Knowledge Graphs in Python

Ernesto Jiménez-Ruiz
Lecturer in Artificial Intelligence

Vocabularies

Vocabularies

- Families of related notions are grouped into *vocabularies*.
- Some important, well-known namespaces—and prefixes:

  Modelling vocabulary:

  `rdf:` `<http://www.w3.org/1999/02/22-rdf-syntax-ns#>` – RDF

  `rdfs:` `<http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

  `owl:` `<http://www.w3.org/2002/07/owl#>` – OWL

  Support vocabularies:

  `foaf:` `<http://xmlns.com/foaf/0.1/>` – Friend of a friend

  `dc:` `<http://purl.org/dc/terms/>` – Dublin Core

  `bfo:` `<http://purl.obolibrary.org/obo/bfo.owl#>` – Basic Formal Ontology

Vocabularies

- Families of related notions are grouped into *vocabularies*.
- Some important, well-known namespaces—and prefixes:

  Modelling vocabulary:

  `rdf:` `<http://www.w3.org/1999/02/22-rdf-syntax-ns#>` – RDF

  `rdfs:` `<http://www.w3.org/2000/01/rdf-schema#>` – RDF Schema

  `owl:` `<http://www.w3.org/2002/07/owl#>` – OWL

  Support vocabularies:

  `foaf:` `<http://xmlns.com/foaf/0.1/>` – Friend of a friend

  `dc:` `<http://purl.org/dc/terms/>` – Dublin Core

  `bfo:` `<http://purl.obolibrary.org/obo/bfo.owl#>` – Basic Formal Ontology

- Usually, a description is published at the namespace base URI.
- Note that the prefix is not standardised.

Example vocabularies: RDF, RDFS

RDF: describing RDF graphs.

- `rdf:Statement`
- `rdf:subject`, `rdf:predicate`, `rdf:object`
- `rdf:type`

RDFS: describing RDF vocabularies.

- `rdfs:Class`
- `rdfs:subClassOf`, `rdfs:subPropertyOf`
- `rdfs:domain`, `rdfs:range`
- `rdfs:label`

Examples:

```
dbr:London rdf:type dbo:City .
dbr:London rdfs:label "London"@en .
dbo:City rdfs:subClassOf dbo:Place .
```

Example vocabularies: OWL

OWL: describing ontologies

- `owl:inverseOf`
- `owl:equivalentClass`
- `owl:disjointWith`
- `owl:sameAs`

Examples:

```
dbr:London owl:sameAs ex:London .
dbo:location owl:inverseOf dbo:isLocatedIn .
dbo:City owl:disjointWith dbo:Person .
dbo:City owl:equivalentClass ex:City .
```

Example vocabularies: FOAF, Dublin Core

FOAF: person data and relations.

- `foaf:Person`
- `foaf:knows`
- `foaf:firstName`,
  `foaf:lastName`,
  `foaf:gender`

Example vocabularies: FOAF, Dublin Core

FOAF: person data and relations.

- `foaf:Person`
- `foaf:knows`
- `foaf:firstName`, `foaf:lastName`, `foaf:gender`

Dublin Core: library metadata.

- `dc:creator`, `dc:contributor`
- `dc:format`, `dc:language`, `dc:licence`

Example vocabularies: FOAF, Dublin Core

FOAF: person data and relations.

- `foaf:Person`
- `foaf:knows`
- `foaf:firstName`, `foaf:lastName`, `foaf:gender`

Dublin Core: library metadata.

- `dc:creator`, `dc:contributor`
- `dc:format`, `dc:language`, `dc:licence`

Examples:

```
city:ernesto rdf:type foaf:Person .
city:ernesto foaf:knows city:carlos .
city:IN3067-INM713 dc:creator city:ernesto .
```

Example vocabularies: BFO

- Basic Formal Ontology:
  `http://www.obofoundry.org/ontology/bfo.html`
- It is an “upper level ontology”
- Lays the foundations of many ontologies in the biological domain.
- *e.g.*, `http://bioportal.bioontology.org/`

Recap: Other vocabularies

From KGs like DBpedia:

- Prefix `dbr:` `<http://dbpedia.org/resource/>`
- Prefix `dbo:` `<http://dbpedia.org/ontology/>`
- Prefix `dbp:` `<http://dbpedia.org/property/>`
- Examples:

```
dbr:london rdf:type dbo:City .
```

New vocabularies:

- Prefix `city:` `<http://www.example.org/university/london/city#>`
- Prefix `phdcourse:` `<http://www.semanticweb.org/ernesto/aalborg/phd/>`

RDF in Python

RDF in Python with RDFLib (i)

- We rely on RDFLib: `from rdflib import Graph`
- Creates empty graph: `g = Graph()`

RDF in Python with RDFLib (i)

