplified definitions of relevant Semantic Associations here for completeness, namely *ρ-path*. A path $P = e_1, p_1, e_2, p_2, e_3, \ldots, e_{n-1}, p_{n-1}, e_n$ is defined as a sequence of RDF statements where each ei, pi, ei+1 represents a single statement with pi the RDF predicate and one of e_i or e_{i+1} is the RDF subject and the other is the RDF object. Two resources *x* and *y* are *ρ-path* associated if there exists a path *p* of length $n > 0$ between them. A *ρ-path* query between two resources should return all such paths.

2.1 The *ρ-path* Problem

Here we define *ρ-path* association queries in graph theoretic terms and state the *ρ-path problem*. The RDF data model consists of four sets of resources: *C* the set of classes, *P* the set of properties, *I* the set of class instances, and *L* the set of literals. We define the set *P'* to be *P* – {*rdf:type, rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range, rdfs:label, rdfs:comment*}. We define an RDF instance graph as a directed, edge labeled graph *G(V,E)* where $V = \left\{ c \in I \middle| \exists p \in P' \wedge \exists \langle subj, pred, obj \rangle \left| (sub = c \vee obj = c) \wedge pred = p \wedge obj \notin L \right. \right\}$ each $c \in V$ will get the label *uri(c)*. For each triple $\langle s, p, o \rangle$ with $p \in P'$ we add an edge *(s, o)*,

with label *uri(p)*, to *E*. In other words, the RDF instance graph includes class instances and relationships between them, but does not include ontology schema information.

We define the *ρ-path* problem as follows:

ρ-path Problem	
Given:	RDF instance graph *G*, vertices *x* and *y* of *G*, and an integer *k*
Find:	All simple, undirected paths *p* with length $\leq k$ in *G* which connect *x* and *y*

We chose the *k-hop* limit and simple paths constraint for practical and computational reasons. Given the many possible ways we can piece together paths from different knowledgebases and the nature of the small-world phenomenon [9], without these constraints a *ρ-path* query on a well-connected graph could easily return most of the RDF dataset. We add the simple path constraint because we do not feel that loops at intermediate nodes in a path add much value to the association, and including these paths will just add to the information overload problem. In addition, we feel that very long *ρ-path* associations (i.e. greater than length 10) will not be easy for humans to understand and thus are less valuable.

2.2 Difficulties of a Distributed Environment

Finding a solution to the *ρ-path problem* in a distributed environment brings a number of challenges. The first one is the nature of a distributed environment. It requires an efficient communication protocol to forward a search between knowledge bases and later to gather results. Design decisions will include what data is sufficient to perform a search and create the final result from the found parts. Furthermore, decisions must be made about when and with what set of parameters to start searches in adjacent knowledgebases.

The distribution of knowledge between nodes causes paths to span over multiple knowledgebases, but a given path can cross the same knowledgebase many times. In a centralized environment, the whole path is included in a single graph. On the contrary, in a distributed environment only parts of the final path are fully included in one knowledgebase. Therefore, it is possible that the final path will include two or more 'sub-paths' from the same knowledgebase. This feature makes merging of final paths from locally computed parts a hard task.

An additional challenge is dividing and optimizing computation to find parts of final paths. For a given maximum length restriction of a path search, it may not be trivial to decide the search length in each participating knowledgebase. We do not want to perform too many calculations that end up being useless in constructing the final result. However, since the number of paths between any two entities grows exponentially with path length, we can gain performance by dividing a large data store over a number of peers. It is likely that each peer will perform searches of length significantly less than the overall hop limit.

3 Related Work

There have been other proposals for querying and searching RDF metadata on P2P networks. In [8] the authors propose a Peer-to-Peer framework for searching for ontologies. In this paper the authors provide the motivation for using the P2P framework as a means of realizing the Semantic Web vision. In [10] the authors present a super-peer-based routing and clustering strategy for peers that maintains RDF metadata. The main idea suggested by them is to use the RDF schema as a means of organizing RDF data. They however enforce a definite topology on the network of super-peers. This structure is a *d-dimensional* hypercube based on the structure suggested in [14]. The choice of such a structure is motivated by the bounded diameter of the networks realized using such structure. Further, since the structure is built using information from the RDF schema of the data stored at the peers, the hypercube structure is both structural and semantic. Other approaches have been suggested in [1] and [3]. There has also been work on a structured P2P RDF repository based on distributed hashing [4].

