OntoforceDataAnalysis

September 23, 2016

1 General statistics about the Ontoforce dataset

- Documentation of wrapper http://rdflib.github.io/sparqlwrapper/doc/latest/
- Statistical queries found here: https://code.google.com/p/void-impl/wiki/SPARQLQueriesForStatistics

```
In [1]: from SPARQLWrapper import SPARQLWrapper, JSON
        import pandas as pd
```

```
In [2]: def performSparqlQuery(sparqlURI, queryStr):
            sparql = SPARQLWrapper(sparqlURI)
            sparql.setQuery(queryStr)
            sparql.setReturnFormat(JSON)
            results = sparql.query().convert()
            return results['results']['bindings']
```

```
In [3]: sparql_endpoint = "http://ec2-54-172-160-219.compute-1.amazonaws.com"
        port = 80

        def generate_endpoint_uri(sparql_endpoint, port):
            return sparql_endpoint + ":" + str(port) + "/sparql"


        virtuoso_endpoint = generate_endpoint_uri(sparql_endpoint,port)
        print(virtuoso_endpoint)
```

```
http://ec2-54-172-160-219.compute-1.amazonaws.com:80/sparql
```

1.1 1. Number of triples => OK

```
In [18]: queryString ="SELECT (COUNT(*) AS ?numtriples) { ?s ?p ?o  }"
         results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
In [19]: print(results)
         print ("Number of triples is: " + str(results[0]['numtriples']['value']))
         print ("2.37 billion triples")
```

```
[{'numtriples': {'value': '2374837593', 'datatype': 'http://www.w3.org/2001/XMLSchema#integer', 'type':
Number of triples is: 2374837593
2.37 billion triples
```

1.2 2. Total number of entities => OK

```
In [22]: queryString ="SELECT (COUNT(distinct ?s) AS ?numsubjects) { ?s a []  }"
         results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
In [24]: print(results)
         print (results[0]['numsubjects']['value'])
         print ("136.3 million entities ")
```

```
[{'numsubjects': {'value': '136313277', 'datatype': 'http://www.w3.org/2001/XMLSchema#integer', 'type':
136313277
136.3 million entities
```

1.3 3. Total number of distinct classes => OK

```
In [25]: queryString = """
         PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
         SELECT (COUNT(distinct ?o) AS ?distinctclasses) { ?s rdf:type ?o }
         """
         results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
In [27]: print(results)
         print (results[0]['distinctclasses']['value'])
         print("2434 distinct classes")
```

```
[{'distinctclasses': {'value': '2434', 'datatype': 'http://www.w3.org/2001/XMLSchema#integer', 'type':
2434
2434 distinct classes
```

1.4 4. Total number of distinct predicates => OK

```
In [28]: queryString = """
         SELECT (COUNT(distinct ?p) as ?distpredicates) { ?s ?p ?o }
         """
         results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
In [31]: print(results)
         print (results[0]['distpredicates']['value'])
         print("1782 distinct predicates")
```

```
[{'distpredicates': {'value': '1782', 'datatype': 'http://www.w3.org/2001/XMLSchema#integer', 'type': 't
1782
1782 distinct predicates
```

