Midterm sim - Fri 05, Nov 2021

November 5, 2021

Scientific Programming - Data Science Master @ University of Trento

THIS IS ONLY A SIMULATION: YOU EARN NOTHING, YOU LOSE NOTHING

0.1 Part A - Terence Hill and Bud Spencer movies

Among the greatest gifts of Italy to the world we can certainly count Terence Hill and Bud Spencer movies.

We took their movies data from Wikidata, a project by the Wikimedia foundation which aims to store only machine-readable data, like numbers, strings, and so on interlinked with many references. Each entity in Wikidata has an identifier, for example Terence Hill is the entity Q243430 and Bud Spencer is Q221074.

You are given some CSVs of movies, all having names ending in `-LG.csv`, where LG can be a language tag like `it`, `en`, `de`, `es`... They mostly contain the same data except for the movie labels which are in the corresponding language. The final goal will be displaying the network of movies and put in evidence the ones co-starring the famous duo.

Each file row contains info about a single actor starring in a movie. Multiple lines with same movie id will mean multiple actors are co-starring. We can see an excerpt of **first four** lines of english version: notice second movie has id Q180638 and is co-starred by both Bud Spencer and Terence Hill

[2]:

```
star,starLabel,movie,movieLabel,firstReleased

http://www.wikidata.org/entity/Q221074,Bud
Spencer,http://www.wikidata.org/entity/Q116187,Thieves and
Robbers,1983-02-11T00:00:00Z

http://www.wikidata.org/entity/Q221074,Bud
Spencer,http://www.wikidata.org/entity/Q180638,Odds and
Evens,1978-10-28T00:00:00Z

http://www.wikidata.org/entity/Q243430,Terence
Hill,http://www.wikidata.org/entity/Q180638,Odds and Evens,1978-10-28T00:00:00Z
```

Now open Jupyter and start editing this notebook `exam-2021-11-05.ipynb`

0.2 load

Write a function that given a `filename_prefix` and list of `languages`, parses the corresponding files and RETURNS a **dictionary of dictionaries**, which maps movies id to movies data, in the format as in the exerpt.

- When a label is missing, you will find instead an id like `Q3778078`: substitute it with empty string (HINT: to recognize ids you might use `is_digit()` method)
- convert date numbers to proper integers
- **DO NOT** put constant ids nor language tags in the code (so no `'Q221074'` nor `'it'` ...)

[3]:
```
import csv

def load(filename_prefix, languages):
    raise Exception('TODO IMPLEMENT ME !')

movies_db = load('bud-spencer-terence-hill-movies', ['en', 'it', 'de'])
#movies_db = load('bud-spencer-terence-hill-movies', ['es', 'en', 'de','it'])
```

Complete expected output can be found in expected_db.py

[4]:
```

```

```
EXERPT:
{
  'Q116187': {
               'actors': [('Q221074', 'Bud Spencer')],
               'first_release': (1983, 2, 11),
               'names': {'de': 'Bud, der Ganovenschreck',
                         'en': 'Thieves and Robbers',
                         'it': 'Cane e gatto'}
             }
  'Q180638': {
               'actors': [('Q221074', 'Bud Spencer'), ('Q243430', 'Terence
Hill')],
               'first_release': (1978, 10, 28),
               'names': {'de': 'Zwei sind nicht zu bremsen',
                         'en': 'Odds and Evens',
                         'it': 'Pari e dispari'}
             }
  'Q231967': {
               'actors': [('Q221074', 'Bud Spencer'), ('Q243430', 'Terence
Hill')],
               'first_release': (1981, 1, 1),
               'names': {'de': 'Zwei Asse trumpfen auf',
                         'en': 'A Friend Is a Treasure',
```

```
                        'it': 'Chi trova un amico, trova un tesoro'}
            }
  .
  .
}
```

[5]:
```
# TESTING
from pprint import pformat; from expected_movies_db import expected_movies_db
for sid in expected_movies_db.keys():
    if sid not in movies_db: print('\nERROR: MISSING movie', sid); break
    for k in expected_movies_db[sid]:
        if k not in movies_db[sid]:
            print('\nERROR at movie', sid,'\n\n   MISSING key:', k); break
        if expected_movies_db[sid][k] != movies_db[sid][k]:
            print('\nERROR at movie', sid, 'key:',k)
            print('  ACTUAL:\n', pformat(movies_db[sid][k]))
            print('  EXPECTED:\n', pformat(expected_movies_db[sid][k]))
            break
if len(movies_db) > len(expected_movies_db):
    print('ERROR! There are more movies than expected!')
    print('  ACTUAL:\n', len(movies_db))
    print('  EXPECTED:\n', len(expected_movies_db))
```