Since the types of queries that we seek to answer are fundamentally different from those described in any of the above approaches, we believe that enforcing a strict topology on the super-peers in the system is undesirable. Two entities could be related in a myriad of different ways. An ad hoc topology of super-peers with sufficient redundancy of data is desirable. We therefore propose a scheme where peers with RDF data joining the network of super-peers advertise their content to all super-peers. This advertisement is then used to compute the extent of overlap between the peers' RDF data and that indexed by each super-peer is computed. The peer then joins the super-peer with minimum overlap, thereby ensuring greater diversity of data indexed by each super-peer. A desirable side-effect of this is that the communication overhead between super-peers is likely to be much lower for a given query, since the probability of finding both end-points of the query indexed by the same super-peer is higher.

4 Our Approach

In this section we present system architecture and details of forming query plans. Query plans are based on knowledgebase borders and distances between them.

4.1 Peer-to-Peer Architecture

We base our solution to the distributed *ρ-path problem* around a hierarchical P2P network topology, see [16] for an overview. In our approach, the network consists of a set *P* of peers and a set *S* of super-peers. System architecture is presented on **Fig. 1**. Knowledgebases (RDF datastores) are stored at the peer level, while indexes are stored at the super-peer level. Each peer contains one or more knowledgebases, and each peer is associated with one super-peer group. Each super-peer keeps an inverted

index mapping resources to peers and/or super-peers. A super-peer does not store any information about resources contained completely outside of its group. A super-peer knows about all other super-peers in the network and can query them to determine which group contains a given resource.

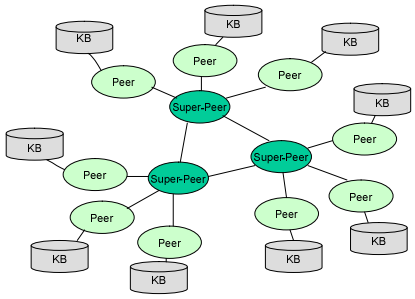

Fig. 1. P2P Architecture

In this scheme super-peers are responsible for query planning, while peers are responsible for query execution. Upon receiving a query, the peer forwards it to its super-peer and asks for a query plan. Acting as coordinator for the query, this super-peer will communicate with other super-peers to form the plan. Query planning consists of formulating subquery planning on two levels. The first task is to plan the query on the super-peer level. The second step requires each super-peer to form a more detailed plan within its group. After receiving the query plan, the peer that initiated the query will directly communicate with the other peers to execute each query.

4.2 Knowledgebase Borders

The solution for correct identification and disambiguation of resources that belong to multiple knowledgebases is a necessary step to allow a search for semantic associations in distributed knowledgebases. If we want to find semantic associations that span over different knowledgebases, *apriori* we need to find what knowledge they share. Shared resources are necessary for a path to go from one knowledgebase to another, but the problem of finding them is not trivial. Consider an example where an organization has a collection of metadata repositories where each repository deals with a logically separate domain. Some real-world entities will have corresponding metadata in more than one repository. For example, Tiger Woods will most likely have annotations from both the sports and business domains since he is both a golfer and a spokesperson for an automotive company. Such a multi-classified entity provides a means of linking the two repositories.

Different knowledgebases may have different schemas and different naming conventions. It may often happen that the same resource in real life will have one identifier in the first knowledge base and another in the second. It is a hard problem to correctly merge or create an equivalence mapping between differently identified

resources that, in fact, describe the same entity. Many attempts have been made at various forms of this problem, for example [5], [6]. In the process of finding commonly shared knowledge we identify resources that belong to both knowledgebases and create an equivalence mapping between them. Each of these resources is called a **border node**. This is the joining point of two or more knowledgebases. When a semantic association path reaches such a node, it can be continued in another knowledgebase. We call the set of all such resources that belong to the intersection of different knowledgebases a **border**. Each border node can belong to only one border, and all borders are pair wise disjoint. This requirement is stated for correct guidance of the search algorithm. Each border is identified by the set of knowledgebases it connects. This determines which knowledgebases are possibly 'reachable' from a given border.

