Medical Informatics

Lecture 11: SPARQL Querying

Dr Areti Manataki

Nanjing Medical University

In the previous lecture

- Turtle serialisation

```
@prefix  dbpedia:  <http://dbpedia.org/resource/> .
@prefix  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  :  <http://usher.ed.ac.uk/medinf/vocab/> .

dbpedia:Leonardo_DiCaprio rdf:type :Actor ;
                          :playedIn dbpedia:The_Wolf_of_Wall_Street ,
                                    dbpedia:Inception .
```

- A short introduction to RDFS

```
@prefix  terms:  <http://usher.ed.ac.uk/medinf/vocab/> .
@prefix  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

terms:Giraffe  rdf:type  rdfs:Class .
terms:Herbivore  rdf:type  rdfs:Class .
terms:Giraffe  rdfs:subClassOf  terms:Herbivore .
```

In this lecture

- Triple & graph patterns
- SPARQL query structure
- Using OPTIONAL & solution modifiers
- Querying multiple sources

SPARQL

- SPARQL (SPARQL Protocol And RDF Query Language) is the standard way to access RDF data.
- The SPARQL query language works closely with the structure of RDF, making use of its graph model.
- It shares many features with SQL.
- It is an official W3C Recommendation
  - SPARQL 1.0 in 2008
  - SPARQL 1.1 in 2013
- Further info at https://www.w3.org/TR/sparql11-query/

Triple pattern

- Triple pattern: `subject predicate object .`
- Variables included for any of the subject, predicate or object.
  - `?actor   uv:playedIn   :Giant .`
- Triple patterns help us select triples from a given RDF graph.

Triple pattern

```
:JamesDean uv:bornIn ?city .
```

Data

```
:JamesDean uv:bornIn  :MarionIndiana .
:JamesDean uv:playedIn :Giant .
:JamesDean uv:playedIn :EastOfEden .
:JamesDean uv:playedIn :RebelWithoutaCause .
```

Triple pattern

- Triple pattern: `subject predicate object .`
- Variables included for any of the subject, predicate or object.
  - `?actor   uv:playedIn   :Giant .`
- Triple patterns help us select triples from a given RDF graph.

Triple pattern

```
:JamesDean uv:playedIn ?film .
```

Data

```
:JamesDean uv:bornIn  :MarionIndiana .
:JamesDean uv:playedIn :Giant .
:JamesDean uv:playedIn :EastOfEden .
:JamesDean uv:playedIn :RebelWithoutaCause .
```

Graph patterns

- Graph pattern: a collection of triple patterns, enclosed in { }

Graph pattern

```
{   :JamesDean uv:playedIn ?film .
    ?film uv:directedBy ?director .   }
```

Data

```
:JamesDean uv:playedIn :Giant .
:JamesDean uv:playedIn :EastOfEden .
:JamesDean uv:playedIn :RebelWithoutaCause .
:Giant uv:directedBy :GeorgeStevens   .
:EastOfEden uv:directedBy :EliaKazan   .
```

SPARQL Query Structure

```
# list of prefixes
PREFIX  pref:  <URI>
...
# result description
SELECT...
# graph to search
FROM ...
# query pattern
WHERE { ... }
# query modifiers
ORDER BY...
```

A simple example

SPARQL query

```
PREFIX : <http://usher.ed.ac.uk/medinf/resource>
PREFIX uv: <http://usher.ed.ac.uk/medinf/vocab/>
SELECT ?film
WHERE {:JamesDean   uv:playedIn   ?film . }
```

Data

```
:JamesDean uv:bornIn  :MarionIndiana .
:JamesDean uv:playedIn :Giant .
:JamesDean uv:playedIn :EastOfEden .
:JamesDean uv:playedIn :RebelWithoutaCause .
:ElizabethTaylor uv:playedIn :Giant .
:Giant uv:directedBy :GeorgeStevens  .
:EastOfEden uv:directedBy :EliaKazan  .
```

Results

?film
:Giant
:EastOfEden
:RebelWithoutaCause

Another example

SPARQL query

```
PREFIX : <http://usher.ed.ac.uk/medinf/resource>
PREFIX uv: <http://usher.ed.ac.uk/medinf/vocab/>
SELECT ?film  ?director
WHERE {:JamesDean   uv:playedIn   ?film .
       ?film   uv:directedBy   ?director .}
```