0.3 save_table

Write a function that given a movies db and a list of `languages`, writes a new file `merged.csv`

- separate actor names with `and`
- use only the year as date
- file must be formatted like this:

[6]:
```
```

```
movie_id,name en,name it,first_release,actors
Q116187,Thieves and Robbers,Cane e gatto,1983,Bud Spencer
Q180638,Odds and Evens,Pari e dispari,1978,Bud Spencer and Terence Hill
```

[7]:
```
import csv

def save_table(movies, languages):
    raise Exception('TODO IMPLEMENT ME !')

save_table(movies_db, ['en','it'])
#save_table(movies_db, ['de'])
```

```
saved file to merged.csv
```

Complete expected file is in `expected-merged.csv`

[8]:
```
# TESTING
with open('expected-merged.csv',encoding='utf-8', newline='') as expected_f:
    with open('merged.csv',encoding='utf-8', newline='') as f:
        expected_reader = csv.reader(expected_f, delimiter=',')
        reader = csv.reader(f, delimiter=',')
        i = 0
        for expected_row in expected_reader:
            try:
                row = next(reader)
            except:
                print('ERROR at row', i, ': ACTUAL rows are less than EXPECTED!
 ↪')
                break
            for j in range(len(expected_row)):
                if expected_row[j] != row[j]:
                    print('ERROR at row', i, '  cell index', j)
                    print(row)
                    print('\nACTUAL  :', row[j])
                    print('\nEXPECTED:', expected_row[j])
                    break
            i += 1
```

0.4 show_graph

Display a NetworkX graph of movies see examples from `since_year` (included) to `until_year` (included), in the given `language`

- display actor names as capitalized
- display co-starred movies, non co-starred movies and actors with different colors by setting node attribute `fillcolor` (see some color names)

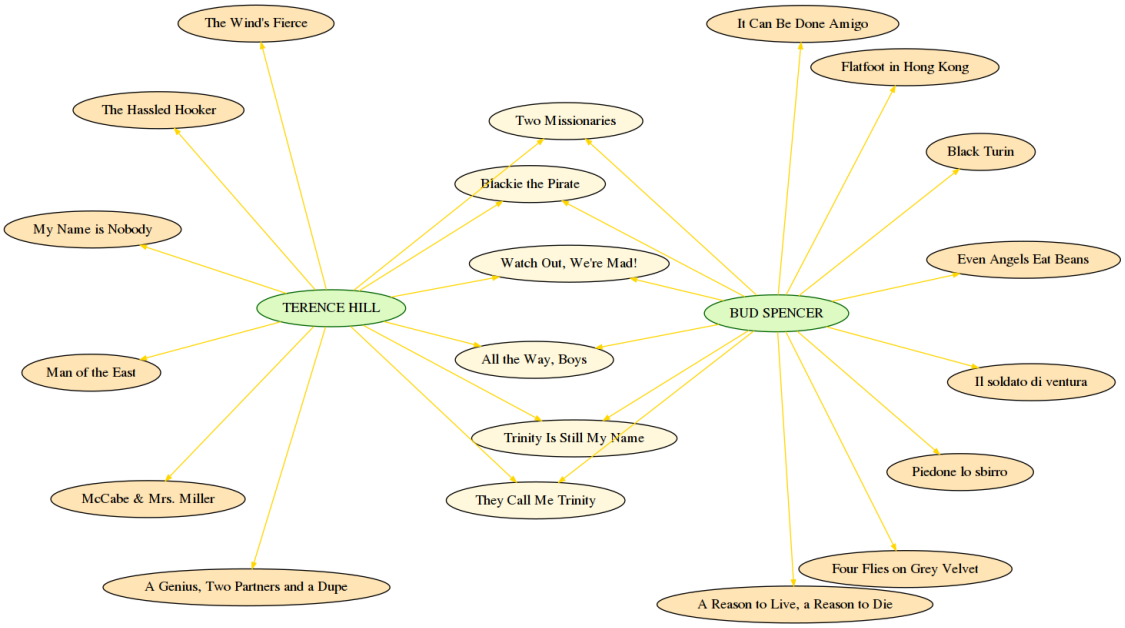
[9]:
```
import networkx as nx
from sciprog import draw_nx

def show_graph(movies, since_year, until_year, language):

    G = nx.DiGraph()
    G.graph['graph']= { 'layout':'neato' }  # don't delete these!

    raise Exception('TODO IMPLEMENT ME !')

show_graph(movies_db, 1970, 1975, 'en')
```



[10]:
```
#show_graph(movies_db, 1970, 1974, 'it')
```

[ ]:
```

```