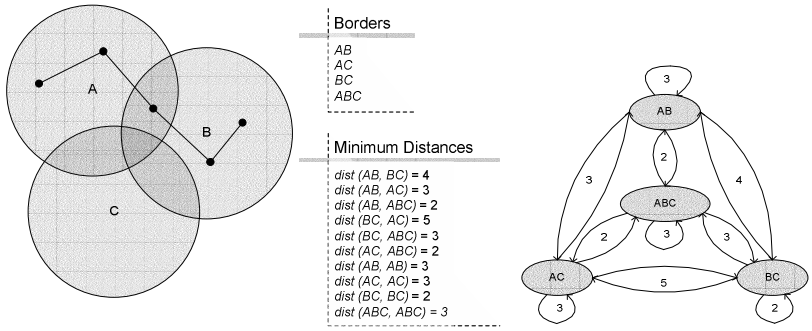

Fig. 2. Knowledgebase borders and Query Plan Graph

To illustrate how borders are defined, consider **Fig. 2**. In this example for knowledge bases *A*, *B*, *C* we have four borders: *AB*, *AC*, *BC* and *ABC*. Nodes that belong to border *ABC* do not belong to border *AB*, *AC* or *AB*. When a query search algorithm reaches nodes in this border *ABC*, it can continue in any knowledgebase belonging to this border. It also means that it can continue an association path search in borders *AB*, *AC* or *BC* as all of them belong to knowledgebases that are reachable from border *ABC*.

4.3 Notion of Minimum Distance between Borders

We use the notion of a minimum distance between knowledgebase borders in combination with the *k-hop* limit to form a hop-limited *ρ-path* subquery. We define the **minimum distance**, *dist (B1, B2)*, between two borders *B1* and *B2* to be the shortest path that connects some node in B1 to some node in B2. The query-specific hop limit for the subquery is used to prune fragments of paths that exceed the *k-hop* limit, thus reducing wasted computations and network traffic. For example, consider the set of knowledgebases in **Fig. 2**. Assume that the start node is contained in knowledge-base *A* and is a distance of 3 hops, 4 hops, and 3 hops from the *AB*, *ABC*,

and *AC* borders respectively. Also assume that the end node is located in knowledgebase *B* at a distance of 2 hops, 4 hops, and 3 hops from the *AB*, *ABC*, and *BC* borders respectively. With this knowledge of borders and minimum distances, if we have a *k-hop* limit of less than 9, we can conclude that there is no point in looking for paths through knowledgebase *C*. This results from the fact that *dist (AC, ABC)*, *dist (BC, ABC)*, and *dist (AC, BC)* makes it impossible to find a path from start to end which passes through any two of these borders. Further, we can compute the upper bounds on the hop-limits of subqueries through knowledgebases *A* and *B* given the border and mini-mum distance information.

4.4 Computing Border Information

Super-peers are responsible for computing the borders in their own group, and super-peers collectively discover borders between groups. The computation of borders assumes that the entity disambiguation problem has been solved, and a single real-world entity is identified by the same URI across knowledgebases. Upon joining a group, a peer sends the corresponding super-peer the set of URIs it contains. Each super-peer maintains a sorted list of all the URIs in its group with proper references to peers. Super-peers exchange messages about their URIs to find overlap, where each message contains one URI. To minimize message communication, an algorithm for super-peer border discovery is based on finding the intersection of two sorted lists, which is $O(n)$.

