Question Answering and Chatbots

3rd Practical exercise – SPARQL queries over Knowledge Graphs

Aleksandr Perevalov

aleksandr.perevalov@hs-anhalt.de

October 31, 2021

Hochschule Anhalt
Anhalt University of Applied Sciences

Knowledge Graphs (KGs)

Knowledge Graphs (KGs)

- are databases

Knowledge Graphs (KGs)

- are databases
- store the data in the graph structure

Knowledge Graphs (KGs)

- are databases
- store the data in the graph structure
- contain reasoning rules

Knowledge Graphs (KGs)

- are databases
- store the data in the graph structure
- contain reasoning rules
- support interlinking with other KGs

Knowledge Graphs (KGs)

- are databases
- store the data in the graph structure
- contain reasoning rules
- support interlinking with other KGs
- store a lot of contextual information

Knowledge Graph – Simple Example

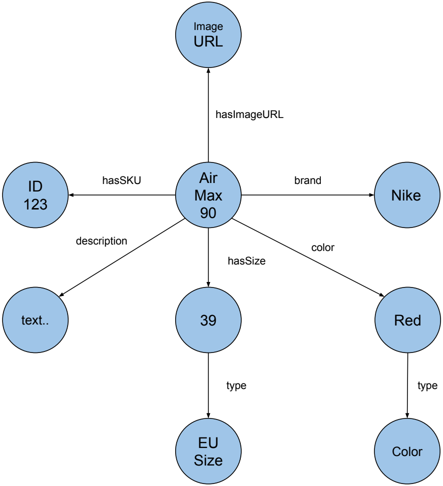


Figure: Online Store KG, can be implemented with a Relational DB

Knowledge Graph – Simple Example + Interlinking

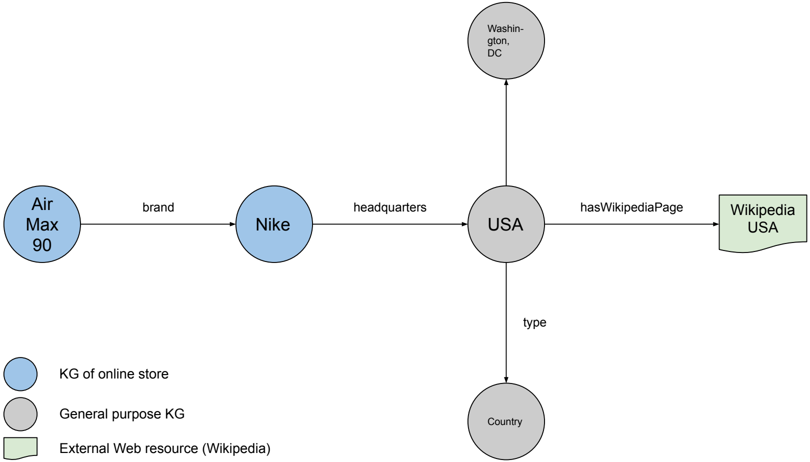
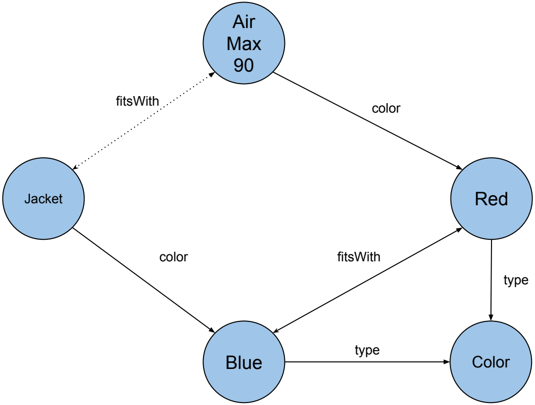


Figure: Online Store KG, interlinked with external Web sources

Knowledge Graph – Simple Example + Reasoning



```
RULE fitsWithRule

IF {
   ?productA <color> ?colorA .
   ?productB <color> ?colorB .
   ?colorB <fitsWith> ?colorA .
}
THEN {
   ?productA <fitsWith> ?productB .
}
```

Figure: Online Store KG with a reasoning rule

Knowledge Graph – Examples

Knowledge Graph – Examples

Knowledge Graph – Examples

Knowledge Graph – Examples

Knowledge Graph – Examples

Freebase™

DBpedia

WIKIDATA

Knowledge Graph – Resource Description Framework

Knowledge Graph – Resource Description Framework

RDF – Resource Description Framework. (A notation for storing data model of knowledge graphs).

