Serving and Querying Open Knowledge Graphs on the Web - Basics

Axel Polleres
What I've planned for today:

- **Basics:**
  - Interlude – some words on syntax...
  - Practical SPARQL on examples querying Open KGs with SPARQL
  - Challenges/limitations of SPARQL over public endpoints

- **Bonus Material (time allowed or upon request):**
  - Serve and query KGs for local processing – HDT
  - Addressing the SPARQL endpoint bottleneck – where are we?
    - Linked Data Fragments
    - Smart-KG
    - Wise-KG
Standard format (RDF) & Standard Query language (SPARQL) for Graph Data

- Data representation
  - RDF ( = Resource Description Framework)
    - a standard Format for publishing Graph Data on the Web.
    - Can be seen as a labeled graph

- Querying
  - SPARQL
    - a query language (similar to SQL) for RDF data

We'll have some riddles for the students at the school.

Do you need a nasty SPARQL query? :)

the SPARQL query abilities could have very different levels among students.

Let's fix that in the tutorial! :)

the SPARQL query abilities could have very different levels among students.

Let's fix that in the tutorial! :)
RDF... we need to talk about syntax!

- We already mentioned... triple of URLs

\[
\langle \text{http://www.polleres.net\#me} \rangle \langle \text{http://xmlns.com/foaf/0.1/workplaceHomepage} \rangle \langle \text{http://www.wu.ac.at} \rangle.
\]

- ... can be seen as an edge in a Graph:
Vocabularies (collections of URIs to define meaning for Links) are identified by a common URI prefix:

The

**RDF Core** (rdf: [http://www.w3.org/1999/02/22-rdf-syntax-ns#](http://www.w3.org/1999/02/22-rdf-syntax-ns#)) and

**RDFS Schema** (rdfs: [http://www.w3.org/2000/01/rdf-schema#](http://www.w3.org/2000/01/rdf-schema#))

vocabularies define basic meaning for relations such as is-A, subclasses/subproperties, (human-readable) labels, etc. according to the **RDF specification**:

- Important URIs that used for links (in many KGs):
  - [http://www.w3.org/1999/02/22-rdf-syntax-ns#type](http://www.w3.org/1999/02/22-rdf-syntax-ns#type) (or short rdf:type)
  - [http://www.w3.org/2000/01/rdf-schema#label](http://www.w3.org/2000/01/rdf-schema#label) (or short rdfs:label)
  - [http://www.w3.org/2000/01/rdf-schema#subPropertyOf](http://www.w3.org/2000/01/rdf-schema#subPropertyOf) (or short rdfs:subPropertyOf)
  - [http://www.w3.org/2000/01/rdf-schema#subClassOf](http://www.w3.org/2000/01/rdf-schema#subClassOf) (or short rdfs:subClassOf)
  - [http://www.w3.org/2000/01/rdf-schema#domain](http://www.w3.org/2000/01/rdf-schema#domain) (or short rdfs:domain)
  - [http://www.w3.org/2000/01/rdf-schema#range](http://www.w3.org/2000/01/rdf-schema#range) (or short rdfs:range)
Other vocabularies:
- **foaf**: Prefix: [http://xmlns.com/foaf/0.1/](http://xmlns.com/foaf/0.1/) ... The "Friend-of-a-friend' vocabulary models common properties of and classes relating to Persons and social relationships. E.g.:
  - **Properties:**
    - name
    - nickname
    - workplaceHomepage
    - knows
  - **Classes:**
    - Agent
    - Person
    - Document
    - Image

- **schema**: Prefix: [http://schema.org/](http://schema.org/) ...
  - Classes and properties important for search engines
  - (found by Google, Microsoft, Yahoo and Yandex)
  - or domain/KG-specific vocabularies, eg.
  - **dbo**: (DBpedia Ontology)
  - **wd:, wdt:** (Wikidata entities and properties)
RDF Syntaxes – A simple RDF file:

simple1.nt in NTriples Syntax:

```xml
```
simple1.ttl in Turtle (Terse RDF Language) Syntax is a bit more readable:

# using the FOAF vocabulary, see http://xmlns.com/foaf/spec/

@prefix : <http://www.example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

:klaus foaf:knows :karl .
:klaus foaf:nickname "Niki".
:alice foaf:name "Alice Wonderland" .
:karl foaf:name "Karl Mustermann" .
:karl foaf:knows :joan .
:bob foaf:name "Robert Mustermann" .
:bob foaf:nickname "Bobby" .
RDF Syntaxes – A simple RDF file:

simple1.ttl in Turtle (Terse RDF Language) Syntax is a bit more readable –

Turtle Syntax also allows some shortcuts to group Triples with common subjects:

# using the FOAF vocabulary, see http://xmlns.com/foaf/spec/

@prefix : <http://www.example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

:klaus foaf:knows :karl ;
   foaf:nickname "Niki".
:karl foaf:name "Karl Mustermann" ; foaf:knows :joan.
:bob foaf:name "Robert Mustermann" ; foaf:nickname "Bobby" .