4.5 Query Plan Graph

To store the border node and minimum distance information we use a graph representation termed a **Query Plan Graph** (QPG). In this weighted, directed graph, the set of nodes represents the set of borders between datasets, and a pair of directed edges (one in each direction) connects every border which shares a dataset (i.e. *AB* and *BC*). In addition, a self-loop edge is added for each border. The weight on each pair of edges is the minimum distance between the two borders. In the case of a loop, the distance is the shortest distance between any two distinct nodes in the border. Intuitively, edges in this graph represent *ρ-path* queries between the set of nodes in each border. The right side of **Fig. 2** shows the QPG resulting from the sub-network on the left side of **Fig. 2**. For purposes of query planning (section 4.6), we would like to know the border topology both at the super-peer level and at the peer level (within a group).

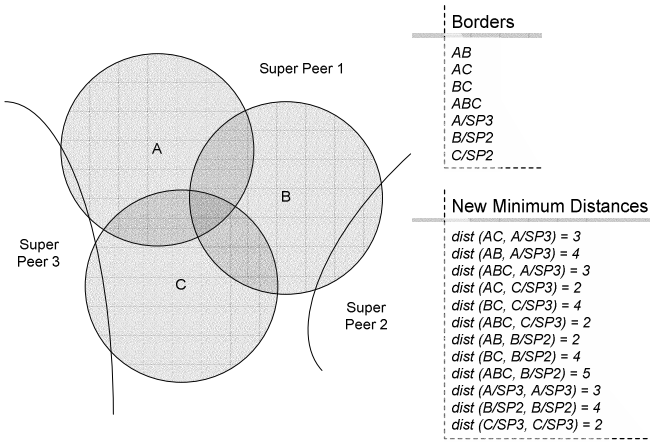

Fig. 3. Knowledgebases showing super-peer overlap

We form a super-peer level QPG known to all super-peers, and we form a peer level QPG known only to the group's super-peer. In addition, we need to add super-peer border information to the peer level QPG so that we can form a mapping from the peer level to the super-peer level. In order to integrate the peer level QPG with the super-peer level graph, we think of the peer level graph as the expansion of an edge in the super-peer level graph. In other words, the super-peer level graph acts as an overlay network for the peer level graph. This recursive relationship has the added benefit of allowing for the possible definition of multiple levels of super-peer overlays. **Fig. 3** and **Fig. 4** illustrate how a super-peer augments its QPG for a peer group to integrate super-peer border topology. Overlaps between other super-peers and peers in the sub-network are computed. In this case, knowledgebases *A* and *C* overlap with super-peer 3 and knowledgebase *C* overlaps with super-peer 2. Nodes are created for these peer/super-peer borders, and one node is created for the corresponding super-peer/super-peer border (gray nodes in **Fig. 4**) and connected by zero-weight edges to each of the peer/super-peer borders. The integrated QPG allows super-peers to find minimum distances between super-peer borders by finding shortest paths between super-peer/super-peer nodes in the peer level QPG. These super-peer/super-peer nodes in the peer level graph map to the nodes in the super-peer level graph, and edge weight is computed by finding the shortest path between the nodes in the peer level graph.

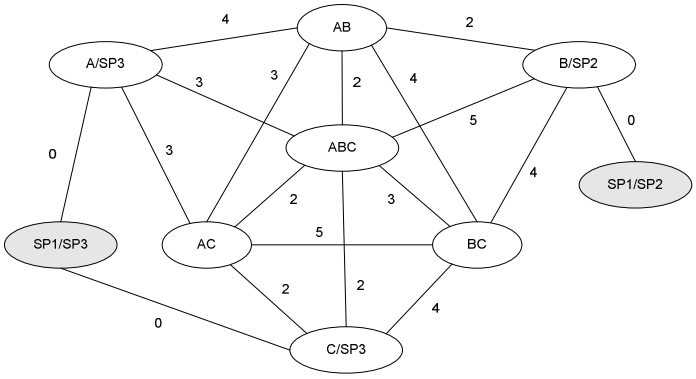

Fig. 4. Query Plan Graph showing super-peer border information. Note that each single edge represents two directed edges and self-loops have been left out for clarity

4.6 From Query Plan Graph to Queries

The process of forming a query plan for a given *ρ-query* starts with the super-peer level graph and moves to the peer level graph. It is based on finding the position of the start and end node in the border topology, then finding paths from the start node to the end node in the super-peer QPG, and then replacing edges in these super-peer level paths with paths in the peer level QPG. After paths are found, we determine upper-bounds on hop limits for queries and compile a list of queries for each knowledgebase.

