Improving Distributed Semantic Search with Hybrid Topology and Peer Recommendation

View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by CiteSeerX

Abstract. In this paper, we propose a novel framework for discovery Semantic Web data in large-scale distributed networks. In this framework, peers dynamically perform topology adaptations to spontaneously create communities with similar semantic interests, so that search requests have a high probability of being satisfied within the local community. For queries which cannot be efficiently solved inside the community, a directory overlay built on Distributed Hash Table (DHT) is used to assist the search. Recommendation from peers of the same community is employed to extract only semantically related results thus improving the precision. Experiments with simulations substantiate that our techniques significantly improve the search efficiency, scalability, and precision.

1 Introduction

Semantic web has been presented as an evolving extension of World Wide Web [1, 2, 3]. With the development of semantic web technologies, more and more semantic web data are generated and widely used in Web applications and enterprise information systems. These data are structured with ontologies [4] for the purpose of comprehensive and transportable machine understanding. To fully utilize the large amount of semantic data, an effective search mechanism customized for semantic web data, especially for ontologies, is needed by human users as well as software agents and services. The unique semantic features and the inherent distributed nature of semantic web data make its discovery highly challenging.

Peer-to-peer (P2P) technology has been used as a solution to distributed resource discovery, since it scales to very large networks, while ensuring high autonomy and fault-tolerance. The recently proposed structured P2P systems in the form of DHTs [5-8] are a promising approach for building massively distributed data management platforms. However, they offer few data management facilities, limited to IR (Information Retrieval) -style keyword search. Keyword search is appropriate for simple file-sharing applications, but is unable to deal with complex semantic queries which have various properties and sophisticated relations with each other.

More recently, a few studies [9, 10] extended the DHT-based P2P to support semantic queries. The basic idea is to map each keyword of a semantic entity to a

Juan Li
Department of Computer Science, North Dakota State University
e-mail: j.li@ndsu.edu

R. Lee, G. Hu, H. Miao (Eds.): Computer and Information Science 2009, SCI 208, pp. 83–92.
springerlink.com © Springe-Verlag Berlin Heidelberg 2009

key. For example, RDFPeer [9] indexes each RDF [20, 21] triple to support semantic RDF query. A query with multiple keywords then uses the DHT to lookup each keyword and returns the intersection. Systems like [8] avoid this multiple lookup and intersection by storing a complete keyword list of an object on each node. In this way, the DHTs can support multi-keywords queries. However, DHTs still have difficulty to support other richer queries, such as wildcard queries, fuzzy queries, and proximity queries. In addition, most DHT-based applications require all peers in the system sharing a uniform ontology schema, which is impractical in reality. These limitations restrict the deployment of DHTs to semantic web data discovery.

To support flexible complex queries, many P2P systems [11, 12] use flooding or maintain a broadcast structure, such as a tree or a super cube, to propagate the queries to the network. For example, to execute an RDF query, Edutella [11] broadcasts the query to the whole hypercube. However, the overhead of flooding and broadcast may cause scalability issues.

To overcome the shortcomings of existing discovery approaches, we propose a hybrid search mechanism, which integrates structured DHT P2P technology with unstructured P2P technology. Recommendation feedback from semantically similar peers is employed to retrieve the most relevant results thus improving the efficiency and precision of searching. In our system, each node is associated with a semantic summary representing the node's interest. Based on the summary, we design a method to compute the semantic similarity between different nodes. The network topology is reconfigured with respect to nodes' semantic similarity, so that peers with similar semantics are close to each other, forming a semantic community. The semantic community is loosely structured as an unstructured P2P overlay, called community overlay. Because of its unstructured topology, the community overlay is able to handle flexible complex queries. The semantic locality property guarantees that the system's query evaluation can be limited to relevant peers only. A structured DHT-based overlay is used to facilitate the construction of the community overlay and to assist evaluating queries which cannot be effectively resolved by the community overlay.

Members in the same community share similar interests hence are able to make recommendations to each other. Recommendations allow users to disambiguate search requests quickly. Moreover, they can personalize query results for users by ranking higher the results that are relevant to users' semantic properties. Therefore, the search quality in terms of both precision and recall is improved. In addition, peers recommend neighbors for each other according to their query experience to adapt to the evolving network property.

With the assistance of peer recommendation, community overlay and directory overlay complement each other, providing efficient search for the system. Compared to search in pure structured P2P systems, our hybrid search system has inherent support for complex semantic query or partial match; in addition, the retrieved results are more relevant. Compared to search in pure unstructured P2P systems, our community-based structure saves the overhead of flooding the query to unrelated nodes, thus enjoying more scalability.

