A Scalable Indexing Mechanism for Ontology-Based Information Integration

Yingjie Li[1], Abir Qasem[2], Jeff Heflin[1]
[1]Department of Computer Science and Engineering, Lehigh University
19 Memorial Dr. West, Bethlehem, PA 18015, U.S.A.
{yil308, heflin}@cse.lehigh.edu
[2]Department of Computer Science, Bridgewater College
402 East College Street, Bridgewater, VA 22812, U.S.A.
aqasem@bridgewater.edu

Abstract

In recent years, there has been an explosion of publicly available RDF and OWL web pages. Typically, these pages are small, heterogeneous and prone to change frequently. In order to effectively integrate them, we propose to adapt a query reformulation algorithm and combine it with an information retrieval inspired index in order to select all sources relevant to a query. We treat each RDF document as a bag of URIs and literals and build an inverted index. Our system first reformulates the user's query into a set of subgoals and then translates these into Boolean queries against the index in order to determine which sources are relevant. Finally, the selected data sources and the relevant ontology mappings are used in conjunction with a description logic reasoner to provide an efficient query answering solution for the Semantic Web. We have evaluated our system using ontology mappings and ten million real world data sources.

1 Introduction

In recent years, many large semantic web knowledge bases (like DBPedia) have followed Linked Open Data[1] guidelines and exposed their contents via a set of dynamically generated web pages about each resource contained within. Unlike traditional applications in distributed databases, these semantic web-enabled pages are often small (containing fewer than 50 triples), heterogeneous (committing to many different schemas or ontologies) and are prone to change frequently. The field of information integration has developed algorithms for querying distributed, heterogeneous databases in such an situation. However, this work often assumes a small number of total data sources, and that human-generated summaries of the content of each are available.

In this paper we build upon our earlier work in ontology-based information integration [5]. In that work, we defined relevance files and adapted the PDMS information integration algorithm [3] to reformulate the query into alternatives to collect relevant sources. The most significant drawbacks to this original approach are: the reliance on human-generated relevance statements for each source, and the inability of the relevance language to satisfactorily summarize sources like people's home pages.

We make two technical contributions in this paper. First, we combine our previous reformulation algorithm with an index inspired by the field of information retrieval (IR). This index is then used to select potentially relevant sources. Second, we conduct a number of experiments to evaluate the characteristics of our system, and demonstrate that it improves upon its predecessor not only by allowing fully automatic index generation, but also by being more selective while retaining completeness.

The rest of the paper is organized as follows: Section 2 describes specific details about our index based source selection algorithm. Then, section 3 presents the experiments that we have conducted to evaluate the algorithm. In section 4, we review related works. Finally, section 5 concludes and discusses future work.

2 Index-based Source Selection Algorithm

Our Index-based Source Selection Algorithm is inspired by the IR techniques. It is well-known that IR approaches treat each document as a bag of words. Unlike typical document collections, the units in semantic web documents are triples not words. Each triple consists of three parts: subject, predicate and object. Both the subject and predicate are always URIs, but the object could be a URI or a Literal

[1]http://linkeddata.org/

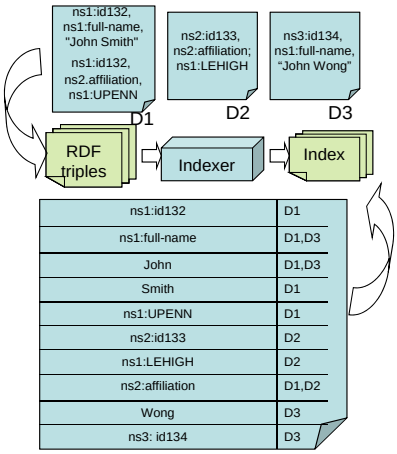


Figure 1. RDF inverted index

/ datatype value. If we let U be the set of URIs and L be the set of Literals, then an RDF document d has the form $d \sqsubseteq U \times U \times (U \sqcup L)$. We then treat an RDF document as a bag of URIs and Literals and create an inverted index. Literal values are indexed using standard IR techniques for free text, thus, each word in the literal is treated as a separate term. In this way, our system can support queries that involve partial string matches. Thus, the terms of the document can be formally expressed as following:

$terms(d) \equiv \{x| < s, p, o > \in d \wedge [x \equiv s \vee x \equiv p \vee (o \in U \wedge x \equiv o) \vee (o \in L \wedge x \in lit - terms(o))]\}$

where $lit\text{-}terms()$ is a function that extracts terms from literals, and may involve typical IR techniques such as stemming and stopwords. Given a document collection D, the dictionary of our system is then $\bigcup_{d \in D} terms(d)$. We have implemented an algorithm that takes an RDF document d and its identifier (e.g., a URL) as input, constructs a virtual document consisting of $terms(d)$, and then indexes this document using Lucene. An example inverted index is given in Figure 1.

Given a conjunctive query, we first adapt the OBII-GNS reformulation algorithm [6] to reformulate the given query into a set of subgoals. Then, an index-compatible Boolean query is constructed to identify potentially relevant data sources. This process is described in Algorithm 1. This algorithm constructs index-compatible Boolean queries and identify potentially relevant data sources. In this process, we need to explain the translation of class membership. After query reformulation, the class membership query would be transformed into a goal node of ns:$Class(x)$ (suppose 'ns' stands for the namespace and '$Class$' stands for the name of one domain class defined in 'ns'). In this case, when we are constructing our Boolean queries, we need to expand the class membership query into the triple pattern with the form "ns:$Class$ AND rdf:$type$". For example, consider a query reformulated into the goals $\{u$:$Professor(x)$, u:$teaches(x$, cs:$proglang)$, j:$works\text{-}at(x, y)\}$, then its Boolean query has the form of (u:$Professor$ AND rdf:$type$) OR (u:$teaches$ AND cs:$proglang$) OR (j:$works\text{-}at$). More details about this algorithm can be found in [4].

Algorithm 1 Source selection by index

SELECTED-BY-INDEX(RQN: Reformulated Query Nodes)

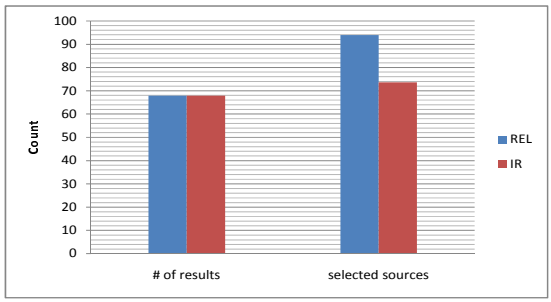
1: $sources \leftarrow \emptyset$
2: **for** each $n \in RQN$ **do**
3: **if** n $typeOf$ $Unary\_Node$ **then**
4: $qterm \leftarrow$ "(*rdf*:*type* AND" $+ n.predicate +$ ")"
5: **else**
6: $qterm \leftarrow$ "(" $+ n.predicate$
7: **if** $n.subject$ $typeOf$ $Constant$ **then**
8: $qterm \leftarrow qterm$+" AND "+$n.subject$
9: **if** n $typeOf$ $owl:ObjectProperty$ **then**
10: **if** $n.object$ $typeOf$ $Constant$ **then**
11: $qterm \leftarrow qterm$+" AND "+$n.object$
12: **else**
13: **if** n $typeOf$ $owl:DatatypeProperty$ **then**
14: $lterms \leftarrow$*lit−terms*(n)
15: **for** each $lterm \in lterms$ **do**
16: $qterm \leftarrow qterm +$ " AND $lterm$"
17: $boolean\_query.add(qterm +$ ") OR")
18: $sources \leftarrow askIndex(INDEX, boolean\_query)$
19: **return** $sources$

3 Evaluation

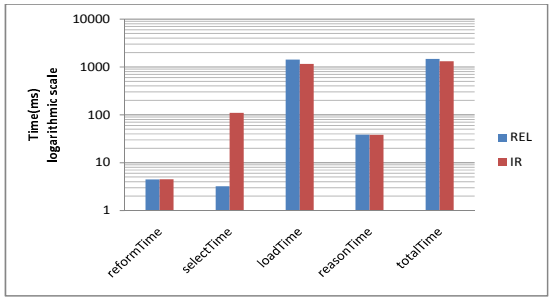
In this section, we have conducted experiments on both synthetic and real world data sets to evaluate our system's source selectivity and scalability. The first is done on a desktop with P4 2.6G CPU and 3G memory running Windows XP professional. The second is done on a workstation with Xeon 2.93G CPU and 6G memory running UNIX. In the indexer implementation, we currently use Lucene as our index builder. In all cases, we use KAON2 as our Reasoner.

