Crawling and Querying Linked Data

Andreas Harth

Joint work with Aidan Hogan, Juergen Umbrich, Marcel Karnstedt, Katja Hose, Robert Isele, Kai-Uwe Sattler, Axel Polleres, Stefan Decker
Outline

- Motivation
- Linked Data Principles
- Crawling Linked Data
- Query Processing over Linked Data
- Conclusion
Motivation

- With increased use of computers more and more data is being stored
  - Organisations rely on data for business decisions
  - Data drives policy decisions in government
  - Individuals rely on data from the Web for information and communication

- Data volumes explode
  - More and more data available on the Web is represented in Semantic Web standards
  - Linking Open Data (LOD) Initiative

- Semantic Web technologies facilitate the integration of data from multiple sources
- Combining data from multiple sources enables insights
Motivation

There are some questions that can’t be answered by Google.

KIT – University of the State of Baden-Wuerttemberg and National Laboratory of the Helmholtz Association
Sample Queries

- Q: news about KIT?
- Q: key people of competitors of IBM?
- Q: funding pattern of Sequoia Capital?
Data Integration with Semantic Web Technologies
Step 1: Data Preparation – Common Data Format
Step 2: Data Integration
Step 3: Interactive Data Exploration

1. Query
2. Answer
Linked Data

- Linked Data provides
  - common data format and access mechanism
  - data integration (~interlinking)
- on a global scale!

- Uses traditional web architecture (URIs, REST)
- Plus a bit of Semantic Web (RDF)

- Scale: in terms of technology
- Scale: in terms of uptake potential
Linked Data Principles*

1. Use **URIs to name things**; not only documents, but also people, locations, concepts, etc.
2. To enable agents (human users and machine agents alike) to look up those names, use **HTTP URIs**
3. When someone looks up a URI we **provide useful information**; with 'useful' in the strict sense we usually mean structured data in RDF.
4. Include **links to other URIs** allowing agents (machines and humans) to **discover more things**

(*) http://www.w3.org/DesignIssues/LinkedData
Correspondence between thing-URI and source-URI

User Agent

HTTP GET

RDF

Web Server

http://www.polleres.net/foaf.rdf#me

http://www.polleres.net/foaf.rdf
Correspondence between thing-URI and source-URI

User Agent

Web Server

http://dbpedia.org/resource/Gordon_Brown

http://dbpedia.org/data/Gordon_Brown

http://dbpedia.org/page/Gordon_Brown
Linked Data Application: Minimal Architecture

User Interface & Applications

Query: SPARQL

Data interchange: RDF, XML, URI/IRI

1. Query
2. Answer
Queries over Linked Data

```sparql
SELECT ?f ?n WHERE {
  timbl:i foaf:knows ?f.
  ?f foaf:name ?n.
}

SELECT ?x1 ?x2 WHERE {
  dblppub:HoganHP08 dc:creator ?a1.
  ?x1 owl:sameAs ?a1.
  ?x2 foaf:knows ?x1.
}
```
Querying Data Across Sources

- Data warehousing or materialisation-based approaches (MAT)

CRAWL → INDEX → SERVE
Example Linked Data Graph

Goal: collect all data
Linked Data Crawler Architecture

1. Get URI from a queue
2. Open connection and fetch content
3. Process and store content
4. Extract new links and put into queue
5. At defined intervals: schedule URIs in queue

Scheduling

- Scheduling done at defined intervals (round-based crawling)
- Reorder URIs for next round
- Wait min. amount of time to avoid denial-of-service attack

Implementation of graph-traversal methods
- breadth-first
- random walks
- best-first search using heuristics to bias for quality or topic
- download-optimised heuristic
Breadth-First Traversal
LDSpider

- Open source Java implementation of a multi-threaded crawler for Linked Data
- Works on small- to medium-sized datasets (up to a couple of hundred million triples)
- Two scheduling methods (breadth-first and optimised for download speed)
- Extendable via hooks

http://code.google.com/p/ldspider
Querying Data Across Sources

- Data warehousing or materialisation-based approaches (MAT)

- Distributed query processing approaches (DQP)
SELECT ?f ?n WHERE {
  timbl:i foaf:knows ?f.
  ?f foaf:name ?n.
}
Problem: Source Selection for Triple Patterns

- (s p o)
- (s p o)
- (s p o)
- (s p o)
- (s p o)
- (s p o)
- (s p o)
- (s p o)
- (s p o)

