Initialization

Dependencies

In [1]:
```
pip install kaggle
```

```
Requirement already satisfied: kaggle in /opt/conda/lib/python3.7/site-packages (1.5.12)
Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-packages (from kaggle) (2.25.1)
Requirement already satisfied: python-slugify in /opt/conda/lib/python3.7/site-packages (from kaggle) (5.0.2)
Requirement already satisfied: urllib3 in /opt/conda/lib/python3.7/site-packages (from kaggle) (1.26.7)
Requirement already satisfied: six>=1.10 in /opt/conda/lib/python3.7/site-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /opt/conda/lib/python3.7/site-packages (from kaggle) (2021.10.8)
Requirement already satisfied: python-dateutil in /opt/conda/lib/python3.7/site-packages (from kaggle) (2.8.2)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages (from kaggle) (4.62.3)
Requirement already satisfied: text-unidecode>=1.3 in /opt/conda/lib/python3.7/site-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/site-packages (from requests->kaggle) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/lib/python3.7/site-packages (from requests->kaggle) (4.0.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import os

root_dir = os.path.dirname(os.path.abspath('PageRank_IMDB.jpynb'))

content_dir = os.path.join(root_dir, "content/")
if not os.path.isdir(content_dir):
    os.mkdir(content_dir)

kaggle_dir = os.path.join(root_dir, ".kaggle/")
if not os.path.isdir(kaggle_dir):
    os.mkdir(kaggle_dir)

variables_dir = os.path.join(content_dir, "variables/")
if not os.path.isdir(variables_dir):
    os.mkdir(variables_dir)
```

In [3]:
```
# DO NOT RUN ON DEBIAN VM, JDK IS PREINSTALLED
#!sudo apt-get install openjdk-11-jdk-headless -qq > /dev/null
```

In [4]:
```
import gc
import json
import zipfile
import pickle
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import sys

def getsize(obj):
    print('{:.2f} MB'.format(sys.getsizeof(obj)/(2**20)))
```

In [5]:
```
#@title API
api_token = {"username":"fabio130497","key":"31a9c66aa9c83a4ed5a4d33acff7e78b"}
with open(os.path.join(kaggle_dir, 'kaggle.json'), 'w') as file:
    json.dump(api_token, file)
#!chmod 600 /home/jupyter/.kaggle/kaggle.json
```

Data acquisition

In [6]:
```
!kaggle datasets download -d ashirwadsangwan/imdb-dataset
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /home/jupyter/.kaggle/kaggle.json'
Downloading imdb-dataset.zip to /home/jupyter
 98%|████████████████████████████████████▉  | 1.42G/1.44G [00:09<00:00, 157MB/s]
100%|█████████████████████████████████████| 1.44G/1.44G [00:09<00:00, 162MB/s]
```

In [7]:
```
with zipfile.ZipFile("imdb-dataset.zip", 'r') as zip_ref:
    zip_ref.extractall(content_dir)
```

In [8]:
```
for filename in os.listdir(content_dir):
    file_path = os.path.join(content_dir, filename)
    try:
        if os.path.isfile(file_path):
            os.unlink(file_path)
    except Exception as e:
        print('Failed to delete %s. Reason: %s' % (file_path, e))
```

In [9]:
```
movie_person_useful_cols = ['tconst', 'nconst', 'category']
subset_mp = None
movie_person = pd.read_csv(os.path.join(content_dir, "title.principals.tsv/title.principals.tsv"), sep="\t", usecols=movie_person_useful_cols,
                           nrows=subset_mp)
```

In [10]:
```
person_useful_cols = ['nconst', 'primaryName', 'primaryProfession']
subset_p = None
person = pd.read_csv(os.path.join(content_dir, "name.basics.tsv/name.basics.tsv"), sep="\t", usecols=person_useful_cols,
                     nrows=subset_p)
```

Filter

In [11]:
```
# Only actor / actress
movie_person = movie_person[movie_person['category'].str.contains('(act).*')]
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: UserWarning: This pattern has match groups. To actually get the groups, use str.extract.
```

Tables match

In [12]:
```
# Match actors in movies with persons database
person = person.merge(movie_person, how='inner', on='nconst')\
               .drop(columns=['tconst', 'category'])\
               .drop_duplicates('nconst')\
               .drop_duplicates('primaryName')\
```