The first step in the query planning process is to augment the super-peer graph with the location of the query endpoints. Upon receiving the *ρ-query* from one of its peers, the super-peer locates the two endpoints for the query. After locating the end-points, temporary nodes and edges are added to the super-peer QPG to reflect the position of the endpoints in the border topology. If an endpoint is contained within its sub-network, the super-peer will locate it with its inverted index. If an endpoint cannot be located in the index, the super peer will broadcast a message to other super-peers to determine its location in the topology. There are two cases for the location of an end-point: either within a border or outside a border. In the latter case, the containing super-peer will respond to the message with a list of edges that specify the endpoint's distance to each border for that super peer. A temporary node is created for the endpoint and the temporary weighted edges are added to the query plan graph. In the former case, the super-peer only notes that the node for the border should be treated as an endpoint.

Once we have found the locations of the endpoints in the border topology, we can search for various ways to piece together paths through different knowledgebases. First, we find all paths (including cycles), with weight less than the original *k-hop* limit, from the start node to the end node in the super-peer QPG.

Knowledgebase	Query
A	*ρ-path (AB, BC, 3)*
A	*ρ-path (AB, ABC, 4)*
A	*ρ-path (AB, A/SP3, 4)*
A	*ρ-path (ABC, A/SP3, 5)*
B	*ρ-path (AB, ABC, 4)*
C	*ρ-path (AC, C/SP3, 2)*
C	*ρ-path (BC, C/SP3, 4)*
C	*ρ-path (ABC, C/SP3, 6)*

Table 1. Resulting subqueries for *ρ-path (SP1/SP2, SP1/SP3, 8)*

Each edge e with weight w in a path p through the QPG now needs to be converted into one or more *k-hop* limited *ρ-path* queries through the lower-level network, in this case the peer level QPG. The first step in the transformation is to determine the upper bound for the hop limit. Assume the original *k-hop* limit is *n* and the path *p* which contains e has a total weight of *W*. The upper bound for the sub-query is $w + (n - W)$. The final weight for the query is the global maximum over all such paths. The second step is to determine to which super-peers we should send the query. A border is identified by the list of containers (super-peers or peers) which form the border. The containers that receive the query are exactly those that appear in the intersection of the container lists of the two borders. For example, an edge connecting the borders *SP2/SP3/SP4* and *SP2/SP3* should go to both *SP2* and *SP3*. After receiving a sub-query, the super-peer will run the aforementioned algorithm for forming query plans recursively on its peer level QPG (with end points equal to the adjacent nodes in the super-peer level edge) and return a peer level query plan. **Table 1** shows the sub-query plan resulting from a *ρ-path (SP1/SP2, SP1/SP3, 8)* query plan request.

4.7 Query Execution

A peer executes a query plan by communicating directly with the corresponding peers. This eliminates a potential super-peer bottleneck because query execution will take much longer than query planning. The query plan groups queries based on the peer that should receive them. The querying peer will then send each peer its entire set of queries at one time. Upon receiving this set of queries, the peer executes each one and merges the results into a single RDF subgraph, which is then transmitted to the calling peer as the result. A trie-based bidirectional breadth-first-search algorithm is used for the path search. Returning an RDF subgraph as opposed to a set of paths both eliminates the exponential-time problem of path enumeration and minimizes network traffic by eliminating node and edge redundancies in the result set. The querying peer collects all such results and merges them to obtain the RDF subgraph that contains the answer to the *ρ-path* query. At this point, path enumeration can be done to form the final answer.