3.1 Source Selectivity Evaluation

Our first experiment attempts to demonstrate that the IR-inspired index is superior to the relevance file indices. We will use IR and REL, respectively to distinguish these two approaches. We conducted this experiment with 50 ontologies, 1000 data sources, and a diameter of 6, meaning that the longest sequence of mapping ontologies between any two domain ontologies is six. In this experiment, our IR index size is 10.8MB with the size of original data sources being 21.5MB. The time to construct the IR index is 5,094ms,



(a)



(b)

Figure 2. Source selection and Response time of IR and REL

while it takes 14,593ms to construct the REL index. We issue 200 random queries.

Figure 2(a) displays the average number of results and average number of selected sources for each query. Observe that IR is more selective than REL in source selection but the query answers are still guaranteed to be the same. In this result, we select approximately 23% fewer sources than in the REL method without losing any completeness. Figure 2(b) compares the response time of both systems including time to reformulate the query (reformTime), time to select sources (selectTime), time to load sources from local disk files (loadTime) and time spent by KAON2 reasoner to answer the query using only the loaded sources (reasonTime). The key observation here is that the totalTime of IR is around 10% smaller than that of REL (1317.655ms vs 1478.385ms). The reason is that in both systems, loading sources is the dominant system cost, so fewer sources selected result in big gains. It should be mentioned that the IR system has a worse select time than REL. This is because the REL system uses a memory-based index, while IR uses a disk-based index to achieve greater scalability.

3.2 Scalability Evaluation

In this section, we will evaluate our system's scalability by using set of real world data sources. We choose a subset of the Billion Triple Challenge (BTC) 2009 data set, focusing on four collections: http://data.semanticweb.org/, http://sws.geonames.org/, http://dbpedia.org and http://dblp.rkbexplorer.com. The total number of triples in this dataset is 73,889,151, which are scattered in 21,008,285 documents. The documents confirm our earlier claims about small size, varying from roughly 5 to 50 triples each. Meanwhile, we also create some mapping ontologies to integrate these heterogeneous documents. Based on this dataset, we have designed 4 queries to evaluate our system (Table 1). The # of terms for each query are determined by the mapping ontologies and local axioms defined for the selected data sources. In this experiment, our index construction time is around 58 hours and its size is around 18GB. Each document takes 10ms on average to be indexed.

Query	Query string	# of Reformulated query terms
1	?person dbpedia:name "James A. Hendler"	6
2	?paper swrc:author swrc:abir-qasem ?paper swrc:author swrc:jeff-heflin	4
3	?person swrc:affiliation swrc:lehigh-university	5
4	?person akt:full-name "Jeff Heflin" ?person swrc:affiliation ?org	11

Table 1. Test queries

Table 2 shows the source selectivity of our system by triple level and document level. Observe that our index based mechanism is quite selective for our designed queries in both terms of number of triples selected and number of documents selected. Our metrics mainly focus on the triple selectivity and the document selectivity. The triple/document selectivity is the ratio of the number of selected triples/documents over the total number of the triples/documents. In this result, both our triple and document selectivity are less than 0.1%. Figure 3 shows the performance of our system for answering these four queries. Observe that our system can scale well to real world data with reasonable reformTime, selectTime, loadTime and reasoning time (note the logarithmic scale). The third observation is that our system performs better for queries Q1, Q2 and Q3 with selective terms such as "James A. Hendler" and "Jeff Heflin". This is because these terms make our system select less sources. For those queries without selective terms such as Q4 having a triple with two variables, the system's performance become worse than that of Q1, Q2 and Q3 even when no answers are returned. We will address this problem in future work.