Note: We will need Turtle Syntax for querying RDF data!
Standards like RDF have lead to (really) big open KGs...

- ... some of which available on the Web
- ... *queryable via SPARQL endpoints*!

plus useful convenience tools:
- [http://prefix.cc/](http://prefix.cc/) ... find out common URI prefixes for formulating queries
- [http://yasgui.triply.cc/](http://yasgui.triply.cc/) ... really nice frontend for querying SPARQL endpoints, e.g. DBpedia
- [https://query.wikidata.org/](https://query.wikidata.org/) ... really nice frontend specifically for querying Wikidata

1,101,215,718 triples/edges 13,602,048,837 triples/edges

Plus tons of APIs (e.g. Python, R packages, etc.)
RDF used in practice on the Web:
DBpedia - a "Database-version" of Wikipedia:

- E.g. from DBpedia

- One of the central datasets of the Linked Open Data-Cloud
- RDF extracted from Wikipedia-Infoboxes
- You can use a language called SPARQL (roughly: SQL for RDF) to do **structured queries** over RDF via Web accessible **SPARQL endpoints**, e.g. [http://dbpedia.org/sparql](http://dbpedia.org/sparql)
  - "Cities in the UK with more than 1M population":

```sparql
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX yago: <http://dbpedia.org/class/yago/>

SELECT DISTINCT ?city ?pop WHERE {
  ?city a yago:City108524735 .
  ?city dbo:populationTotal ?pop
  FILTER ( ?pop > 1000000 )
}
```

Structured queries (SPARQL):

Try it on [yasgui.triply.cc](http://yasgui.triply.cc) ... short link to the query: [https://api.triplydb.com/s/Of19_c3-e](https://api.triplydb.com/s/Of19_c3-e)
RDF used in practice on the Web: Another Open Knowledge Graph: Wikidata

- Slightly different idea than DBpedia:
  - a Wikimedia foundation project itself
  - put simply: "replace factual data within Wikipedia by a (graph) Database"

- Wikidata can also be queried as RDF with SPARQL!
Let's learn some SPARQL with Wikidata

- “Simple” surface query:

  Which cities in the UK have more than 1M people?

  ```sparql
  SELECT DISTINCT ?city WHERE {
    FILTER (?population > 1000000) }
  ```

- What’s this?
Let's learn some SPARQL with Wikidata

- You can try out the queries on [http://query.wikidata.org/](http://query.wikidata.org/)

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) ... Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) ... instanceof

**Triple Patterns (TPs):** Try this query for

"Give me 10 buildings"

https://w.wiki/4TAP

```sparql
1 SELECT *
2 WHERE {
3   ?X wdt:P31 wd:Q41176
4   }
5 LIMIT 10
```
Let's learn some SPARQL with Wikidata

- You can try out the queries on [http://query.wikidata.org/](http://query.wikidata.org/)

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) ... Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) ... instanceof

**Basic Graph patterns (BGPs):** "Join" between edges/triples:

"Give me 10 buildings in Austria"

```sparql
1 SELECT *
2 WHERE {
5 }
6 LIMIT 10
```
Let's learn some SPARQL with wikidata

- You can try out the queries on http://query.wikidata.org/

https://www.wikidata.org/entity/Q41176 (wd:Q41176) ... Building
http://www.wikidata.org/prop/direct/P31 (wdt:P31) ... instanceOf

**UNION** between patterns:

"Give me 10 buildings in **Austria or Germany**"

```sparql
1 SELECT *
2 WHERE {
4   { {?X wdt:P17 wd:Q40 . } UNION
5     {?X wdt:P17 wd:Q183 . } }
6 }
7 LIMIT 10
```
Let's learn some SPARQL with wikidata

- You can try out the queries on [http://query.wikidata.org/](http://query.wikidata.org/)

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) ... Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) ... instanceOf

