The Semantic Web

RDF, SPARQL and software APIs

4IZ440 Knowledge Representation and Reasoning on the WWW

Josef Petrák | me@jspetrak.name

Worth a note

- Open Data initiave around world and Europe open statistical and government data, usually in RDF format.
- It has several success stories
  - http://headtoweb.posterous.com/open-data-success-stories
- Google is integrating RDF encoded as RDFa in pages into its search results.

- UEP is not behind – DIKE research group KEG started Czech–Slovak semantic initiative called Semantics
- And we have a RDF-based website deployed, see http://keg.vse.cz/

Outline

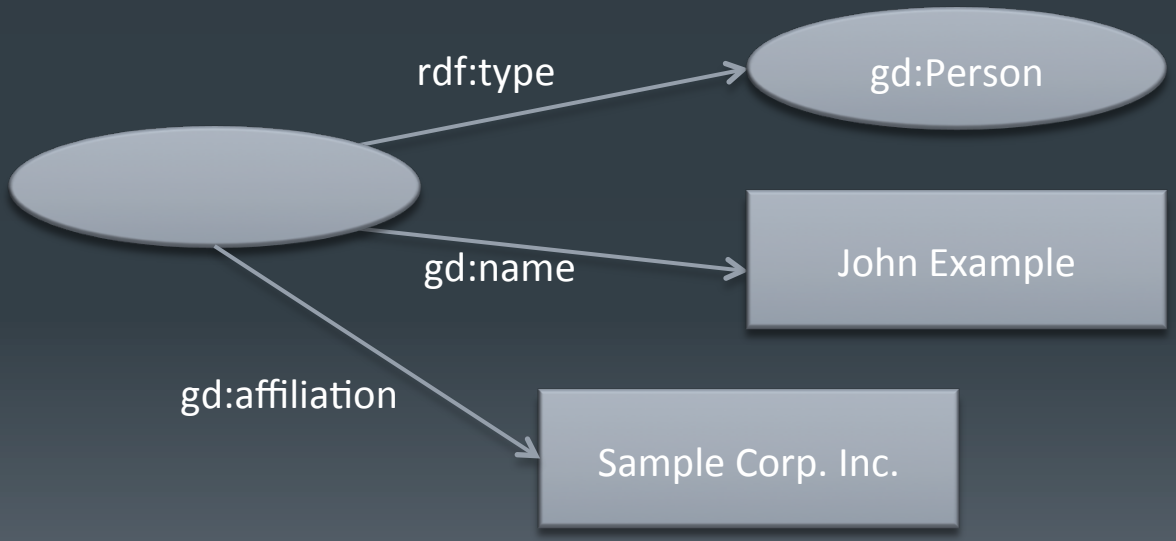
- RDF representation formats
- Data handling approaches
- Software APIs overview
- Approaches in examples
- Motivation example: web application

RDF representation

Syntaxes

- RDF has several syntaxes.
- A “graph” is the reference syntax.
- The W3C endorsed file format is RDF/XML
- Other file formats:
  - N-Triples
  - N3
  - RDF/JSON
  - RDFa

RDF graph




```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX gd: <http://rdf.data-vocabulary.org/#>
```

RDF/XML

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:gd="http://rdf.data-vocabulary.org/#">
  <rdf:Resource>
    <rdf:type rdf:resource="http://rdf.data-vocabulary.org/#Person"/>
    <gd:name>John Example</gd:name>
    <gd:affiliation>Sample Corp. Inc.</gd:affiliation>
  </rdf:Resource>
</rdf:RDF>
```

RDF/XML (shortened)

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:gd="http://rdf.data-vocabulary.org/#">
  <gd:Person>
    <gd:name>John Example</gd:name>
    <gd:affiliation>Sample Corp. Inc.</gd:affiliation>
  </gd:Person>
</rdf:RDF>
```

N3 Notation

```
@prefix gd: <http://rdf.data-vocabulary.org/#> .

[] a gd:Person;
  gd:name “John Example”;
  gd:affiliation “Sample Corp. Inc.” .
```

N–Triples

```
_:a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://rdf.data-vocabulary.org/#Person> .
_:a <http://rdf.data-vocabulary.org/#name>
        “John Example” .
_:a <http://rdf.data-vocabulary.org/#affiliation>
        “Sample Corp. Inc.” .
```

- Primarily used to defined RDF test cases
- Subset of N3 Notation

RDF/JSON

```
{
  “_:a“ : {
    “http://www.w3.org/1999/02/22-rdf-syntax-ns#” : [
      {
        “value“ : “http://rdf.data-vocabulary.org/#Person”, “type“ : “uri“
      }
    ],
    “http://rdf.data-vocabulary.org/#name” : [
      {
        “value“ : “John Example“, “type“ : “literal“
      }
    ],
    “http://rdf.data-vocabulary.org/#affiliation” : [
      {
        “value“ : “Sample Corp. Inc.“, “type“ : “literal“
      }
    ]
  }
}
```

