Semantic Web
6. Linked Open Data

Gerd Gröner, Matthias Thimm
{groener,thimm}@uni-koblenz.de

Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau

July 12, 2013
RDF, RDFS, OWL vocabularies

- RDF
  - rdf:Statement
  - rdf:type
  - rdf:subject, rdf:predicate, rdf:object
  - ...

- RDFS
  - rdfs:Resource
  - rdfs:Class
  - rdfs:subClassOf

- OWL
  - owl:disjointOf
  - owl:sameAs
  - owl:allValuesFrom
RDF, RDFS, OWL vocabularies

- RDF
  - rdf:Statement
  - rdf:type
  - rdf:subject, rdf:predicate, rdf:object
  - ...

- RDFS
  - rdfs:Resource
  - rdfs:Class
  - rdfs:subClassOf
  - ...

What is missing?
RDF, RDFS, OWL vocabularies

- **RDF**
  - rdf:Statement
  - rdf:type
  - rdf:subject, rdf:predicate, rdf:object
  - ...

- **RDFS**
  - rdfs:Resource
  - rdfs:Class
  - rdfs:subClassOf
  - ...

- **OWL**
  - owl:disjointOf
  - owl:sameAs
  - owl:allValuesFrom
  - ...

What is missing?
RDF, RDFS, OWL vocabularies

- RDF
  - rdf:Statement
  - rdf:type
  - rdf:subject, rdf:predicate, rdf:object
  - ...

- RDFS
  - rdfs:Resource
  - rdfs:Class
  - rdfs:subClassOf
  - ...

- OWL
  - owl:disjointOf
  - owl:sameAs
  - owl:allValuesFrom
  - ...

What is missing?
Linked Open Data

→ everyone has to use it! Properly!
Linked Open Data

- RDF, RDFS, OWL can be used to describe a local isolated data set
- linking resources to other data sets openly enables *linked open data*
Linked Open Data

- RDF encourages to use URIs to denote things
Linked Open Data

- RDF encourages to use URIs to denote things
- Linked Open Data *demands* it
Tim Berners-Lee:

“Don’t say “colour” say <http://example.com/2002/std6#col>”
Outline

1. Linked Open Data Principles
2. Some examples
3. Applications
4. Summary and Exercises
Outline

1. Linked Open Data Principles
2. Some examples
3. Applications
4. Summary and Exercises
Linked Open Data Principles

- use URIs as names for things
- use HTTP URIs so those names can be dereferenced
- return useful information upon lookup of those URIs (e.g. RDF)
- include links by using URIs that dereference to remote documents
Linked Open Data Principles

- use URIs as names for things
- use HTTP URIs so those names can be dereferenced
- return useful information upon lookup of those URIs (e.g. RDF)
- include links by using URIs that dereference to remote documents

Linked Open Data Scheme

- Publish under an open license
- Publish structured data
- Use non-proprietary formats
- Use URIs to identify things
- Link your data to other data
Five stars of Linked Open Data conformance
Publish under an open license

As a consumer

- You can store it locally (on your hard drive or on an USB stick).
- You can enter the data into any other system.
- You can change the data as you wish.
- You can share the data with anyone you like.

As a publisher

- It’s simple to publish.
- You do not have explain repeatedly to others that they can use your data.

5stardata.info
Publish structured data

As a consumer
- You can directly process it with proprietary software to aggregate it, perform calculations, visualise it, etc.
- You can export it into another (structured) format.

As a publisher
- It’s still simple to publish.
Use non-propriety formats

As a consumer

- You can manipulate the data in any way you like, without being confined by the capabilities of any particular software.

As a publisher

- You might need converters or plug-ins to export the data from the proprietary format.
- It’s still rather simple to publish
Use URIs to identify things

As a consumer

▶ You can link to it from any other place (on the Web or locally).
▶ You may be able to reuse existing tools and libraries, even if they only understand parts of the pattern the publisher used.
▶ Understanding the structure of an RDF Graph of data can be more effort
▶ You can combine the data safely with other data.

As a publisher

▶ Other data publishers can now link into your data
▶ You typically invest some time slicing and dicing your data.
▶ You’ll need to assign URIs to data items and think about how to represent the data.
Link your data to other data

As a consumer

▶ You can discover more data while consuming the data.
▶ You can directly learn about the data schema.
▶ You now have to deal with broken data links.
▶ Presenting data from an arbitrary link as fact is as risky as letting people include content from any website in your pages. Caution, trust and common sense are all still necessary.

As a publisher

▶ You make your data discoverable.
▶ You increase the value of your data.
▶ You own organisation will gain the same benefits from the links as the consumers.
▶ You’ll need to invest resources to link your data to other data on the Web.
▶ You may need to repair broken or incorrect links.
Outline

1. Linked Open Data Principles
2. Some examples
3. Applications
4. Summary and Exercises
“DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web.”