**FILTERs** (similar to WHERE conditions in SQL):

"Give me the **German labels** of 10 buildings in Austria or Germany"

https://w.wiki/4TAk

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SELECT ?L</td>
</tr>
<tr>
<td>2</td>
<td>WHERE {</td>
</tr>
<tr>
<td>3</td>
<td>?X wdt:P31 wd:Q41176 ;</td>
</tr>
<tr>
<td>4</td>
<td>rdfs:label ?L .</td>
</tr>
<tr>
<td>5</td>
<td>{ {?X wdt:P17 wd:Q40 .} UNION {?X wdt:P17 wd:Q183 .} }</td>
</tr>
<tr>
<td>6</td>
<td>FILTER (lang(?L) = &quot;en&quot;)</td>
</tr>
<tr>
<td>7</td>
<td>}</td>
</tr>
<tr>
<td>8</td>
<td>LIMIT 10</td>
</tr>
</tbody>
</table>
Let's learn some SPARQL with wikidata

- You can try out the queries on [http://query.wikidata.org/](http://query.wikidata.org/)

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) … Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) … instanceOf

**OPTIONAL** (similar to OUTER JOIN in SQL):

"Give me the German labels of 10 buildings in Austria and their architect (if available)"

https://w.wiki/4TAn

```
SELECT ?L ?A
WHERE {
  ?X wdt:P31 wd:Q41176 ;
  rdfs:label ?L ;
  wdt:P17 wd:Q40 .
  FILTER (lang(?L) ="en")
  OPTIONAL {?X wdt:P84 ?A }
  }
LIMIT 10
```
Full details of SPARQL and many more examples:

- [https://www.w3.org/TR/sparql11-query/](https://www.w3.org/TR/sparql11-query/)

- Supported by various modern graph databases.
What I've planned for today:

- **Basics:**
  - Interlude – some words on syntax...
  - Practical SPARQL on examples querying Open KGs with SPARQL
  - **Challenges/limitations of SPARQL over public endpoints**

- **Bonus Material (time allowed or upon request):**
  - Serve and query KGs for local processing – HDT
  - Addressing the SPARQL endpoint bottleneck – where are we?
    - Linked Data Fragments
    - Smart-KG
    - Wise-KG
(Some) Challenges:

1. **Challenge 1**: How to query Contextualized Data (e.g. temporal, provenance, …)
2. **Challenge 2**: What about real graph queries (paths, paths across distributed data)?
3. **Challenge 3**: Scalability (and costs of hosting) SPARQL endpoints
4. **Challenge 4**: Mixing querying and reasoning (how? how to scale?)
5. **Challenge 5**: Sustainability of RDF and SPARQL resources
Challenge 1: Often, you also need to deal with **contextualized** information

- E.g. from

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX yago: <http://dbpedia.org/class/yago/>

SELECT DISTINCT ?city ?pop WHERE {
  ?city a yago:City108524735 .
  ?city dbo:country :Italy .
  FILTER ( ?pop > 1000000 )
}
```

```
http://yasgui.org/short/UVOyhX8ft
```

""Cities in the **Italy** with more than 1M population:""
Challenge 1: Wikidata as RDF ... In Wikidata even context information can be queried by SPARQL

- However, Wikidata has more complex info: (temporal context, provenance, ...)
  - Rome:
  - https://www.wikidata.org/wiki/Q220

... Can I query that with SPARQL? Yes!

```
SELECT ?city (min(?time) as ?year) WHERE {
  FILTER (?population > 1000000 )
} GROUP BY ?city
```

https://w.wiki/4rs
Challenge 1: Contextualized information in RDF

- no standard as of yet. State of affairs:
  - Wikidata has its own proprietary extension (cf. last slide)
  - Alternative representations/engines involve Property Graphs
  - ongoing work: RDF*/SPARQL* community group

**Rome** capitalOf **Italy**

**Q220**
Name: Rome  
Type: City  
added: 2019-06-11  
author: @Alice

**Q5119**
name: Italy  
type: capitalOf  
added: 2019-06-12  
author: @Bob

**Q38**
validityrFrom: 1861
Challenge 2: Path queries

- While it is possible to do path queries in SPARQL via property path expressions, it is still not possible to return paths in SPARQL1.1:

i.e.: what is the (shortest) path ?Path connecting ?city and wd:Q515?

```sparql
SELECT DISTINCT ?city ?Path WHERE {
  FILTER (?population > 1000000) }
```
Challenge 2: Path queries – prototype solution

Common problem in graphs, not doable with SPARQL, but with extensions [1]: “Give me the (k) shortest paths between two nodes?”