1.5 Idea: show

1.6 5. Total number of distinct subject nodes => Timeout

```
In [32]: queryString = """
         SELECT (COUNT(DISTINCT ?s ) AS ?distsubjects) {  ?s ?p ?o   }
         """
         results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)

<ipython-input-32-681fd25f14c3> in <module>()
  2 SELECT (COUNT(DISTINCT ?s ) AS ?distsubjects) {  ?s ?p ?o   }
  3 """
```

```
----> 4 results = performSparqlQuery(virtuoso_endpoint, queryString)


    <ipython-input-10-369c16384698> in performSparqlQuery(sparqlURI, queryStr)
      3     sparql.setQuery(queryStr)
      4     sparql.setReturnFormat(JSON)
----> 5     results = sparql.query().convert()
      6     return results['results']['bindings']
      7


    /home/ddewitte/anaconda3/lib/python3.4/site-packages/SPARQLWrapper/Wrapper.py in query(self)
    533             @rtype: L{QueryResult} instance
    534         """
--> 535         return QueryResult(self._query())
    536
    537     def queryAndConvert(self):


    /home/ddewitte/anaconda3/lib/python3.4/site-packages/SPARQLWrapper/Wrapper.py in _query(self)
    503
    504         try:
--> 505             response = urlopener(request)
    506             return response, self.returnFormat
    507         except urllib.error.HTTPError as e:


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in urlopen(url, data, timeout, cafile,
    159     else:
    160         opener = _opener
--> 161     return opener.open(url, data, timeout)
    162
    163 def install_opener(opener):


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in open(self, fullurl, data, timeout)
    461             req = meth(req)
    462
--> 463         response = self._open(req, data)
    464
    465         # post-process response


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in _open(self, req, data)
    479         protocol = req.type
    480         result = self._call_chain(self.handle_open, protocol, protocol +
--> 481                                   '_open', req)
    482         if result:
    483             return result


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in _call_chain(self, chain, kind, meth_
    439         for handler in handlers:
    440             func = getattr(handler, meth_name)
```

```
--> 441             result = func(*args)
    442             if result is not None:
    443                 return result


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in http_open(self, req)
   1208
   1209     def http_open(self, req):
-> 1210         return self.do_open(http.client.HTTPConnection, req)
   1211
   1212     http_request = AbstractHTTPHandler.do_request_


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in do_open(self, http_class, req, **htt
   1183             except OSError as err: # timeout error
   1184                 raise URLError(err)
-> 1185             r = h.getresponse()
   1186         except:
   1187             h.close()


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in getresponse(self)
   1169
   1170         try:
-> 1171             response.begin()
   1172             assert response.will_close != _UNKNOWN
   1173             self.__state = _CS_IDLE


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in begin(self)
    349         # read until we get a non-100 response
    350         while True:
--> 351             version, status, reason = self._read_status()
    352             if status != CONTINUE:
    353                 break


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in _read_status(self)
    311
    312     def _read_status(self):
--> 313         line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
    314         if len(line) > _MAXLINE:
    315             raise LineTooLong("status line")


    /home/ddewitte/anaconda3/lib/python3.4/socket.py in readinto(self, b)
    372         while True:
    373             try:
--> 374                 return self._sock.recv_into(b)
    375             except timeout:
    376                 self._timeout_occurred = True


    KeyboardInterrupt:
```

```
In [ ]: print(results)
        print (results[0]['distsubjects']['value'])
```

1.7 6. Total number of distinct object nodes => OK

```
In [16]: queryString = """
         SELECT (COUNT(DISTINCT ?o ) AS ?distobjects) {  ?s ?p ?o  filter(!isLiteral(?o)) }
         """
         results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
In [18]: print(results)
         print (results[0]['distobjects']['value'])
         print ( '286.7 million object nodes')
```

```
[{'distobjects': {'type': 'typed-literal', 'value': '286749072', 'datatype': 'http://www.w3.org/2001/XMI
286749072
286.7 million object nodes
```

1.8 7. Exhaustive list of classes used in the dataset (NDA) => Timeout

```
In [ ]: queryString = """
        SELECT DISTINCT ?type { ?s a ?type }
        """
        results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
In [ ]: with open("OntoforceClasses.txt") as f:
            for c in results:
                f.write(c['type']['value'])
```

1.9 8. Exhaustive list of properties used in the dataset (NDA) => OK

```
In [4]: queryString = """
        SELECT DISTINCT ?p { ?s ?p ?o }
        """
        results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
In [8]: with open("OntoforceProperties.txt", 'w+') as f:
            for c in results:
                f.write(c['p']['value'] + "\n")
```

```
In [9]: #IDEE: aantal predicates per namespace? ns.ontoforce.com, purl.rdf.ebi,... => wsch nog leuke pl
        #predicates van ontoforce zelf zijn?
```

1.10 9. Table: class vs. total number of instances of the class (NDA) => OK

```
In [14]: queryString = """
         SELECT  ?class (COUNT(?s) AS ?count ) { ?s a ?class } GROUP BY ?class ORDER BY ?count
         """
         results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
In [15]: with open("OntoforceInstancesPerClass.txt", 'w+') as f:
             for c in results:
                 f.write(c['class']['value'] + "\t")
                 f.write(c['count']['value'] + "\n")
```

```
In [ ]: ## Distribution

        classes = []
        counts = []

        for c in results:
                classes.append(c['class']['value'])
                counts.append(c['count']['value'])

        cc_dict = { "classes": classes, "counts": counts}

In [ ]: #classmethod DataFrame.from_dict(data, orient='columns', dtype=None)
```