- Given a triple pattern, which source can contribute bindings for the triple pattern?
Schema-Level Indices [Stuckenschmidt et al. 2004]

- Keep index of properties and/or classes contained in sources

- \( (?s \#p \?o) \), \( (?s \text{ RDF:type} \#o) \)

- Covers only queries containing schema-level elements

- Commonly used properties select potentially too many sources

```sql
SELECT ?f ?n WHERE {
  timbl:i foaf:knows ?f.
  ?f foaf:name ?n.
}
```

```sql
SELECT ?x1 ?x2 WHERE {
  dblppub:HoganHP08 dc:creator ?a1.
  ?x1 owl:sameAs ?a1.
  ?x2 foaf:knows ?x1.
}
```
Direct Lookup (DL) [Hartig et al. 2009]

- Exploits correspondence between thing-URI and source-URI
- Linked Data sources (aka RDF files) return typically triples with a subject corresponding to the source
- Sometimes the sources return triples with object corresponding to the source

- (#s ?p ?o), (#s #p ?o), (#s #p #o)
- (?s ?p #o), (?s #p #o)

- Incomplete wrt. patterns but also wrt. to URI reuse across sources
- Limited parallelism, unclear how to schedule lookups

```sql
SELECT ?f ?n WHERE {
  timbl:i foaf:knows ?f.
  ?f foaf:name ?n.
}
```

```sql
SELECT ?x1 ?x2 WHERE {
  dblppub:HoganHP08 dc:creator ?a1.
  ?x1 owl:sameAs ?a1.
  ?x2 foaf:knows ?x1.
}
```
Approximate Data Summaries

- Combined description of schema-level and instance-level
- Use approximation to reduce index size (incurs false positives)
- Possible to use entire query for source selection
- Parallel lookups since sources can be determined for the entire query

- (?s ?p ?o), (#s ?p ?o), (?s #p ?o), (?s ?p #o), (#s #p ?o), (#s ?p #o), (?s #p #o)
- and combinations of triple patterns

SELECT ?f ?n WHERE {
  timbl:i foaf:knows ?f.
  ?f foaf:name ?n.
} 

SELECT ?x1 ?x2 WHERE {
  dblppub:HoganHP08 dc:creator ?a1.
  ?x1 owl:sameAs ?a1.
  ?x2 foaf:knows ?x1.
}
Hash-based Data Summaries

- Source: axelpol.rdf : (#axel foaf:homepage http://polleres.net/axel/)
- 1. Hash each triple element e.g. to (323, 232, 124)
- 2. Insert into equi-width histogram bucket
- 3. Lookup (e.g. (#axel ?p ?o) to determine relevant buckets containing source-URIs
Enter QTrees

- Combination of histograms and R-trees inheriting the benefits of both data structures
  - indexing multidimensional data
  - capturing attribute correlations (buckets – hash function) cluster together
  - dealing with sparse data (does not need to span entire data space)
  - efficient lookups via MBBs
QTREE STRUCTURE

- Data space represented as tree
- Nodes represent minimum bounding boxes (MBBs)
- Buckets
  - Store cardinality and set of sources, e.g. (5, { axelpol.rdf, …})
  - Store cardinality per source: e.g. source { axelpol.rdf:5 }

![Diagram of QTree Structure]
Join Estimation in QTree

- 1st triple pattern lookup
- 2nd triple pattern lookup
- determine which buckets overlap in join dimension (e.g. subject dimension)
- check how much they overlap – assume equal distribution to estimate results cardinality
- do parallel lookup on top-k (or all) sources
Experiments Setup

- Breadth-first crawl from http://www.w3.org/People/Berners-Lee/card
- 3m triples from 16k sources

- 100 generated queries in star- and path-shape (S-1, S-2, P-1, P-2, P-3)

- Benefit
- Effect of ranking
- Performance
Benefit

- Total number of sources $T$ in the data
- Number of estimated sources $E$
- Benefit: $1.0 - \frac{E}{T}$
Conclusion

- Linked Data provides means for common access to large amounts of data
- Query processing via direct lookups (DL) over Linked Data works
- QTree is memory efficient and can support the direct lookup query approach
- Possible to combine crawling or DL (for index priming) with QTree
- Exciting and novel research problem
  - lots of sources with little amounts of data
  - community-driven collaborative knowledge engineering
  - network lookup on URIs with complexity $O(1)$ (at a sufficiently high level of abstraction)