```
:b :p :d, :e.
:c :p :b, :d.
:d :p :e.
:e :p :c.
```

We solved this by extending SPARQL [1] with bidirectional BFS over HDT

```
rdf2hdt.sh -rdftype turtle testgraph.ttl testgraph.hdt
```

```
hdtsparql.sh testgraph.hdt "PREFIX ppf: <java:at.ac.wu.arqext.path.> SELECT * WHERE{ ?path ppf:topk (:a :d 2) }"
```

Open research question(s): e.g. But how to do this effectively in a **Federated** setting?

k=2

Still interesting question also in (Graph)DB Theory... regarding entailments, coverage of queries in such settings.

[Savenkov et al, SEMANTiCS 2017]
Challenge 3: Scalability of SPARQL endpoints?

Challenge 3.1: serve complex/long running queries to single users

Example:
"Classes with their number of instances"

Challenge 3.2: serve many queries to many users concurrently

https://w.wiki/4mTj

[Fernández et al. 2013, JWS][Vergbourgh et al. 2016, JWS]

Observation 2:
- Serving SPARQL endpoints sustainably is too hard/expensive?

→ Linked Data has rather evolved into a set of few, but huge, popular (Open) Knowledge Graphs:
Challenge 1: serve complex/long running queries to single users

Challenge 2: serve many queries to many users concurrently

https://iccl.inf.tu-dresden.de/web/Wikidata_SPARQL_Logs/

<table>
<thead>
<tr>
<th>Interval</th>
<th>First day</th>
<th>Last day</th>
<th>Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval 1</td>
<td>2017-06-12</td>
<td>2017-07-09</td>
<td>59,547,909</td>
</tr>
</tbody>
</table>

[Fernández et al. 2013, JWS][Vergbourgh et al. 2016, JWS]

A lot of work has been done in the past on (deductive reasoning over KGs) in particular to retrieve implicit answers through exploiting the OWL and RDFS semantics.

... e.g. by query rewriting or materialisation.

However:
1) existing KGs are inconsistent
2) some important KGs don't use OWL and RDFS
Challenge 4: Reasoning and Inconsistencies
Existing KGs aren’t consistent 😞 [1]

- E.g. DBpedia

Dbpedia Ontology:

dbo:Agent owl:disjointWith dbo:Place.

dbo:Country rdfs:subClassOf dbo:Place.

dbo:Organisation rdfs:subClassOf dbo:Agent.

[Bischof et al. 2014]
Challenge 4: Reasoning and Inconsistencies

important KGs don't use OWL and RDFS

- Wikidata!

```
SELECT DISTINCT ?city ?Path WHERE { 
  FILTER (?population > 1000000) }
```

use "somewhat similar" properties:
wdt:P31 ~ rdf:type
wdt:P279 ~ rdfs:subClassOf
Challenge 4: Reasoning and Inconsistencies

Prefixes can carry semantics (not only owl: and rdfs:) ...

- Wikidata metamodel "by prefixes":

```
wd  http://www.wikidata.org/entity/
wdt http://www.wikidata.org/prop/direct/
p   http://www.wikidata.org/prop/

wds http://www.wikidata.org/entity/statement/
pq  http://www.wikidata.org/prop/qualifier/
ps  http://www.wikidata.org/prop/statement/
```

[Haller et al., ESWC 2022]
Challenge 5: Sustainability of RDF and OWL resources on the Web...

- Vocabularies? FOAF:

- Linked Open Data? [Polleres et al.2020, SWJ]:

  "Among the mentioned 5435 resources in the 1281 "LOD"-tagged datasets on datahub.io, there are only 1917 resources URLs that could be dereferenced."
What I've planned for today:

- **Basics:**
  - Interlude – some words on syntax...
  - Practical SPARQL on examples querying Open KGs with SPARQL
  - Challenges/limitations of SPARQL over public endpoints

- **Bonus Material (time allowed or upon request):**
  - Serve and query KGs for local processing – HDT
  - Addressing the SPARQL endpoint bottleneck – where are we?
    - Linked Data Fragments
    - Smart-KG
    - Wise-KG


6. Ruben Verborgh, Miel Vander Sande, Pieter Colpaert, Sam Coppens, Erik Mannens, Rik Van de Walle: Web-Scale Querying through Linked Data Fragments. LDOW 2014


9. S. Clearly-Strange, Is Happening At ISWS. (2022), check what you can find about http://dbpedia.org/resource/Bertinoro https://w3id.org/framester/isws2022_th.owl#hauntedBy some entity, at this SPARQL endpoint: http://etna.istc.cnr.it/framester2/sparql and get all the info you are able to find about it!