1.11 10. Table: property vs. total number of triples using the property (NDA) => OK

```
In [10]: queryString = """
         SELECT  ?p (COUNT(?s) AS ?count ) { ?s ?p ?o } GROUP BY ?p ORDER BY ?count
         """
         results = performSparqlQuery(virtuoso_endpoint, queryString)

In [12]: with open("OntoforceTriplesPerProperty.txt",'w+') as f:

             for c in results:
                 f.write(c['p']['value'] + "\t")
                 f.write(c['count']['value'] + "\n")

In [ ]: ## Distribution

        predicates = []
        counts = []

        for c in results:
                predicates.append(c['p']['value'])
                counts.append(c['count']['value'])

        cc_dict = { "predicates": predicates, "counts": counts}

In [ ]: #classmethod DataFrame.from_dict(data, orient='columns', dtype=None)
```

1.12 11. Table: property vs. total number of distinct subjects in triples using the property => Timeout

```
In [13]: queryString = """
         SELECT  ?p (COUNT(DISTINCT ?s ) AS ?count ) { ?s ?p ?o } GROUP BY ?p ORDER BY ?count
         """
         results = performSparqlQuery(virtuoso_endpoint, queryString)


        ---------------------------------------------------------------------------

        KeyboardInterrupt                         Traceback (most recent call last)

        <ipython-input-13-7fab9a0e291e> in <module>()
          2 SELECT  ?p (COUNT(DISTINCT ?s ) AS ?count ) { ?s ?p ?o } GROUP BY ?p ORDER BY ?count
```

```
  3 """
----> 4 results = performSparqlQuery(virtuoso_endpoint, queryString)


<ipython-input-2-369c16384698> in performSparqlQuery(sparqlURI, queryStr)
  3     sparql.setQuery(queryStr)
  4     sparql.setReturnFormat(JSON)
----> 5     results = sparql.query().convert()
  6     return results['results']['bindings']
  7


/home/ddewitte/anaconda3/lib/python3.4/site-packages/SPARQLWrapper/Wrapper.py in query(self)
533             @rtype: L{QueryResult} instance
534         """
--> 535         return QueryResult(self._query())
536
537     def queryAndConvert(self):


/home/ddewitte/anaconda3/lib/python3.4/site-packages/SPARQLWrapper/Wrapper.py in _query(self)
503
504         try:
--> 505             response = urlopener(request)
506             return response, self.returnFormat
507         except urllib.error.HTTPError as e:


/home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in urlopen(url, data, timeout, cafile,
159     else:
160         opener = _opener
--> 161     return opener.open(url, data, timeout)
162
163 def install_opener(opener):


/home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in open(self, fullurl, data, timeout)
461             req = meth(req)
462
--> 463         response = self._open(req, data)
464
465         # post-process response


/home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in _open(self, req, data)
479         protocol = req.type
480         result = self._call_chain(self.handle_open, protocol, protocol +
--> 481                                   '_open', req)
482         if result:
483             return result


/home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in _call_chain(self, chain, kind, meth_
439         for handler in handlers:
```

```
    440             func = getattr(handler, meth_name)
--> 441             result = func(*args)
    442             if result is not None:
    443                 return result


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in http_open(self, req)
   1208
   1209     def http_open(self, req):
-> 1210         return self.do_open(http.client.HTTPConnection, req)
   1211
   1212     http_request = AbstractHTTPHandler.do_request_


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in do_open(self, http_class, req, **htt
   1183             except OSError as err: # timeout error
   1184                 raise URLError(err)
-> 1185             r = h.getresponse()
   1186         except:
   1187             h.close()


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in getresponse(self)
   1169
   1170         try:
-> 1171             response.begin()
   1172             assert response.will_close != _UNKNOWN
   1173             self.__state = _CS_IDLE


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in begin(self)
    349         # read until we get a non-100 response
    350         while True:
--> 351             version, status, reason = self._read_status()
    352             if status != CONTINUE:
    353                 break


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in _read_status(self)
    311
    312     def _read_status(self):
--> 313         line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
    314         if len(line) > _MAXLINE:
    315             raise LineTooLong("status line")


    /home/ddewitte/anaconda3/lib/python3.4/socket.py in readinto(self, b)
    372         while True:
    373             try:
--> 374                 return self._sock.recv_into(b)
    375             except timeout:
    376                 self._timeout_occurred = True
```

```
        KeyboardInterrupt:
```

In [ ]:
```
with open("OntoforceDistinctSubjectsPerProperty.txt") as f:

    for c in results:
        f.write(c['p']['value'])
        f.write(c['count']['value'])
```