2 System Overview

This section gives an overview of the system architecture. The proposed system consists of two logical overlays – an unstructured community overlay and a structured directory overlay –taking different roles for efficient operations of the system.

Query evaluation is mainly performed in the community overlay. In a community overlay, peers are connected to those sharing similar semantic interests. As a result, the query propagation tends to first reach those that are more likely to possess the data being searched for. This semantic locality property enables the community overlay to answer most queries originated from the local community. Unlike DHTs, community overlay does not specify any requirements for the query format, hence is able to handle any arbitrary types of complex queries. For the above reasons, a large portion of complex queries can be resolved inside the local community. However, it is still possible that a small portion of queries cannot be answered within the community overlays a peer belongs to, even if the peer may belong to multiple communities. Peers may have more interests which cannot be covered by the community overlays they reside. In this case, the index maintained by the directory overlay can be consulted for hints about where to forward the query for a second try.

The directory overlay is built on top of DHT protocols. It provides a high-level directory service for the system by indexing abstract ontology skeletons. The directory overlay has two main functionalities: (1) It facilitates the construction of community overlay. (2) It resolves queries not covered by the community overlay. Unlike community overlay, directory overlay does not give exact answers of a particular query; instead, it locates all peers possessing semantic keywords of the query. Then the query will be broadcasted to all peers related to the keywords for further evaluation. However, a keyword may have multiple meanings, not all of these meanings match the requestor's intention. Simply forwarding the query to all peers containing the keywords is not accurate and consumes lots of unnecessary network bandwidth.

The directory overlay employs peers' recommendation and feedback to solve the aforementioned semantic ambiguity problem. After receiving results from the directory overlay, the requestor first checks the validity of the results. Then it reports its findings back to the directory overlay nodes. The feedback will benefit future requesters with similar interests.

A physical node may be involved in both of these two overlays. Community overlay and directory overlay benefit from each other: directory overlay facilitate the construction of community overlay, while feedback from communities improves the search precision of directory overlay. Working together, these two overlays improve the search efficiency and accuracy of the system.

3 Community Overlay

The construction of the community is a topology adaptation process, i.e., to make the system's dynamic topology match the semantic clustering of peers. The

community topology enables queries to be quickly propagated among relevant peers. In addition, this topology allows semantically related nodes to establish ontology mappings.

3.1 Semantic Similarity

To find semantically similar neighbours, peers should be able to measure semantic similarity between each other. There has been extensive research [17 - 19] focusing on measuring the semantic similarity between two objects in the field of information retrieval and information integration. However, their methods are very comprehensive and computationally intensive. In this paper, we propose a lightweight method to compute the semantic similarity between two nodes.

Our system supports semantic web data represented as OWL ontology. OWL ontology can be divided into two parts: the terminological box (TBox) and the assertion box (ABox) as defined in the description logic terminology [16]. TBox ontology defines the high-level concepts and their relationships. It is a good abstraction of the ontology's semantics and structure. Therefore, we use a node's TBox ontology to represent its semantic interest. In particular, we use keywords of a node's TBox ontology as its ontology summary. However, a semantic meaning may be represented by different keywords in different ontologies, while it is also possible that the same keyword in different ontologies means totally different things. Ontology comparison based on TBox keywords may not yield satisfying results. In order to solve this problem, we extend each concept with its semantic meanings in WordNet [22]. We use two most important relationships in WordNet – synonyms and hypernym – to expand concepts. In this way, semantically related concepts would have overlaps.

After extension, a node's ontology summary set may get a number of unrelated words, because each concept may have many senses (meanings), but not all of them are related to the ontology context. A problem causing the ambiguity of concepts is that the extension does not make use of any relations in the ontology, which are important clues to infer the semantic meanings of concepts. To further refine the semantic meaning of a particular concept, we utilize relations between the concepts in an ontology to remove unrelated senses from the summary set. Since the dominant semantic relation in an ontology is the *subsumption* relation, we use the *subsumption* relation and the sense disambiguation information provided by WordNet to refine the summary. It is based on a principle that a concept's semantic meaning should be consistent with its super-class's meaning. We use this principle to remove those inconsistent meanings. For every concept in an ontology, we check each of its senses; if a sense's hypernym has overlap with this concept's parent's senses, then we keep this sense and the overlapped parent's sense to the ontology summary set. Otherwise, they are removed from the set. In this way we can refine the summary and reduce imprecision.