- We rely on RDFLib: `from rdflib import Graph`
- Creates empty graph: `g = Graph()`
- Loads an RDF graph: `g.parse("beatles.ttl", format="ttl")`
- Saves and RDF graph:
  `g.serialize(destination='beatles.rdf', format='xml')`

RDF in Python with RDFLib (i)

- We rely on RDFLib: `from rdflib import Graph`
- Creates empty graph: `g = Graph()`
- Loads an RDF graph: `g.parse("beatles.ttl", format="ttl")`
- Saves and RDF graph:
  `g.serialize(destination='beatles.rdf', format='xml')`
- Iterates over a graph:

```
for s, p, o in g:
    print((s.n3(), p.n3(), o.n3()))
```

RDF in Python with RDFLib (ii)

- Basic triple elements: `from rdflib import URIRef, BNode, Literal`
- Creates an URI:
  `ernesto = URIRef("http://ex.org/univ/city#ernesto")`
- Creates a blank node: `bnode = BNode()`
- Creates a literal: `year = Literal('2021', datatype=XSD.gYear)`

RDF in Python with RDFLib (iii)

– Namespaces:

```
from rdflib import Namespace
```

RDF in Python with RDFLib (iii)

- Namespaces:
  `from rdflib import Namespace`
- Default namespaces and vocabulary: `from rdflib.namespace import OWL, RDF, RDFS, FOAF, XSD`
  - *e.g.*, RDF.type is equivalent to
    `URIRef("http://www.w3.org/1999/02/22-rdf-syntax-ns#type")`

RDF in Python with RDFLib (iii)

- Namespaces:
  `from rdflib import Namespace`
- Default namespaces and vocabulary: `from rdflib.namespace import OWL, RDF, RDFS, FOAF, XSD`
  - *e.g.*, RDF.type is equivalent to
    `URIRef("http://www.w3.org/1999/02/22-rdf-syntax-ns#type")`
- User defined:
  `city = Namespace("http://ex.org/univ/city#")`
  - *e.g.*, `city.ernesto` is equivalent to
    `URIRef("http://ex.org/univ/city#ernesto")`

RDF in Python with RDFLib (iv)

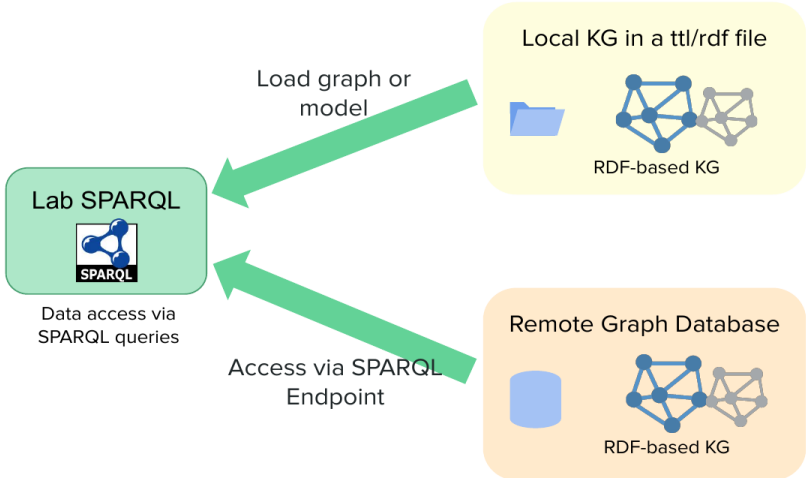
– Adding triples:

```
g.add((city.ernesto, RDF.type, FOAF.Person))
g.add((city.ernesto, FOAF.name, name))
g.add((city.ernesto, city.teaches, city.inm713))
```

– Prefixes: `g.bind("city", city)`

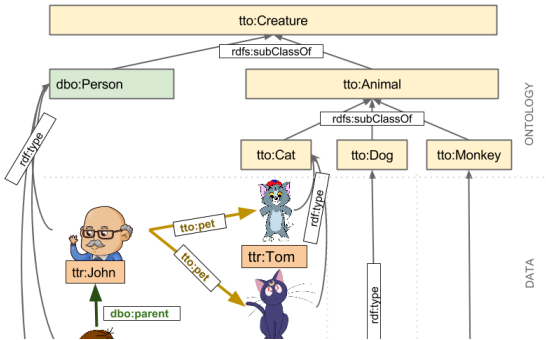
SPARQL in Python

SPARQL: local and remote KG access



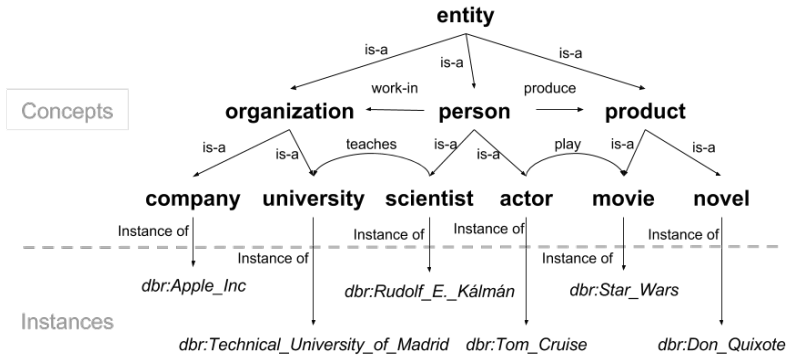
SPARQL Playground

– Based on discontinued platform to learn SPARQL.
`http://sparql-playground.sib.swiss/`