In addition, we compared the selector component of our system with Sindice. The motivation is to see how effective a local "bag of URIs" index is vs. querying a remote semantic web search engine to retrieve the relevant triples. The

Query	# of Results	# of Selected triples	# of Selected documents
1	142	715	143
2	2	46	9
3	15	163	20
4	16	25342	5069

Table 2. Source selectivity

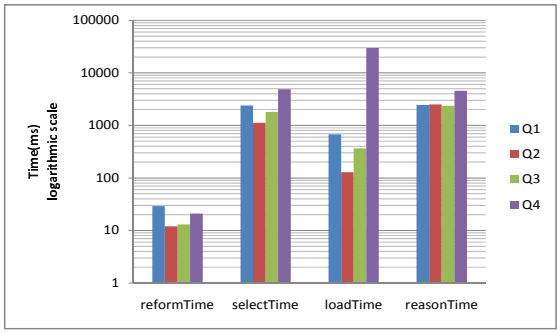


Figure 3. Performance of IR

experimental results show that our system performs faster than that of using Sindice. Due to limited space, we can not describe its details. We refer the readers to Li et al. [4] for more.

4 Related Work

Currently, there are mainly two areas of work related with our paper: Information Integration and RDF indexes.

In Information Integration, Haase and Motik [2] developed a mapping system for OWL that involves relating conjuctive queries. However, they do not explicitly address the issue of distributed data, and provide no means of indexing the relevant sources. Peer-to-peer (P2P) semantic web systems like Bibster [1] address the distributed nature of the Web, but are insufficient for our purpose because they rely on a common ontology. Peers in Bibster might have different data, but use the same ontologies. Heremes [7] translates a keyword query provided by the user into a federated query and then decomposes this into separate SPARQL queries that are issued to web data sources. However, it does not account for rich schema heterogeneity.

Regarding work on RDF indexes, Hexastore attempts to achieve scalability by replicating each triple six times: one for each sorting order of subject, predicate and object [9]. It has been demonstrated that this strategy results in good response time for conjunctive queries. The major disadvantage of this approach is that the indexes are quite expensive in terms of space. GRIN is a novel index developed specifically for graph-matching queries in RDF [8], but it still is not clear how it could be adapted for a distributed context.

5 Conclusions and Future Work

In this paper, we have proposed a scalable IR-inspired indexing mechanism for ontology-based information integration. The experiments demonstrated that our new system is better than our prior work with higher source selectivity and 10% improvement in response time. We have also shown our system is able to respond to many queries on a 20 million document real world data set in seconds.

However, there is still significant room for improvement. First, our current algorithm's performance will decline significantly when there are triple patterns that lack constants. We intend to develop an optimizer that can use the most selective triple patterns to solve this kind of queries. Second, our algorithm needs to be adapted to locate relevant $owl{:}sameAs$ statements. We believe that solving such problems will lead to a pragmatic solution for querying a large, distributed, and ever changing Semantic Web.

References

[1] P. Haase, J. Broekstra, M. Ehrig, M. Menken, P. Mika, M. Olko, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *International Semantic Web Conference*, pages 122–136, 2004.

[2] P. Haase and B. Motik. A mapping system for the integration of owl-dl ontologies. In *IHIS '05: Proceedings of the first international workshop on Interoperability of heterogeneous information systems*, pages 9–16, New York, NY, USA, 2005. ACM.

[3] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. *Data Engineering, International Conference on*, page 505, 2003.

[4] Y. Li, A. Qasem, and J. Heflin. A scalable indexing mechanism for ontology-based information integration. Technical Report LU-CSE-10-001, Lehigh University, 2010.

[5] A. Qasem, D. A. Dimitrov, and J. Heflin. Efficient selection and integration of data sources for answering semantic web queries. *International Conference on Semantic Computing*, pages 245–252, 2008.

[6] A. Qasem, D. A. Dimitrov, and J. Heflin. Goal node search for semantic web source selection. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, pages 566–569, 2008.

[7] T. Tran, H. Wang, and P. Haase. Hermes: Data web search on a pay-as-you-go integration infrastructure. *Web Semant.*, 7(3):189–203, 2009.

[8] O. Udrea, A. Pugliese, and V. S. Subrahmanian. Grin: A graph based rdf index. In *AAAI*, pages 1465–1470, 2007.

[9] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *Proc. VLDB Endow.*, pages 1008–1019, 2008.