To compare two ontologies, we define an ontology similarity function based on the refined ontology summary. The definition is based on Tversky's "Ratio Model" [23] which is evaluated by set operations and is in agreement with an information-theoretic definition of similarity [24]. Assume A and B are two nodes,

and their ontology summary are S(A) and S(B) respectively. The semantic similarity between node A and node B is defined as:

$$sim(A,B) = \frac{|S(A) \cap S(B)|}{|S(A) \cap S(B)| + \alpha |S(A) - S(B)| + \beta |S(B) - S(A)|}$$

In the above equations, "∩" denotes set intersection, "–" is set difference, while "| |" represents set cardinality, "α" and "β" are parameters that provide for differences in focus on the different components. The similarity *sim*, between A and B, is defined in terms of the semantic concepts common to A and B: S(A)∩S(B), the concepts that are distinctive to A: S(A)–S(B), and the features that are distinctive to B: S(B) – S(A). Two nodes, node A and node B are said to be semantically related if their semantic similarity measure, *sim(A,B)*, exceeds the user-defined similarity threshold t ($0<t\leq 1$).

3.2 Community Construction

The construction of an ontology-based overlay is a process of finding semantically related neighbors. A node joins the network by connecting to one or more bootstrapping neighbors. The bootstrapping neighbors try to recommend some other neighbors to this new node according to their semantic. If the bootstrapping neighbors do not have such recommendation information at hand, the new joining node will issue a neighbor-discovery query. The neighbor discovery query contains the new node's ontology summary compressed with a Bloom Filter [25]. It then uses strategies (such as [13-15]) to efficiently propagate the neighbour discovery query over clusters. Nodes receiving the query compute its semantic similarity with the new node based on the semantic summary. Semantically related nodes then return a positive reply to the new node. If there are not enough neighbors discovered within the hops limited by TTL, the new node will turn to the directory overlay for assistance. After the neighbor-discovery process, a new node is positioned to the right community. Inside the community overlay, nodes randomly connect with their neighbors. Queries looking for particular contents can be forwarded inside the community overlay using flooding- or random-walk-based simple forwarding algorithms.

Because of the dynamic property of the large-scale network, and the evolution of nodes' ontology property, neighbor discovery for a node is not once and for all, but rather the first-step of the topology adaptation scheme. Based on the query experiences a node may add or delete neighbors accordingly. At the same time, it recommends new neighbors to its existing neighbors. As a result, the network topology is reconfigured with respect to peers' dynamic semantic properties, and peers with similar ontologies are always close to each other.

4 Directory Overlay

As a facilitator and complement of the community overlay, the directory overlay indexes top-level semantic interests and unpopular semantic concepts. As mentioned,

OWL ontology can be divided into two parts: TBox and ABox. Similar to a database schema, a node's TBox knowledge is more abstract, describing the node's high-level concepts and their relationships. In contrast, ABox includes concrete data and relations, for example, the instances of classes defined in the TBox. Directory overlay indexes TBox and ABox ontology for different purpose: TBox indexing helps nodes locate communities, while ABox indexing assists nodes finding instances which cannot be quickly located in the community overlay.

The directory overlay is constructed according to the mechanism of the corresponding DHT overlay. We employ RDFPeer's indexing method presented by M. Cai *et al* [9]. The basic idea is to divide RDF description into triples and then index the triples in a DHT overlay. We store each triple three times by applying a hash function to its *subject*, *predicate*, and *object*. In this way, a query providing partial information of a triple can be handled. Peers register their top semantic interests in the form of TBox ontology through the *insert(key,value)* operation in the directory overlay. The directory overlay node in charging of that key maintains a Least Recently Used (LRU) cache storing contact information of registered peers. A neighbor discovery query can get contacts of other peers interested in the same ontology through this directory overlay node. Then the new node can connect with these contacts and join their community. At the same time, the new node registers to the directory overlay by adding itself to the cache of the indexing node. A node with multiple interests can register with multiple indexing nodes. The directory overlay also indexes unpopular ABox instances which cannot be quickly located inside the community.

5 Semantic Query Evaluation

The semantic community reduces the search time and decreases the network traffic by minimizing the number of messages circulating between nodes. There are many strategies, such as [13-15] to effectively propagate queries in an unstructured P2P network. Popular data items are more likely to be located quickly since they have more replicas in the community, whereas an unpopular data item cannot be found unless a large number or all of the peers are searched. Also, queries for data in other semantic communities are unlikely to be solved inside the local community overlay. For these cases, nodes turn to the directory overlay to get assistance.