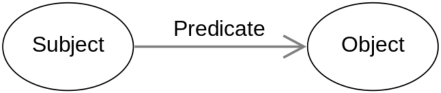


Figure: Basic RDF graph – A Triple: Subject-Predicate(Relation)-Object

Knowledge Graph – Resource Description Framework

RDF – Resource Description Framework. (A notation for storing data model of knowledge graphs).

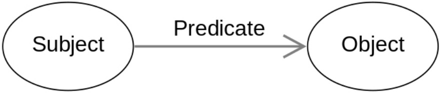


Figure: Basic RDF graph – A Triple: Subject-Predicate(Relation)-Object

RDF can be serialized to XML (and many other formats). However, it is easier to store it in human-friendly format **TTL or Turtle**. The database for RDF is called a **Triplestore**.

RDF on the Web

HTML to RDF converter: https://www.w3.org/2012/sde/

RDF on the Web

HTML to RDF converter: https://www.w3.org/2012/sde/

- Online Clothing Store

RDF on the Web

HTML to RDF converter: https://www.w3.org/2012/sde/

- Online Clothing Store
- Articles on the Web

RDF on the Web

HTML to RDF converter: https://www.w3.org/2012/sde/

- Online Clothing Store
- Articles on the Web
- Youtube

What is SPARQL

SPARQL – is a query language for data stored in RDF format.

Sources: https://www.w3.org/TR/rdf-sparql-query/ ,
https://www.w3.org/TR/sparql11-update/

What is SPARQL

SPARQL – is a query language for data stored in RDF format.
Some of the forms of SPARQL queries:

Sources: https://www.w3.org/TR/rdf-sparql-query/ ,
https://www.w3.org/TR/sparql11-update/

What is SPARQL

SPARQL – is a query language for data stored in RDF format.

Some of the forms of SPARQL queries:

- SELECT – returns variables and their bindings according to a query.

Sources: https://www.w3.org/TR/rdf-sparql-query/ , https://www.w3.org/TR/sparql11-update/

What is SPARQL

SPARQL – is a query language for data stored in RDF format.
Some of the forms of SPARQL queries:

- SELECT – returns variables and their bindings according to a query.
- ASK – test whether or not a query pattern has a solution (bool).

Sources: https://www.w3.org/TR/rdf-sparql-query/ ,
https://www.w3.org/TR/sparql11-update/

What is SPARQL

SPARQL – is a query language for data stored in RDF format.
Some of the forms of SPARQL queries:

- SELECT – returns variables and their bindings according to a query.
- ASK – test whether or not a query pattern has a solution (bool).
- INSERT – adds triples, given inline in the query.

Sources: https://www.w3.org/TR/rdf-sparql-query/ ,
https://www.w3.org/TR/sparql11-update/

What is SPARQL

SPARQL – is a query language for data stored in RDF format.
Some of the forms of SPARQL queries:

- SELECT – returns variables and their bindings according to a query.
- ASK – test whether or not a query pattern has a solution (bool).
- INSERT – adds triples, given inline in the query.
- DESCRIBE – "describes" the resolved resources in a query.

Sources: https://www.w3.org/TR/rdf-sparql-query/ ,
https://www.w3.org/TR/sparql11-update/

SELECT query over DBpedia

Question: “Name a person.”

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?uri WHERE {
  ?uri rdf:type dbo:Person . # resource of type Person
}
LIMIT 1 # only one result will be given
```

SELECT query over DBpedia

Question: “Name a person born in Brooklyn.”

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?uri WHERE {
  ?uri rdf:type dbo:Person . # resource of type Person
  ?uri dbo:birthPlace dbr:Brooklyn . # with birth place Brooklyn
}
LIMIT 1 # only one result will be given
```

SELECT query over DBpedia

Question: “Name a person born in Brooklyn after 1980.”