In [ ]:
```
## Distribution

predicates = []
counts = []

for c in results:
        predicates.append(c['p']['value'])
        counts.append(c['count']['value'])

cc_dict = { "predicates": predicates, "counts": counts}
```

In [ ]:
```
#classmethod DataFrame.from_dict(data, orient='columns', dtype=None)
```

1.13 12. Table: property vs. total number of distinct objects in triples using the property => Timeout

In [19]:
```
queryString = """
SELECT  ?p (COUNT(DISTINCT ?o ) AS ?count ) { ?s ?p ?o } GROUP BY ?p ORDER BY ?count
"""
results = performSparqlQuery(virtuoso_endpoint, queryString)
```

```
---------------------------------------------------------------------------

KeyboardInterrupt                         Traceback (most recent call last)

<ipython-input-19-e10357a052e0> in <module>()
  2 SELECT  ?p (COUNT(DISTINCT ?o ) AS ?count ) { ?s ?p ?o } GROUP BY ?p ORDER BY ?count
  3 """
----> 4 results = performSparqlQuery(virtuoso_endpoint, queryString)


<ipython-input-2-369c16384698> in performSparqlQuery(sparqlURI, queryStr)
  3     sparql.setQuery(queryStr)
  4     sparql.setReturnFormat(JSON)
----> 5     results = sparql.query().convert()
  6     return results['results']['bindings']
  7


/home/ddewitte/anaconda3/lib/python3.4/site-packages/SPARQLWrapper/Wrapper.py in query(self)
533             @rtype: L{QueryResult} instance
534         """
--> 535         return QueryResult(self._query())
536
537     def queryAndConvert(self):
```

```
    /home/ddewitte/anaconda3/lib/python3.4/site-packages/SPARQLWrapper/Wrapper.py in _query(self)
    503
    504         try:
--> 505             response = urlopener(request)
    506             return response, self.returnFormat
    507         except urllib.error.HTTPError as e:


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in urlopen(url, data, timeout, cafile,
    159     else:
    160         opener = _opener
--> 161     return opener.open(url, data, timeout)
    162
    163 def install_opener(opener):


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in open(self, fullurl, data, timeout)
    461             req = meth(req)
    462
--> 463         response = self._open(req, data)
    464
    465         # post-process response


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in _open(self, req, data)
    479         protocol = req.type
    480         result = self._call_chain(self.handle_open, protocol, protocol +
--> 481                                   '_open', req)
    482         if result:
    483             return result


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in _call_chain(self, chain, kind, meth_
    439         for handler in handlers:
    440             func = getattr(handler, meth_name)
--> 441             result = func(*args)
    442             if result is not None:
    443                 return result


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in http_open(self, req)
   1208
   1209     def http_open(self, req):
-> 1210         return self.do_open(http.client.HTTPConnection, req)
   1211
   1212     http_request = AbstractHTTPHandler.do_request_


    /home/ddewitte/anaconda3/lib/python3.4/urllib/request.py in do_open(self, http_class, req, **htt
   1183             except OSError as err: # timeout error
   1184                 raise URLError(err)
-> 1185             r = h.getresponse()
```

```
   1186         except:
   1187             h.close()


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in getresponse(self)
   1169
   1170         try:
-> 1171             response.begin()
   1172             assert response.will_close != _UNKNOWN
   1173             self.__state = _CS_IDLE


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in begin(self)
    349         # read until we get a non-100 response
    350         while True:
--> 351             version, status, reason = self._read_status()
    352             if status != CONTINUE:
    353                 break


    /home/ddewitte/anaconda3/lib/python3.4/http/client.py in _read_status(self)
    311
    312     def _read_status(self):
--> 313         line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
    314         if len(line) > _MAXLINE:
    315             raise LineTooLong("status line")


    /home/ddewitte/anaconda3/lib/python3.4/socket.py in readinto(self, b)
    372         while True:
    373             try:
--> 374                 return self._sock.recv_into(b)
    375             except timeout:
    376                 self._timeout_occurred = True


    KeyboardInterrupt:
```

```
In [ ]: with open("OntoforceDistinctObjectsPerProperty.txt") as f:

            for c in results:
                f.write(c['p']['value'])
                f.write(c['count']['value'])
```

```
In [ ]: ## Distribution

        predicates = []
        counts = []

        for c in results:
                predicates.append(c['p']['value'])
                counts.append(c['count']['value'])

        cc_dict = { "predicates": predicates, "counts": counts}
```

```
In [ ]: #classmethod DataFrame.from_dict(data, orient='columns', dtype=None)
```