Directory overlay indexes top semantic interests and unpopular instances, thus is able to give hints to queries which cannot be solved by the community overlay. A node can find interested community by lookup its interest in the directory overlay, then connects to all related nodes returned. For unpopular ABox instances, DHT indexing has the semantic ambiguity problem. For example, it is difficult to figure out whether the search term *palm* is a company (company: palm), a technology (operating system: palm), or a product (PDA:palm). We solve the ambiguity problem with community recommendation feedbacks.

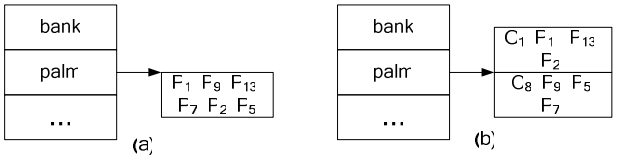

Fig. 1 Example of a data entry stored in an index node

To facilitate query refinement with community feedback, the indexing peers need to perform some additional tasks. Besides storing the ABox keywords, an indexing peer is also responsible for maintaining clusters of peers related to each sense of the keyword. Figure 1 shows an example of a data entry stored in an indexing peer. There are six peers related to the term, *palm*. Initially when a node issues a query related to term *palm* trying to find information about a PDA, all six peers are returned to the requester as shown in Figure 1 (a). The requester will contact each of them, although only three of them (P_1, P_{13}, P_2) are related to PDA. After the requester contacts all these six peers and evaluates their data, it returns its feedback (i.e., which peers are related) to the indexing peer. The indexing peer will link those related three peers with the requester's community, as shown in Figure 1 (b). Next time, a requester from the same community will take advantage of this clustering and be given only the three related peers. In this way, the precision of the query evaluation is improved and the network traffic is reduced.

6 Experiment

As it is difficult to find representative real world ontology data, we have chosen to generate test data artificially. The algorithm starts with generating the ontology schema (TBox). Each schema includes the definition of a number of classes and properties. The classes and properties may form a multilevel hierarchy. Then the classes are instantiated by creating a number of individuals of the classes. To generate an RDF instance triple t, we first randomly choose an instance of a class C among the classes to be the subject: *sub(t).* A property p of C is chosen as the predicate *pre(t)*, and a value from the range of p to be the object: *obj(t).* If the range of the selected property p are instances of a class C', then *obj(t)* is a resource; otherwise, it is a literal. The queries are generated by randomly replacing parts of the created triples with variables.

The directory overlay is implemented as a Pastry [6] virtual network in Java. Each peer is assigned a 160-bit identifier, representing 80 digits (each digit uses 2 bits) with base b=2. After the network topology has been established, nodes publish their TBox knowledge and some unpopular ABox data on the overlay network. Then nodes are randomly picked to issue queries. Each experiment is run ten times with different random seeds, and the results are the average of these ten sets of results.

We examine the system performance in three different aspects, namely scalability, efficiency, and precision by executing the experiment in different network

configurations. The performance is measured using two Information Retrieval (IR) standards: recall and precision. Recall refers to completeness of retrieval of relevant items, as defined below:

$$recall = \frac{|\, relevantDocuments \cap retrievedDocuments \,|}{|\, relevantDocuments \,|}$$

Precision measures the purity of the search results, or how well a search avoids returning results that are not relevant. The "document" in the IR definition represents a resource in our experiment.

$$precision = \frac{|\, relevantDocuments \cap retrievedDocuments \,|}{|\, retrievedDocuments \,|}$$

First, we vary the number of nodes from 2^9 to 2^{15} to test the scalability of the system. The results are listed in Figure 2. Our hybrid system gets higher recall in all these different sized networks. In addition, our recall decreases less with the increase in network size.

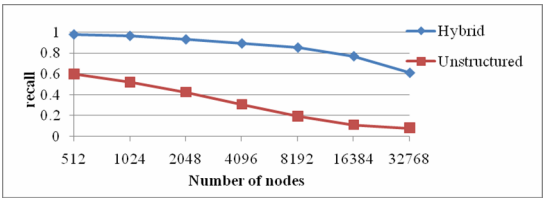

Fig. 2 Recall rate vs. network size

Figure 3 illustrates the system efficiency by showing the relationship between query recall rate and query TTL. With a small TTL, our system gets a higher recall rate, i.e., resolves queries faster.