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?uri WHERE {
  ?uri rdf:type dbo:Person . # resource of type Person
  ?uri dbo:birthPlace dbr:Brooklyn . # with birth place Brooklyn
  ?uri dbp:birthDate ?birthDate . # get a birth date
  FILTER(?birthDate > 1980) . # filter by birth date
}
LIMIT 1 # only one result will be given
```

SELECT query over DBpedia

Question: “Name a person born in Brooklyn after 1980.”

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?name WHERE {
  ?uri rdfs:label ?name . # get name
  ?uri rdf:type dbo:Person . # resource of type Person
  ?uri dbo:birthPlace dbr:Brooklyn . # with birth place Brooklyn
  ?uri dbp:birthDate ?birthDate . # get a birth date
  FILTER(?birthDate > 1980 && LANG(?name) = 'en') .
}
LIMIT 1 # only one result will be given
```

SELECT query over Wikidata

Question: “Name a person born in Brooklyn after 1980.”

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?name
WHERE {
  ?uri rdfs:label ?name . # a resource with a name
  ?uri wdt:P31 wd:Q5 . # instance of (P31) human (Q5)
  ?uri wdt:P19 wd:Q18419 . # birth place (P19) brooklyn (Q18419)
  ?uri wdt:P569 ?birthDate . # birth date (P569)
  FILTER(YEAR(?birthDate) > 1980 && LANG(?name) = "en") .
}
LIMIT 1
```

ASK query over DBpedia

Question: "Does Donald Trump have children?"

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

ASK
WHERE  {
  dbr:Donald_Trump dbo:child ?children . # if resolved then True
}
```

ASK query over DBpedia

Question: “Does Donald Trump have more than 3 children?”

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
ASK {
  { # subquery to get the num of children
    SELECT (COUNT(?children) AS ?howMany)
    WHERE  {
        dbr:Donald_Trump dbo:child ?children .
    }
  } # end of subquery
  FILTER(?howMany > 3) . # filter by constraint
}
```

INSERT query

Store metadata of a NER/NEL component:

```
# this prefix doesn't exist (just an example)
# but we can define it (as a standard/specification)
PREFIX qa: <http://www.ins.hs-anhalt.de/ns/qaannotation/>
PREFIX dbr: <http://dbpedia.org/resource/>

INSERT DATA
{
    GRAPH <replace-with-graph-id>
      {
        <urn:qa:id1> qa:qText "Does Donald Trump have children?" .
        <urn:qa:id1> qa:entities dbr:Donald_Trump  .
        <urn:qa:id1> qa:component <urn:qa:ner:id1>  .
      }
}
```

INSERT query

Store metadata of a Relation Prediction component:

```
# this prefix doesn't exist (just an example)
# but we can define it (as a standard/specification)
PREFIX qa: <http://www.ins.hs-anhalt.de/ns/qaannotation/>
PREFIX dbo: <http://dbpedia.org/ontology/>