DBpedia Knowledge Graph (i)

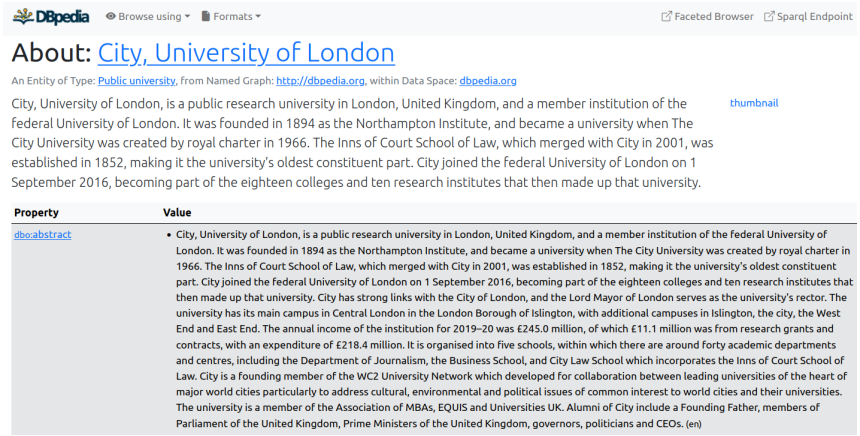
– Ontology/KG: `https://www.dbpedia.org/resources/ontology/`



(*) Image from `https://github.com/gsi-upm/sematch/`

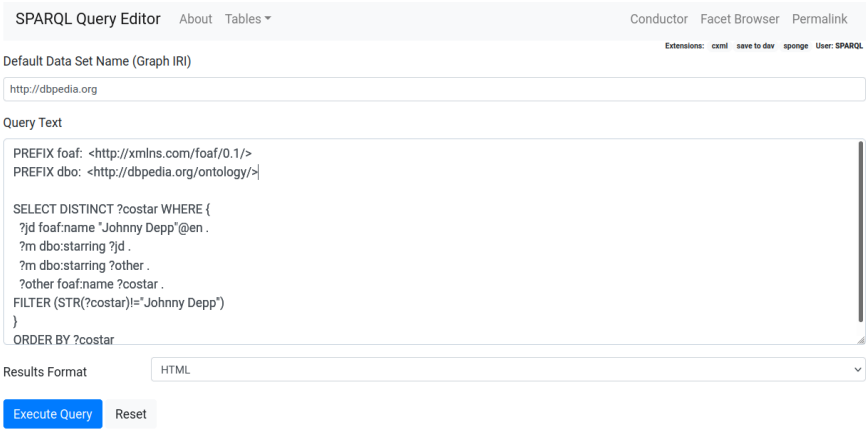
DBpedia Knowledge Graph (ii)

– Linked data Interface: `https://www.dbpedia.org/resources/linked-data/`



DBPedia Knowledge Graph (iii)

– SPARQL Endpoint: `http://dbpedia.org/sparql`



SPARQL in Python: Querying Local Graph with RDFLib

– Querying a local Graph:

```
qres = g.query(
    """SELECT ?thing ?name WHRE {
        ?thing tto:sex "female" .
        ?thing dbp:name ?name .
    }""")
```

– Iterate over the results:

```
for row in qres:
    print("%s is female with name ‘%s’" % (str(row.thing),str(row.name)))
```

– `row` is a dictionary with the RDF terms that match the output variables.

SPARQL in Python: Remote Access with SPARQLWrapper (i)

- SPARQLWrapper: deals with the connection to a SPARQL endpoint
- A SPARQL Endpoint is a service to receive and process SPARQL queries following a protocol.
- Connection: `sparql_web = SPARQLWrapper("http://dbpedia.org/sparql")`
- Set results format (default XML): `sparql_web.setReturnFormat(JSON)`

SPARQL in Python: Remote Access with SPARQLWrapper (ii)

- Set SPARQL query:

```
sparql_web.setQuery("""
    SELECT DISTINCT ?costar WHERE {
        ?jd foaf:name "Johnny Depp"@en .
        ?m dbo:starring ?jd .
        ?m dbo:starring ?costar .  }
""")
```

- Get (json) results: `results = sparql_web.query().convert()`
- Iterate over the (json) results:

```
for result in results["results"]["bindings"]:
    print(result["costar"]["value"])
```