In [13]:
```
movie_person = movie_person.merge(person, how='inner', on='nconst')\
               .drop(columns=['primaryName','primaryProfession'])
```

Re-indexing and resize

In [14]:
```
person.reset_index(drop=True, inplace=True)

actors = person.index

# Useful dictionaries
n_person = dict(zip(person.index, person['nconst']))
person_n = dict(zip(person['nconst'], person.index))

person
```

Out[14]:

	nconst	primaryName	primaryProfession
0	nm0000001	Fred Astaire	soundtrack,actor,miscellaneous
1	nm0000002	Lauren Bacall	actress,soundtrack
2	nm0000003	Brigitte Bardot	actress,soundtrack,producer
3	nm0000004	John Belushi	actor,writer,soundtrack
4	nm0000005	Ingmar Bergman	writer,director,actor
...	...	...	...
1675599	nm9993697	Zakariya Ganim	actor
1675600	nm9993698	Sebi John	actor
1675601	nm9993699	Dani Jacob	actor
1675602	nm9993700	Sexy Angel	actress
1675603	nm9993701	Sanjai Kuriakose	actor

1675604 rows × 3 columns

In [15]:
```
movie_person.reset_index(drop=True, inplace=True)
movie_person
```

Out[15]:

	tconst	nconst	category
0	tt0000005	nm0443482	actor
1	tt0000005	nm0653042	actor
2	tt0000007	nm0179163	actor
3	tt0003116	nm0179163	actor
4	tt0003730	nm0179163	actor
...	...	...	...
14059979	tt9916756	nm10781824	actress
14059980	tt9916764	nm10538641	actor
14059981	tt9916856	nm10538650	actress
14059982	tt9916856	nm10538646	actor
14059983	tt9916856	nm10538647	actress

14059984 rows × 3 columns

Storage

In [16]:
```
with open(os.path.join(variables_dir, 'person.pkl'), 'wb') as outp:
    pickle.dump(person, outp)
```

In [17]:
```
with open(os.path.join(variables_dir, 'actors.pkl'), 'wb') as outp:
    pickle.dump(actors, outp)
```

In [18]:
```
with open(os.path.join(variables_dir, 'movie_person.pkl'), 'wb') as outp:
    pickle.dump(movie_person, outp)
```

In [19]:
```
with open(os.path.join(variables_dir, 'person_n.pkl'), 'wb') as outp:
    pickle.dump(person_n, outp)
```

In [20]:
```
with open(os.path.join(variables_dir, 'n_person.pkl'), 'wb') as outp:
    pickle.dump(n_person, outp)
```

Actor Graph

Links creation (!! heavy RAM usage ~ 15 GB !!)

In [21]:
```
def get_links(S):
  links = list()
  for i1, val1 in S.iteritems():
    for i2, val2 in S.iteritems():
      if i1 < i2:
        links.append( (person_n[val1], person_n[val2]) )
        links.append( (person_n[val2], person_n[val1]) )

  return links
```

In [22]:
```
# Group by title
grouped_mp = movie_person[['tconst', 'nconst']].groupby('tconst')
```

In [23]:
```
# Before links creation: RAM wipe
del person
del movie_person
del n_person
gc.collect()
```

Out[23]:
```
207
```

In [24]:
```
links_series = grouped_mp.agg(get_links)['nconst']
```

In [25]:
```
# Flat and unique list

links = [tup for i in range(len(links_series)) for tup in links_series[i]]
links = list(set(links))
```

In [26]:
```
del links_series
with open(os.path.join(variables_dir, 'links.pkl'), 'wb') as outp:
    pickle.dump(links, outp)
```

Graph

In [27]:
```
def get_graph(actors, links):
    g = nx.DiGraph()

    for p in actors:
        g.add_node(p)

    for (a, b) in links:
        g.add_edge(actors[a], actors[b])

    return g

def get_connection_matrix(actors, links):
    incidency = {}
    for u in range(len(actors)):
        incidency[u] = []

    for (a, b) in links:
        incidency[a].append(b)

    connection_matrix = []
    for a in incidency:
        for b in incidency[a]:
            connection_matrix.append((b, a, 1./len(incidency[a])))

    return connection_matrix
```

In [28]:
```
connection_matrix = get_connection_matrix(actors, links)
```

In [29]:
```
with open(os.path.join(variables_dir, 'connection_matrix.pkl'), 'wb') as outp:
    pickle.dump(connection_matrix, outp)
```