INSERT DATA
{
    GRAPH <replace-with-graph-id>
      {
        <urn:qa:id2> qa:qText "Does Donald Trump have children?" .
        <urn:qa:id2> qa:relations dbo:child  .
        <urn:qa:id2> qa:component <urn:qa:classifier:id1>  .
      }
}
```

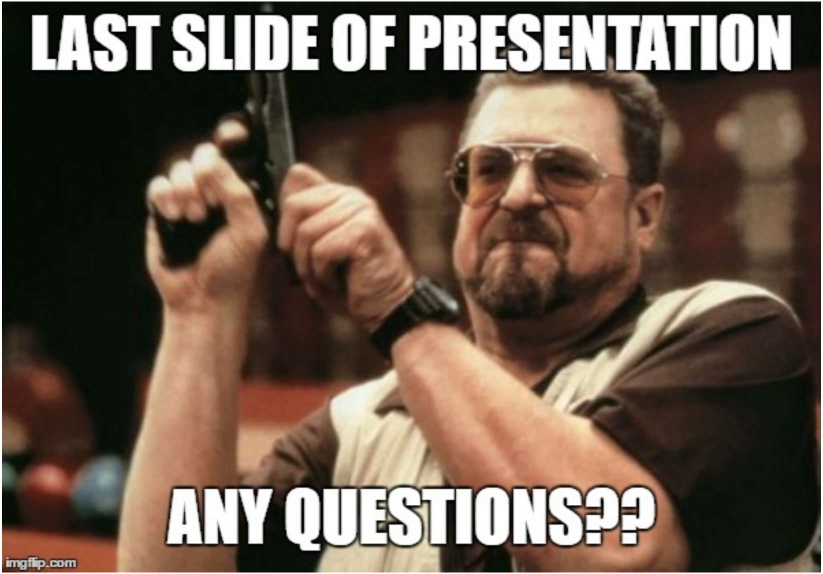


Figure: Any questions?

Exercise 3 – SPARQL queries over Knowledge Graphs

TODOs:

Exercise 3 – SPARQL queries over Knowledge Graphs

TODOs:

- Depending on your exercise variant manually write SPARQL queries for the corresponding questions (over DBpedia and Wikidata).

Exercise 3 – SPARQL queries over Knowledge Graphs

TODOs:

- Depending on your exercise variant manually write SPARQL queries for the corresponding questions (over DBpedia and Wikidata).
- Write a script that reads a query, executes it on a knowledge graph, fetches the answer, and writes it to a JSON file.

Exercise 3 – SPARQL queries over Knowledge Graphs

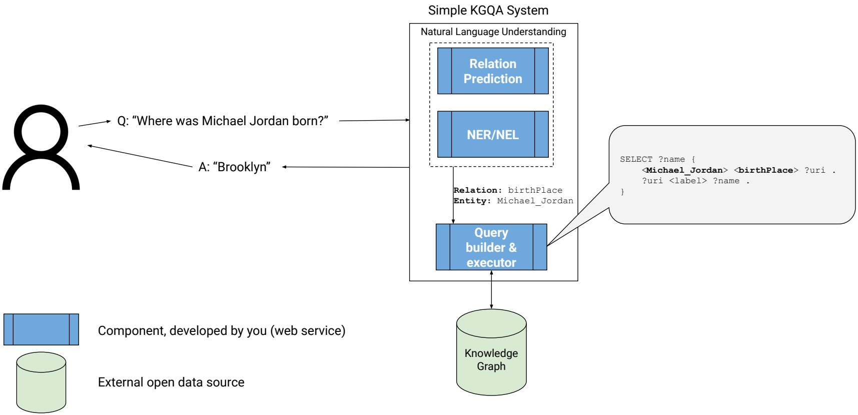
TODOs:

- Depending on your exercise variant manually write SPARQL queries for the corresponding questions (over DBpedia and Wikidata).
- Write a script that reads a query, executes it on a knowledge graph, fetches the answer, and writes it to a JSON file.

Feel free to check example queries or ask help from your teachers.

Let’s do the exercise.

Exercise 4 – next week


Simple KGQA System
Natural Language Understanding
Relation Prediction
NER/NEL
Query builder & executor
Knowledge Graph
Q: "Where was Michael Jordan born?"
A: "Brooklyn"
Component, developed by you (web service)
External open data source

```
Relation: birthPlace
Entity: Michael_Jordan
```

```
SELECT ?name {
    <Michael_Jordan> <birthPlace> ?uri .
    ?uri <label> ?name .
}
```

Figure: Architecture of a Simple KGQA system

Started from the zero, now we here:

0 Introduction;

1 NER & NEL;

2 Question classification & Web service/API;

3 **SPARQL queries over Knowledge Graphs**;

4 Simple KGQA system – based on exercises 0, 1, 2, 3;

5 Qanary Framework – component oriented approach;

6 Simple ODQA system?;

7 Evaluation of QA systems.