5 Conclusions and Future Evaluations

This paper presented an approach to efficiently execute Semantic Association queries across a peer-to-peer network. The problem of executing path searches across distributed RDF data sets brings many unique challenges. The network organization used in existing approaches to searching RDF data over P2P networks makes path searching difficult. This is due to network communication inefficiencies when piecing together paths and difficulty determining when to stop a path search in one peer and continue it in another. Our approach solves the first problem by keeping logically related, and thus very connected, components of the RDF graph at the same peer. Moreover, the use of border nodes and border distances solves the second problem of when to stop a search in one peer and continue it in another.

Currently, we have a prototype implementation on top of a simulated network. We plan to complete a full implementation with JXTA™ [7]. For performance evaluations, we want to test its efficiency in terms of computation time and network traffic. We will use a synthetic RDF graph generator developed in the LSDIS lab to generate large, well-connected datasets. In addition we can test on real-world data by using the Semantic Web Technology Evaluation Ontology or TAP [15]. We would like to compare network traffic data and running time for a path search on our system versus other systems, for example [4]. In addition, we would like to analyze the amount of data that is transferred over the network but not used in the final answer, and it would be interesting to see what percentage of the dataset is transferred on average for a query. We also want to see how the relative size of border node sets in comparison with the over-all size of knowledgebases impacts performance. Finally, we want to look at performance gains from parallelism over a distributed dataset versus a path search in a single large dataset.

Acknowledgements

We thank all SemDIS project members for their insightful comments and revision suggestions. This project is funded by NSF-ITR-IDM Award#0325464 (SemDIS: Discovering Complex Relationships in the Semantic Web) and NSF-ITR-IDM Award#0219649 (Semantic Association Identification and Knowledge Discovery for National Security Applications).

References

1. Aberer, K. and Hauswirth, M., Semantic gossiping. in *Database and Information Systems Research for Semantic Web and Enterprises, Invitational Workshop*, (University of Georgia, Amicalola Falls and State Park, Georgia, 2002).
2. Anyanwu, K. and Sheth, A., r-Queries: Enabling Querying for Semantic Associations on the Semantic Web. in *The Twelfth International World Wide Web Conference*, (Budapest, Hungary, 2003).

3. Bernstein, P., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L. and Zaihrayeu, l., Data Management for Peer-to-Peer Computing: A Vision. in *Fifth International Workshop on the Web and Databases*, (Madison, Wisconsin, 2002).
4. Cai, M. and Frank, M., RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network. in *The Thirteenth International World Wide Web Conference*, (New York, New York, 2004).
5. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R.V., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J.A. and Zien, J.Y., SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. in *The Twelfth International World Wide Web Conference*, (Budapest, Hungary, 2003), 178-186.
6. Dong, X., Halevy, A. and Madhavan, J., Reference Reconciliation in Complex Information Spaces. in *ACM SIGMOD/PODS Conference*, (Baltimore, Maryland, 2005).
7. JXTA. http://www.jxta.org/.
8. Maedche, A., Motik, B., Stojanovic, L., Studer, R. and Volz, R., An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies. in *The Twelfth International World Wide Web Conference*, (Budapest, Hungary, 2003).
9. Milgram, S. The Small World Problem. *Psychology Today*, *May 1967*. 60-67.
10. Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I. and Löser, A., Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-To-Peer Networks. in *The Twelfth International World Wide Web Conference*, (Budapest, Hungary, 2002).
11. Pike, W., Ahlqvist, O., Gahegan, M. and Oswal, S., Supporting Collaborative Science through a Knowledge and Data Management Portal. in *ISWC 2003 Workshop Semantic Web Technologies for Searching and Retrieving Scientific Data*, (Sanibel Island, Florida, 2003).
12. RDF. http://www.w3.org/RDF/.
13. RDFS. http://www.w3.org/TR/rdf-schema/.
14. Schlosser, M., Sintek, M., Decker, S. and Nejdl, W., HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks. in *International Workshop on Agents and Peer-to-Peer Computing*, (Bologna, Italy, 2002).
15. TAP. http://tap.stanford.edu/.
16. Yang, B. and Garcia-Molina, H., Designing a Super-Peer Network. in *19th International Conference on Data Engineering*, (Bangalore, India, 2003).