LDIF - A Framework for Large-Scale Linked Data Integration

Andreas Schultz, Freie Universität Berlin
Andrea Matteini, mes | semantics
Robert Isele, Freie Universität Berlin
Pablo N. Mendes, Freie Universität Berlin
Christian Bizer, Freie Universität Berlin
Christian Becker, mes | semantics

With contributions from:
Hannes Mühleisen, Freie Universität Berlin
Outline

- Challenges for consuming Linked Data
- LDIF – Linked Data Integration Framework
  - Data Acquisition
  - Data Translation
  - Identity Resolution
  - Quality Assessment and Fusion
  - Output
- Benchmarks
Linked Open Data
Linked Data Challenges

- The Web of Data is heterogeneous
- Many different vocabularies are in use
- Over 60% of all Linked Data sources use proprietary vocabularies\(^1\)

⇒ Consumer has to normalize the vocabularies

Most widely used vocabularies in the LOD cloud (08/10/2011)

\(^1\)http://www4.wiwiss.fu-berlin.de/lodcloud/state/
Linked Data Challenges

- Over 30 billion triples published as Linked Open Data
  - But only 500 million links

- Many data sources are not sufficiently interlinked
  - Most data sources only link to one other data source

⇒ Consumer needs to generate additional links

LOD data sets by the number of other data sources that are target of outgoing RDF links
Linked Data Challenges

- The quality of data on the Web is very mixed
- Linked Data Providers differ in:
  - Levels of knowledge
  - Views of the world
  - Different intension
- Information in Linked Data Sources may be:
  - Wrong
  - Biased
  - Inconsistent
  - Outdated
LDIF – Linked Data Integration Framework

- LDIF:
  - Gather Linked Data from the Web
  - Translate the data into a clean local target representation
  - While keeping track of data provenance
- Open source (Apache License, Version 2.0)
- Collaboration between Freie Universität Berlin and mes|semantics
- Available for download at: http://ldif.wbsg.de/
Architecture of Linked Data Applications

Application Layer

Application Code

Data Access, Integration and Storage Layer

LDIF

Web Data Access Module → Data Translation Module → Identity Resolution Module → Data Quality and Fusion Module → Integrated Web Data

Web of Data

Publication Layer

LD Wrapper
- Database A
- Database B
- CMS
- RDFa
- RDF/X ML

HTTP
LDIF Pipeline

1. Collect data
2. Map to Schema
3. Resolve Identities
4. Quality Assessment & Fusion
5. Output
Data Access

• Data can be loaded from a variety of sources:
  • RDF dumps (various formats)
  • SPARQL Endpoints
  • Crawling Linked Data

• We consider a simple use case:
  • We retrieve and integrate 2 data sources:
    • Freebase using crawling
    • MusicBrainz using SPARQL endpoint

• Complete example with 4 data sources at:
  http://www.assembla.com/code/ldif/git/nodes/master/ldif/examples/music/
Freebase

<importJob>
  <dataSource>Freebase</dataSource>
  <refreshSchedule>onStartup</refreshSchedule>
  <crawlImportJob>
    <seedURIs>
    </seedURIs>
    <predicatesToFollow>
      <uri>http://rdf.freebase.com/ns/music.artist.genre</uri>
      <uri>http://rdf.freebase.com/ns/music.genre.albums</uri>
      <uri>http://rdf.freebase.com/ns/music.genre.artists</uri>
      <uri>http://rdf.freebase.com/ns/music.album.artist</uri>
      <uri>http://rdf.freebase.com/ns/music.artist.album</uri>
    </predicatesToFollow>
    <levels>2</levels>
    <resourceLimit>100000</resourceLimit>
  </crawlImportJob>
</importJob>
<importJob>
  <internalId>musicbrainz.3</internalId>
  <dataSource>MusicBrainz_Talis</dataSource>
  <refreshSchedule>onStartup</refreshSchedule>
  <sparqlImportJob>
    <endpointLocation>
      http://api.talis.com/stores/musicbrainz/services/sparql
    </endpointLocation>
    <tripleLimit>5000</tripleLimit>
    <sparqlPatterns>
      <pattern>?s a &lt;http://purl.org/ontology/mo/SoloMusicArtist&gt;</pattern>
    </sparqlPatterns>
  </sparqlImportJob>
</importJob>
Problems

- Different ways to represent the same information
- Different Uris for the same real world object
- Conflicting information
Data Translation

- LDIF uses R2R for vocabulary mapping
- Translates data to a single target vocabulary
- R2R Mapping Language:
  - Mappings expressed in RDF (Turtle)
  - Simple mappings using OWL / RDFs statements (\texttt{x rdfs:subClassOf y})
  - Complex mappings with SPARQL expressivity
  - Transformation functions
Mapping Freebase

<http://rdf.freebase.com/ns/music.artist> rdfs:subClassOf mo:MusicArtist


mp:mapToFoafMember
  a r2r:Mapping;
  r2r:sourcePattern "?SUBJ mo:member_of ?o";
  r2r:targetPattern "?o foaf:member ?SUBJ" .
Mapping MusicBrainz

mp:bioToDBpediaBirthDate a r2r:Mapping ;
  ?event a bio:Birth . ?event bio:date ?o" ;
r2r:targetPattern "?SUBJ dbpedia-owl:birthDate ?'o'^^xsd:date" .
Resolve Identities

- LDIF uses the Silk Link Discovery Framework to resolve identities
- Declarative Silk Link Specification Language is used to define linkage rules
- A Linkage Rule defines the conditions that must hold true for two entities to be considered a duplicate
A linkage rule is represented as a tree consisting of 4 types of operators:

- **RDF paths**
  - Similar to SPARQL 1.1 Property Paths
  - Examples:
    - ?movie/dbpedia:director/rdfs:label
    - ?person/label[@lang='en']

- **Transformations**
  - Transforms the result set of an RDF paths
  - Variety of built-in transformations
  - Examples:
    - LowerCase
    - RegexReplace
    - Stem

- **Similarity Metrics**
  - Similarity of two inputs based on a user-defined metric.
  - Examples:
    - Various string similarity metrics
    - Geographic similarity
    - Date similarity

- **Aggregations**
  - Aggregates multiple similarity metrics
  - Examples:
    - Min, Max, Average
    - Quadratic Mean
    - Geometric Mean
Example Linkage Rule

<LinkageRule>
  <Aggregate type="average">
    <Compare metric="equality">
      <Input path="?a/rdfs:label" />
      <Input path="?b/rdfs:label" />
    </Compare>
  </Aggregate>
  <Aggregate type="max">
    <Compare metric="jaccard" threshold="0.4">
      <Input path="?a\foaf:member/rdfs:label" />
      <Input path="?a\foaf:member/rdfs:label" />
    </Compare>
    <Compare metric="date" threshold="0.0">
      <Input path="?a/dbpediaowl:birthDate" />
      <Input path="?b/dbpediaowl:birthDate" />
    </Compare>
  </Aggregate>
</LinkageRule>
Example

- Entities which identify the same real world object are connected using owl:sameAs links
- owl:sameAs links are to be fused into a single entity
Quality Assessment

- LDIF uses Sieve\(^1\) for Quality Assessment
- Quality Assessment Metrics composed by:
  - ScoringFunction (generically applicable to given data types)
  - Quality Indicator as input (adaptable to use case)
- Output describes input within a quality dimension, according to a user’s definition of quality.
- We consider a simple example with 2 dimensions:
  - Recency
  - Reputation

\(^1\)http://sieve.wbsg.de
Recency

• Configuration:

```xml
<QualityAssessment>
  <AssessmentMetric id="sieve:recency">
    <ScoringFunction class="TimeCloseness">
      <Param name="timeSpan" value="7" />
      <Input path="?GRAPH/provenance:lastUpdated" />
    </ScoringFunction>
  </AssessmentMetric>
</QualityAssessment>
```

• Example:

```
freebase:bob_marley sieve:recency "0.4"
mb:ed2ac1e9-d51d sieve:recency "0.8"
```
Reputation

• Configuration:

```xml
<QualityAssessment>
  <AssessmentMetric id="sieve:reputation">
    <ScoringFunction class="ScoredList">
      <Param name="priority">
        Value="http://musicbrainz.dataincubator.org
               http://rdf.freebase.com/ns" />
    </ScoringFunction>
  </AssessmentMetric>
</QualityAssessment>
```

• Example:

```
freebase:bob_marley  sieve:reputation  "0.45"
mb:ed2ac1e9-d51d     sieve:reputation  "0.9"
```
Data Fusion

“fusing multiple records representing the same real-world object into a single, consistent, and clean representation”

(Bleiholder & Naumann, 2008)

• Input:
  • (Potentially) conflicting data
  • Quality metadata describing input

• Execution:
  • Use existing or custom FusionFunctions

• Output:
  • Clean data, according to user’s definition of clean
Fusion Example

- Configuration:

```xml
<Fusion>
  <Class name="mo:MusicalArtist">
    <Property name="rdfs:label">
      <FusionFunction class="PassItOn" />
    </Property>
    <Property name="dbpedia-owl:birthDate">
      <FusionFunction class="KeepSingleValueByQualityScore"
                      metric="sieve:reputation">
      </FusionFunction>
    </Property>
  </Class>
</Fusion>
```
Result

Bob Marley

1945-02-06

rdfs:label

dbpedia-owl:birthDate

owl:sameAs

mb:ed2ac1e9-d51d

freebase:bob_marley

foaf:member

ldif:bob_marley

ldif:bob_marley_the_wailers
Data Output

• Output options:
  • N-Quads
  • N-Triples
  • SPARQL Update Stream

• Provenance tracking using Named Graphs
Versions

- In-memory
  - keeps all intermediate results in memory
  - fast, but scalability limited by local RAM
- RDF Store (TDB)
  - stores intermediate results in a Jena TDB RDF store
  - can process more data than In-memory but scalability is limited by the RDF store
- Cluster (Hadoop)
  - scales by parallelizing work across multiple machines using Hadoop
  - can process a virtually unlimited amount of data
Benchmarks

KEGG GENES VS. UNIPROT (SINGLE MACHINE)

100M Triples:

![100M run times chart]

- In-memory[2]: 75.4 min
- TDB[3]: 220 min
- Hadoop FUB 2-slaves: 88 min

300M Triples:

![300M run times chart]

- TDB[3]: 560.8 min
- Hadoop FUB 2-slaves: 149 min

Machine: Intel i7 950, 3.07GHz (quadcore), 24GB
Benchmarks

**KEGG GENES VS. UNIPROT (CLUSTER)**

300M Triples:

![300M run times on Hadoop EC2 clusters](image)

3.6B Triples:

![3.69B (UniProt vs. KEGG GENES) run times](image)

Machines: Amazon EC2 c1.medium instances
Thank You!

- Get LDIF at: http://ldif.wbsg.de
- Development of LDIF has been supported in part by:
  - Vulcan Inc. as part of its Project Halo
  - LOD2 - Creating Knowledge out of Interlinked Data
  - PlanetData - A European Network of Excellence on Large-Scale Data Management