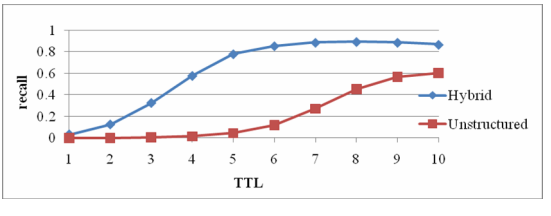

Fig. 3 Recall rate vs. TTL

To testify the effect of community recommendation, we create a special experimental scenario which uses a small-sized dictionary D to generate the ontology data. We randomly pick S words from D, representing polysemy or homonymy (words with multiple meanings); if these words appear in different communities, they represent different meanings. In this experiment, we count the number of

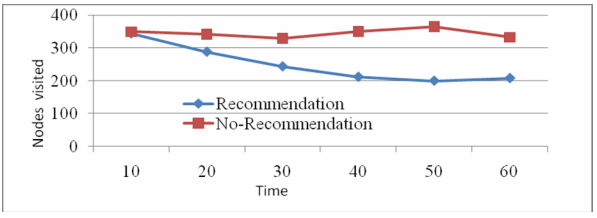

Fig. 4 Effect of recommendation

nodes visited to find 30 results at different time period. As shown in Figure 4, with the time going, using community feedback may reduce the number of nodes to be explored. Because feedback from communities helps eliminating semantic ambiguity of the directory overlay, queries are only forwarded to the most relevant nodes. Consequently, the precision of the search is increased.

7 Conclusion

The main contribution of this paper is to present an effective framework for query evaluation in a large-scale distributed network. Our system combines the structured and unstructured P2P topology to form a hybrid architecture. We organize nodes' topology according to their semantic similarity, so that queries can be focused in semantically related regions only. For queries that cannot be effectively resolved in the semantic community, they can be sent to a structured directory overlay. Recommendations from users are cached for disambiguating future queries. Simulation experiments demonstrate that this framework improves the scalability, efficiency and precision of search in a large semantic heterogeneous network.

References

1. Berners-Lee, T., Fischetti, M.: Weaving the Web, The original design and ultimate destiny of the World Wide Web, Harper (1999)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American (May 2001)
3. Fensel, D., Musen, M. (eds.): The Semantic Web: A Brain for Humankind. IEEE Intelligent Systems (March/April 2001)
4. Gruber, T.R.: Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal Human-Computer Studies (1995)
5. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing, Technical Report, UCB 2000 (2000)
6. Rowstron, A., Druschel, P.: Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, p. 329. Springer, Heidelberg (2001)

7. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: ACM SIGCOMM (2001)
8. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: ACM SIGCOMM (2001)
9. Cai, M., Frank, M.: RDFPeers: A scalable distributed RDF repository based on a structured peer-to-peer network. In: Proc. of WWW conference, New York, USA (May 2004)
10. OntoGrid project, `http://www.ontogrid.net/`
11. Nejdl, W., et al.: EDUTELLA: a P2P Networking Infrastructure Based on RDF. In: Proceedings of the 11th international conference on World Wide Web (WWW) (2002)
12. Arumugam, M., Sheth, A., Arpinar, I.B.: Towards peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web. In: Proc. of the International World Wide Web Conference (2002)
13. Li, J., Vuong, S.: SOON: A Scalable Self-Organized Overlay Network for Distributed Information Retrieval. In: Proceedings of the 19th IFIP/IEEE International Workshop on Distributed Systems (2008)
14. Chawathe, Y., Ratnasam, S., Breslau, L., Lanhan, N., Shenker, S.: Making Gnutella-like P2P Systems Scalable. In: Proceedings of ACM SIGCOMM 2003 (2003)
15. Li, J., Vuong, S.: Efa: an Efficient Content Routing Algorithm in Large Peer-to-Peer Overlay Networks. In: Proceedings of the Third IEEE Peer-to-Peer Computing (2003)
16. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, Cambridge (2003)
17. Jiang, J., Conrath, D.: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In: Proceeding of the Int'l Conf. Computational Linguistics (1997)
18. Lee, J., Kim, M., Lee, Y.: Information Retrieval Based on Conceptual Distance in IS-A Hierarchies. J. Documentation (1993)
19. M. A. Rodriguez, M. J. c
20. Lin, D.: An information-theoretic definition of similarity. In: Proc. 15th International Conf. on Machine Learning, pp. 296–304. Morgan Kaufmann, San Francisco (1998)
21. Bloom, B.: Space/time tradeoffs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)