“Semantification” of Wikipedia:
- 764,000 persons
- 573,000 places
- 333,000 creative works
- 192,000 organizations
- 202,000 species
- 5,500 diseases.
Excerpt from the DBpedia ontology:

- `owl:Thing`
  - `dbpedia:Agent`
    - `dbpedia:Family`
    - `dbpedia:Organisation`
    - `dbpedia:Person`
  - `dbpedia:Event`
    - `dbpedia:Convention`
    - `dbpedia:FilmFestival`
    - `dbpedia:SportsEvent`
  - `dbpedia:Language`
  - `dbpedia:Name`
  - `dbpedia:Place`
    - `dbpedia:Monument`
    - `dbpedia:PopulatedPlace`
    - `dbpedia:WineRegion`
FOAF

- FOAF = Friend of a Friend
- Vocabulary for describing people, links between them, things they create and do

<table>
<thead>
<tr>
<th>FOAF Core</th>
<th>Social Web</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• nick</td>
</tr>
<tr>
<td></td>
<td>• mbox</td>
</tr>
<tr>
<td></td>
<td>• homepage</td>
</tr>
<tr>
<td></td>
<td>• weblog</td>
</tr>
<tr>
<td></td>
<td>• openid</td>
</tr>
<tr>
<td></td>
<td>• jabberID</td>
</tr>
<tr>
<td></td>
<td>• mbox_sha1sum</td>
</tr>
<tr>
<td></td>
<td>• interest</td>
</tr>
<tr>
<td></td>
<td>• topic_interest</td>
</tr>
<tr>
<td></td>
<td>• topic (page)</td>
</tr>
<tr>
<td></td>
<td>• workplaceHomepage</td>
</tr>
<tr>
<td></td>
<td>• workInfoHomepage</td>
</tr>
<tr>
<td></td>
<td>• schoolHomepage</td>
</tr>
<tr>
<td></td>
<td>• publications</td>
</tr>
<tr>
<td></td>
<td>• currentProject</td>
</tr>
<tr>
<td></td>
<td>• pastProject</td>
</tr>
<tr>
<td></td>
<td>• account</td>
</tr>
<tr>
<td></td>
<td>• OnlineAccount</td>
</tr>
<tr>
<td></td>
<td>• accountName</td>
</tr>
<tr>
<td></td>
<td>• accountServiceHomepage</td>
</tr>
<tr>
<td></td>
<td>• PersonalProfileDocument</td>
</tr>
<tr>
<td></td>
<td>• tipjar</td>
</tr>
<tr>
<td></td>
<td>• sha1</td>
</tr>
<tr>
<td></td>
<td>• thumbnail</td>
</tr>
<tr>
<td></td>
<td>• logo</td>
</tr>
<tr>
<td>Agent</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td></td>
</tr>
<tr>
<td>img</td>
<td></td>
</tr>
<tr>
<td>depiction (depicts)</td>
<td></td>
</tr>
<tr>
<td>familyName</td>
<td></td>
</tr>
<tr>
<td>givenName</td>
<td></td>
</tr>
<tr>
<td>knows</td>
<td></td>
</tr>
<tr>
<td>based_near</td>
<td></td>
</tr>
<tr>
<td>age</td>
<td></td>
</tr>
<tr>
<td>made (maker)</td>
<td></td>
</tr>
<tr>
<td>primaryTopic (primaryTopicOf)</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td></td>
</tr>
<tr>
<td>member</td>
<td></td>
</tr>
<tr>
<td>Document</td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td></td>
</tr>
</tbody>
</table>
Dublin Core

- Vocabulary for describing the creation and publishing of general works

Gerd Gröner, Matthias Thimm
Dublin Core

- Vocabulary for describing the creation and publishing of general works
- Namespace:
  xmlns:dc="http://purl.org/dc/elements/1.1"
- Examples
  - dc:creator
Dublin Core

- Vocabulary for describing the creation and publishing of general works
- Namespace:
  xmlns:dc="http://purl.org/dc/elements/1.1"
- Examples
  - dc:creator
  - dc:title
Dublin Core

- Vocabulary for describing the creation and publishing of general works
- Namespace:
  xmlns:dc="http://purl.org/dc/elements/1.1"
- Examples
  - dc:creator
  - dc:title
  - dc:rights
Dublin Core

- Vocabulary for describing the creation and publishing of general works
- Namespace:
  xmlns:dc="http://purl.org/dc/elements/1.1"
- Examples
  - dc:creator
  - dc:title
  - dc:rights
  - dc:publisher
Dublin Core

- Vocabulary for describing the creation and publishing of general works
- Namespace:
  xmlns:dc="http://purl.org/dc/elements/1.1"
- Examples
  - dc:creator
  - dc:title
  - dc:rights
  - dc:publisher
  - dc:date
Dublin Core

- Vocabulary for describing the creation and publishing of general works
- Namespace:
  xmlns:dc="http://purl.org/dc/elements/1.1"
- Examples
  - dc:creator
  - dc:title
  - dc:rights
  - dc:publisher
  - dc:date
  - dc:description
Dublin Core

- Vocabulary for describing the creation and publishing of general works
- Namespace:
  xmlns:dc="http://purl.org/dc/elements/1.1"
- Examples
  - dc:creator
  - dc:title
  - dc:rights
  - dc:publisher
  - dc:date
  - dc:description
  - dc:source
Dublin Core

- Vocabulary for describing the creation and publishing of general works
- Namespace:
  xmlns:dc="http://purl.org/dc/elements/1.1"
- Examples
  - dc:creator
  - dc:title
  - dc:rights
  - dc:publisher
  - dc:date
  - dc:description
  - dc:source
  - dc:language
Outline

1. Linked Open Data Principles
2. Some examples
3. Applications
4. Summary and Exercises
Applications

What can I do with this?