RDFa

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:gd="http://rdf.data-vocabulary.org/#">

<p typeof="gd:Person">
  <em property="gd:name">John Example</em> is working
  at <em property="gd:affiliation">Sample Corp. Inc.</em>.
</p>

</html>
```

Resources about RDF formats

- RDF/XML
  - http://www.w3.org/TR/rdf-syntax-grammar/
- N3 Notation
  - http://www.w3.org/DesignIssues/Notation3
  - http://www.w3.org/2000/10/swap/Primer
- N–Triples
  - http://www.w3.org/TR/rdf-testcases/#ntriples
- RDF/JSON
  - http://n2.talis.com/wiki/RDF_JSON_Specification
- RDFa
  - http://www.w3.org/TR/rdfa-syntax/

Task #1

1. Use Google Data Vocabulary
   - http://www.data-vocabulary.org/Person/
2. Created a Notation3 file with demo data about people

Data handling

RDF models

- Statement–centric
  - Working with triples of ?subject ?predicate ?object
- Resource–centric
  - Working with resources having properties and their values
- Ontology–centric
  - Working with classes, properties, and individuals as defined in selected vocabulary/schema/ontology
- Named graph
  - Triples belongs to a graph with URI name
  - Working with quads of ?graph ?subject ?predicate ?object

Software APIs

Java

- Jena
  - http://openjena.org/
  - http://sourceforge.net/projects/jena/
- Sesame
  - http://www.openrdf.org/
- Shellac RDFa Parser
  - https://github.com/shellac/java-rdfa

PHP

- RDF API for PHP (RAP)
  - http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/
  - http://sourceforge.net/projects/rdfapi-php/
- ARC2
  - http://arc.semsol.org/

Ruby

- RDF.rb
  - http://rdf.rubyforge.org/
  - https://github.com/bendiken/rdf
  - Plus various modules

Code Examples

File-read

- Task: Read RDF data from a Notation3 file format into memory model.

Read N3 (Jena)

```
InputStream is =
  FileManager.get().open("samples/people.n3");

Model model = ModelFactory.createDefaultModel();
RDFReader r = model.getReader("N3");
r.read(model, is, null);
is.close();
```

Read N3 (RDF.rb)

```
repository = RDF::Repository.new

RDF::N3::Reader.open('samples/people.n3') { |reader|
  repository <<  reader
}
```

Read N3 (ARC2)

```
$parser = ARC2::getRDFParser();
$parser->parse(‘samples/people.n3');

$triples = $parser->getTriples();
```

Read N3 (RAP)

```
$model = ModelFactory::getDefaultModel();
$model->load('samples/people.n3');
```

- This code won't run, because there is N3Reader bug.
- Works also for RDF/XML and N–Triples formats.

Traverse triples

- Task: Take a memory model and list all or particular RDF triples.

Traverse triples (Jena)

```
Model model = ModelFactory.createDefaultModel();
// Load data into model

StmtIterator i = model.listStatements();
while (i.hasNext()) {
  Statement stmt = i.nextStatement();
  Resource subject = stmt.getSubject();
  Property predicate = stmt.getPredicate();
  RDFNode object = stmt.getObject();

  // Printing out
}
```

Query Model (Jena)

```
Resource rPerson = model.getResource("http://rdf.data-
vocabulary.org/#Person");
Property rName = model.getProperty("http://rdf.data-vocabulary.org/
#name");

StmtIterator si = model.listStatements(null, RDF.type, rPerson);
while (si.hasNext()) {
  Resource r = si.nextStatement().getSubject();
  StmtIterator sii = model.listStatements(r, rName, (RDFNode)null);
  if (sii.hasNext()) {
    System.out.println(sii.nextStatement().getObject().toString());
  }
}
```

Traverse triples (RDF.rb)

```
repository = RDF::Repository.new
# Load data

repository.each_statement { |statement|
  puts statement
}
```

Query model (RDF.rb)

```
GD = RDF::Vocabulary.new('http://rdf.data-vocabulary.org/#')

repository = RDF::Repository.new
# Load data

query = RDF::Query.new({
  :person => {
    RDF.type => GD.Person, GD.name => :name
  }
})

query.execute(graph).each do |person|
  puts "name=#{person.name}"
end
```

Query SPARQL Endpoint

- Task: Query a SPARQL endpoint and write out the data. E.g. from DBPedia, write one random blackboard gag written by Bart Simpson.

Query SPARQL Endpoint (Jena)