Results

?film	?director
:Giant	:GeorgeStevens
:EastOfEden	:EliaKazan

Data

```
:JamesDean uv:bornIn  :MarionIndiana .
:JamesDean uv:playedIn :Giant .
:JamesDean uv:playedIn :EastOfEden .
:JamesDean uv:playedIn :RebelWithoutaCause .
:ElizabethTaylor uv:playedIn :Giant .
:Giant uv:directedBy :GeorgeStevens  .
:EastOfEden uv:directedBy :EliaKazan  .
```

Using OPTIONAL

SPARQL query

```
PREFIX : <http://usher.ed.ac.uk/medinf/resource>
PREFIX uv: <http://usher.ed.ac.uk/medinf/vocab/>
SELECT ?film  ?director
WHERE {:JamesDean   uv:playedIn   ?film .
       OPTIONAL  { ?film  uv:directedBy  ?director . }
      }
```

Data

```
:JamesDean uv:bornIn  :MarionIndiana .
:JamesDean uv:playedIn :Giant .
:JamesDean uv:playedIn :EastOfEden .
:JamesDean uv:playedIn :RebelWithoutaCause .
:ElizabethTaylor uv:playedIn :Giant .
:Giant uv:directedBy :GeorgeStevens  .
:EastOfEden uv:directedBy :EliaKazan  .
```

Results

?film	?director
:Giant	:GeorgeStevens
:EastOfEden	:EliaKazan
:RebelWithoutaCause	

Using solution modifiers

- Query patterns generate an unordered collection of solutions.
- In order to reorganise the solutions, we use solution sequence modifiers.
- Commonly used solution sequence modifiers:
  - DISTINCT: ensures solutions in the sequence are unique
  - ORDER BY: puts the solutions in order
  - LIMIT: restricts the number of solutions
  - OFFSET: controls where the solutions start from in the overall sequence of solutions

Using solution modifiers: DINSTINCT

SPARQL query

```
PREFIX : <http://usher.ed.ac.uk/medinf/resource>
PREFIX uv: <http://usher.ed.ac.uk/medinf/vocab/>
SELECT ?actor
WHERE {?actor   uv:playedIn   ?film . }
```

Data

```
:JamesDean uv:bornIn  :MarionIndiana .
:JamesDean uv:playedIn :Giant .
:JamesDean uv:playedIn :EastOfEden .
:JamesDean uv:playedIn :RebelWithoutaCause .
:ElizabethTaylor uv:playedIn :Giant .
:Giant uv:directedBy :GeorgeStevens  .
:EastOfEden uv:directedBy :EliaKazan  .
```

Results

?actor
:JamesDean
:JamesDean
:JamesDean
:JamesDean
:ElizabethTaylor

Using solution modifiers: DINSTINCT

SPARQL query

```
PREFIX : <http://usher.ed.ac.uk/medinf/resource>
PREFIX uv: <http://usher.ed.ac.uk/medinf/vocab/>
SELECT DISTINCT ?actor
WHERE {?actor   uv:playedIn   ?film . }
```

Data

```
:JamesDean uv:bornIn  :MarionIndiana .
:JamesDean uv:playedIn :Giant .
:JamesDean uv:playedIn :EastOfEden .
:JamesDean uv:playedIn :RebelWithoutaCause .
:ElizabethTaylor uv:playedIn :Giant .
:Giant uv:directedBy :GeorgeStevens  .
:EastOfEden uv:directedBy :EliaKazan  .
```

Results

?actor
:JamesDean
:ElizabethTaylor

FROM-part

- By using the FROM clause you can specify against which RDF dataset(s) your SPARQL query will be run.
- This is typically a URI that is addressable via HTTP when the query is executed.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?o
FROM <https://www.w3.org/People/Berners-Lee/card.rdf>
WHERE { ?s foaf:family_name ?o }
```

FROM-part

- By using the FROM clause you can specify against which RDF dataset(s) your SPARQL query will be run.
- This is typically a URI that is addressable via HTTP when the query is executed.
- Multiple data sources can be used.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?o
FROM <https://www.w3.org/People/Berners-Lee/card.rdf>
FROM <https://aic.ai.wu.ac.at/~polleres/foaf.rdf>
WHERE { ?s foaf:family_name ?o }
```

Conclusions

- Query structure
- Simple SPARQL queries

SPARQL query

```
PREFIX : <http://usher.ed.ac.uk/medinf/resource>
PREFIX uv: <http://usher.ed.ac.uk/medinf/vocab/>
SELECT ?film
WHERE {:JamesDean   uv:playedIn   ?film . }
```

- Using OPTIONAL & solution modifiers
- Querying multiple sources
- In the next lecture we'll discuss the principles of Linked Data.

Acknowledgements