Project 1 : Query answering over Linked Data

Author : François Amat

Contact: amat.francois@gmail.com

This project goal is to answer to the following two questions :

- Q1 : Can you find the answers to (one of) the questions proposed in cours1 (pages 10-15) or the question mentioned in Tim Berners-Lee's TED talk (2009) 1 between 11'20 and 12'30?

- Q2 : Propose another question in natural language and give your answer by checking Linked Data.

Q1

- Cours 1 Questions:

    - Find that landmark article on data integration written by an Indian researcher in the 1990s. This question is quite hard because it has a lot of complex attributes :
        - how do we define landmark
        - how can we check that a researcher is Indian
        - What concepts are behind the "data integration"

    In order to answer that I first begun to search a graph with all the research papers, I could not find one, I tried to query the website arxiv.org with their api but I was very unhappy of the results (queries took a long time and I could not figure out a easy way to search among titles nor abstracts).

    Then I found the aminer.org website, I manage to download the large 14Go archive containing researchs papers in a form of JSON files.

    I exported all the json files into a mongodb local server and I queried it with a python programm.

    I define *landmark* as the number of references (degrees in graph)

I check if one of the author is Indian or not by searching them in google. It can be denoted that a classifier that associate familly name and nationality can be implemented.
Additional ressources

```
from pymongo import MongoClient
import networkx as nx
f = open("results"+date_name+".txt","w")
MONGO_URL= "localhost"
MONGO_PORT=27017
client = MongoClient(MONGO_URL, int(MONGO_PORT))
db = client.aminer
articles = db.articles
```

In the following code I query the mongodb server to get articles between 1990 and 2000 that correspond to the regex given in argument.
After that, I construct a graph with all the results given after the query.

```
def get_all_articles(query,client=client):
  start = time.perf_counter()
  db = client.aminer
  #db.authenticate(MONGO_USERNAME, MONGO_PASSWORD)
  articles = db.articles
  graph  = nx.DiGraph()

  for article in articles.find({'title':{ '$regex': '^'+query+''},
                                'year':{ '$gt': 1990, '$lt': 2000 }}):
      graph.add_node(article['id'])
      if("references" in article.keys()):
          for ref in article['references']:
              if not ref in graph:
                      graph.add_node(ref)
              graph.add_edge(article['id'], ref)
  elapsed = time.perf_counter() - start
  print('Elapsed %.3f seconds.' % elapsed)
  return graph
```

In the following code, I sort the graph to the *landmark* metric which is the degree of the article in the graph.
Then, I take the first 5 results.

```
def find_authors_bigger_degree_article(G):
  start = time.perf_counter()

  db = client.aminer
  articles = db.articles
```

```
    couples = [(node,val) for (node, val) in G.degree()]
    couples.sort(key= lambda x: -x[1])
    best_articles = []

    for coup in couples[:5]:
        for article in articles.find({'id':coup[0]}):
            best_articles.append(article)
    elapsed = time.perf_counter() - start
    authors = []
    print('Elapsed1 %.3f seconds.' % elapsed)
    for articles in best_articles:
        authors.append(articles["authors"])
        print(articles["authors"])
        print(articles["authors"],file=f)
    elapsed2 = time.perf_counter() - elapsed
    print('Elapsed2 %.3f seconds.' % elapsed2)
    print(authors, file=f)
    return couples, best_articles, authors
```

Results :

On the queries : ["integration","*.data.*integration.*","*.integration..**"]
I obtained the following authors :

```
Christine PARENT
Stefano SPACCAPIETRA
Christine PARENT
Stefano SPACCAPIETRA
Ken McGarry
James Chambers
Giles Oatley
Paea Lependu
Dejing Dou
Ken McGarry
Sheila Garfleld
Nick Morris
O. Bodenreider
```

After a google check none of theses authors are Indian.
So I change the query from data integration to data warehouse
(query = ["*.data.*ware.**"]) I obtain these authors :

```
Duane Jeffrey Jarc
Jennifer Widom
Dallan Quass
Barinderpal Singh Mumick
```

```
Inderpal Singh Mumick
Yannis Kotidis
Nick Roussopoulos
Douglas E. Bassett Jr
Michael B. Eisen
Mark S. Boguski
Rosetta Inpharmatics
Duane Jeffrey Jarc
Michael Bliss Feldman
```

I found that Barinderpal Singh Mumick is Indian, therefore the paper :
Maintenance of data cubes and summary tables in a warehouse, published in 1997, quoted 447 times is a good result.

It can be also denoted that all the computation done above is very heavy, each query took me around 10 minutes and to find the best papers around 40-50 minutes.

- Are lobsters spiders ?

  This question can be translated by "Are the lobster in the same class of the spiders ? "
  As the fact that spiders are a subclass of arachnid,
  we can ask if lobster are a subclass of arachnid :

```
prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix ex:   <http://example.org/>
prefix rdfs: <http://www.w3.org/2000/01/rdf?schema#>
prefix db:   <http://dbpedia.org/resource/>

ASK
WHERE {db:lobster rdfs::subClassOf db:Arachnid . }
```

  The result is *False*.
  Indeed, If we want to construct the graph where an animal is a lobster and is of subClassOf arcachnid and we obtain an empty graph :

```
 prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix ex:   <http://example.org/>
prefix rdfs: <http://www.w3.org/2000/01/rdf?schema#>
prefix db:   <http://dbpedia.org/resource/>
prefix owl: <http://www.w3.org/2002/07/owl#>

Construct{?a rdfs:label db:Arachnid}
  WHERE {
  ?a rdfs:subClassOf db:Arachnid .
  ?a owl:equivalentClass db:lobster. }
```

Q2

Propose another question in natural language and give your answer by checking Linked Data.

My question is "what are the name of the sofware developping methods":

```
construct {?method rdfs:label ?name }
WHERE {
  ?method wdt:P31 wd:Q1378470.
  ?method rdfs:label ?name.
  FILTER(LANGMATCHES(LANG(?name), "en"))
}
LIMIT 100
```

This query can be executed here