```
String qa = "PREFIX skos: <http://www.w3.org/2004/02/skos/core#>"
                + "PREFIX dbpprop: <http://dbpedia.org/property/>"
                + "PREFIX dcterms: <http://purl.org/dc/terms/>"
                + "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"
                + "SELECT ?gag WHERE { "
                + "?series skos:broader <http://dbpedia.org/resource/
Category:The_Simpsons_episodes> ."
                + "  ?episode dcterms:subject ?series ."
                + "  ?episode dbpprop:blackboard ?gag ."
                + "}"
                + "LIMIT 1000";
Query q = QueryFactory.create(qa);
QueryExecution qe =
  QueryExecutionFactory.sparqlService("http://dbpedia.org/sparql", q);
ResultSet rs = qe.execSelect();
ResultSetFormatter.out(System.out, rs, q);
qe.close();
```

Query SPARQL Endpoint (RDF.rb)

```
sparql =
  SPARQL::Client.new("http://dbpedia.org/sparql")

result = sparql.query("SELECT ?gag WHERE {
  ?series skos:broader <http://dbpedia.org/resource/
Category:The_Simpsons_episodes> .
  ?episode dcterms:subject ?series .
  ?episode dbpprop:blackboard ?gag .
}
LIMIT 1000")

puts result[rand(result.length)][:gag]
```

Query SPARQL Endpoint (RAP)

```
$client =
  ModelFactory::getSparqlClient("http://www.exampleSparqlService.net:2020/
example");

$query = new ClientQuery();
$query->query('PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
          PREFIX dbpprop: <http://dbpedia.org/property/>
          PREFIX dcterms: <http://purl.org/dc/terms/>
          PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
          SELECT ?gag WHERE {
            ?series skos:broader <http://dbpedia.org/resource/
Category:The_Simpsons_episodes> .
            ?episode dcterms:subject ?series .
            ?episode dbpprop:blackboard ?gag .
          } LIMIT 1000');
$result = $client->query($query);

foreach($result as $line){ echo($line['?gag']->toString()); }
```

Query SPARQL Endpoint (ARC2)

```
$store =
  ARC2::getRemoteStore(
    array('remote_store_endpoint' => 'http://dbpedia.org/sparql'));

$q = ‘PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
         PREFIX dbpprop: <http://dbpedia.org/property/>
         PREFIX dcterms: <http://purl.org/dc/terms/>
         PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
         SELECT ?gag WHERE {
           ?series skos:broader <http://dbpedia.org/resource/
Category:The_Simpsons_episodes> .
           ?episode dcterms:subject ?series .
           ?episode dbpprop:blackboard ?gag .
         } LIMIT 1000’;

$rows = $store->query($q, 'rows');
```

Task #2

1. Select one of frameworks (preferably Jena)
2. Load data from your Notation3 file
3. Write out names and work titles of people

Motivation Example

Knowledge Engineering Group Website

- http://keg.vse.cz/
- RDF outside
  - Data dumps in RDF/XML
  - Web pages enriched with RDFa
- RDF inside
  - Data created by SPARQL `INSERT`
  - Data queried by SPARQL `SELECT`
  - Data updated by SPARQL `DELETE` and `SELECT`
  - Data manipulated by user–friendly forms
- Ongoing: data integration with ISIS VŠE and other department applications

SPARQL: Create data

```
INSERT INTO <http://keg.vse.cz/> {
  <http://keg.vse.cz/resource/event/1> rdf:type ical:Vevent .
  <http://keg.vse.cz/resource/event/1> ical:uid ”1" .
  <http://keg.vse.cz/resource/event/1> ical:summary ”IZI440" .
  <http://keg.vse.cz/resource/event/1> ical:dtstart ”2011-02-28" .
}
```

SPARQL: Read data

```
SELECT ?summary ?uid ?dtstart ?dtend
WHERE {
  <http://keg.vse.cz/resource/event/1> rdf:type ical:Vevent .
  <http://keg.vse.cz/resource/event/1> ical:uid ?uid .
  <http://keg.vse.cz/resource/event/1> ical:summary ?summary .
  <http://keg.vse.cz/resource/event/1> ical:dtstart ?dtstart .
  OPTIONAL {
    <http://keg.vse.cz/resource/event/1> ical:dtend ?dtend .
  }
}
```

SPARQL: Update data

```
DELETE {
  <http://keg.vse.cz/resource/event/1> ?p ?o .
}

INSERT INTO <http://keg.vse.cz/> {
  <http://keg.vse.cz/resource/event/1> rdf:type ical:Vevent .
  <http://keg.vse.cz/resource/event/1> ical:uid ”1" .
  <http://keg.vse.cz/resource/event/1> ical:summary ”4IZ440" .
  <http://keg.vse.cz/resource/event/1> ical:dtstart ”2011-02-28" .
}
```

References

- http://dsic.zapisky.info/RDF/FOAF/parsingWithPHP/
- http://zapisky.info/?item=zverejnime-akademicke-projekty-samozrejme-semanticky

- **BOOK** – John Hebeler (Author), Matthew Fisher (Author), Ryan Blace (Author), Andrew Perez-Lopez (Author), Mike Dean (Foreword): *Semantic Web Programming*, Wiley, 2009

Questions?

Thank you