Linked Data Browsers

Linked Data Mashups

Search Engines

Thing

Thing

Thing

Thing

Thing

Type links

Type links

Type links

Type links
Use case example: SchemEx

Persons that are
- Politicians and
- Actors

? 

courtesy of Ansgar Scherp
SELECT ?x
FROM ...
WHERE {
  ?x rdf:type ex:Actor .
  ?x rdf:type ex:Politician .
}

“Politician and Actor”
Use case example: SchemEx (cont’d)

Naive Approach:

- Download the entire LOD cloud
- Put it into a (really) large triple store
- Process the data and extract schema
- Provide lookup

Drawbacks:

- Big machinery
- Late in processing the data
- High effort to scale with LOD cloud

Can we do it smarter?
Use case example: SchemEx (cont’d)

Naive Approach:
- Download the entire LOD cloud
- Put it into a (really) large triple store
- Process the data and extract schema
- Provide lookup

Drawbacks:
- Big machinery
- Late in processing the data
- High effort to scale with LOD cloud
Naive Approach:
- Download the entire LOD cloud
- Put it into a (really) large triple store
- Process the data and extract schema
- Provide lookup

Drawbacks:
- Big machinery
- Late in processing the data
- High effort to scale with LOD cloud

Can we do it smarter?
Use case example: SchemEx (cont’d)

Idea:

- Schema-level index
  - Define families of graph patterns
  - Assign instances to graph patterns
  - Map graph patterns to context (source URI)
Use case example: SchemEx (cont’d)

Idea:

▶ **Schema-level index**
  ▶ Define families of graph patterns
  ▶ Assign instances to graph patterns
  ▶ Map graph patterns to context (source URI)

▶ **Construction**
  ▶ Stream-based for scalability
  ▶ Little loss of accuracy
Use case example: SchemEx (cont’d)

Idea:
- Schema-level index
  - Define families of graph patterns
  - Assign instances to graph patterns
  - Map graph patterns to context (source URI)
- Construction
  - Stream-based for scalability
  - Little loss of accuracy
- Note
  - Index defined over instances
  - But stores the context
Use case example: SchemEx (cont’d)

- **n-Quads**

  `<subject> <predicate> <object> <context>`

- **Example:**

  `<http://www.w3.org/People/Connolly/#me>`
  `<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>`
  `<http://xmlns.com/foaf/0.1/Person>`
  `<http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>`

![Diagram of Linked Open Data]
Use case example: SchemEx (cont’d)

- Stream-based schema extraction
- While crawling the data
Use case example: SchemEx (cont’d)

- Stream of n-quads (coming from a LD crawler)

  ... Q16, Q15, Q14, Q13, Q12, Q11, Q10, Q9, Q8, Q7, Q6, Q5, Q4, Q3, Q2, Q1

- Linear runtime complexity wrt # of input triples
Use case example: SchemEx (cont’d)

- **RDF classes**
- **Type clusters**
- **Equivalence classes**
- **Data sources**

Diagram:
- **Classes**: C1, C2, C3, ..., Ck
- **Type clusters**: TC1, TC2, ..., TCm
- **Equivalence classes**: EQC1, EQC2, ..., EQCn
- **Data sources**: DS1, DS2, DS3, DS4, DS5, ..., DSx

Relationships:
- hasEQClass
- hasDataSource
- consistsOf
Layer 1: RDF Classes

- All instances of a particular type
Layer 2: Type Clusters

- All instances belonging to exactly the same set of types
Layer 3: Equivalence Classes

- Two instances are equivalent iff:
  - They are in the same TC
  - They have the same properties
  - The property targets are in the same TC

- Similar to 1-Bisimulation
Use case example: SchemEx (cont’d)

- Stream of n-quads (coming from a LD crawler)

- Linear runtime complexity wrt # of input triples
Summary

- Linked Open Data as the application area of RDF, RDFS, OWL
- Linked Open Data Scheme
  - Publish under an open license
  - Publish structured data
  - Use non-proprietary formats
  - Use URIs to identify things
  - Link your data to other data
- Examples: DBpedia, FOAF, Dublin Core
- Applications: SchemEx
Pointers to further reading

- 5stardata.info: The five stars of Linked Open Data Conformance
No exercises today
Have a nice weekend!