M222: Analysis of Social Networks
University of Athens
Deprt. of Informatics & Telecommunications

Programming Project II

Today's Date: May 1st, 2020
Due Date: June 9st, 2020

PREAMBLE

In this project, you will design, implement and demonstrate a REST API that uses a Neo4j database and allows for efficient retrieval of information related to movies.

PROBLEM DESCRIPTION:

The challenge to addressed in the context of this project is to expose information about movies, cast, crew, users and their ratings through a REST API. You will download and review raw movie data[1,2] to come up with an appropriate schema for the Labeled Property Graph model. You will then process the data to create a Graph Database than can answer related queries.

You will focus on the following files only:

- **movies_metadata.csv**: The main Movies Metadata file. Contains metadata for 45,000 movies. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
- **credits.csv**: Consists of Cast and Crew Information for all our movies.
- **ratings.csv**: Comprises the ratings of users for the movies.
- **links.csv**: Contains the TMDB and IMDB IDs of all the movies.

Using these files you will create a network schema according to the Labeled Property Graph Model. Take extra care while modeling the data as nodes, relationships and properties. Please note that you can use the *links.csv* file to find out the cast and crew members of a movie using the imdbId and the tmdbId. Similarly, you can use the same file to link a movie with its respective ratings by using the movieId and the imdbId.

IMPLEMENTATION ASPECTS:

You will use the `Neo4j` Graph Database in this project, to model and persist the data provided and build a REST API that queries this database to expose the data. You can find a simple example in Python (`hello-world-py2neo.py`) using the Neo4j database here:
`https://github.com/panagiotisl/neo4j-py2neo-vagrant-example`.

You can use the Vagrant file provided in this example to get a Neo4j database up and running or set-up your own database. The Vagrantfile pre-installs libraries py2neo and Flask which can help you with this project. However, you are free to use any programming language or library.

[1] `https://tinyurl.com/y9osu2pa`
[2] `https://www.kaggle.com/rounakbanik/the-movies-dataset`

Moreover, you have the option of limiting the data you will use in this project by following one of the approaches below:

1. Use the top-K movies with regard to their budget.
2. Use the top-K movies with regard to their user ratings.

In any case, if you do limit the number of movies that you will use, you should use at least 300 movies and you should also include 3 movies of your choice that you will specify in your report.

Your database should be able to answer the following queries that will be exposed through your API:

1. Find the movies (title, tagline, release_date) that a particular person was part of their cast.
2. Find the movies (title, tagline, release_date) a particular person directed / wrote / produced. (3 methods)
3. What characters has a particular actor embodied between a particular time range of release dates?
4. Find the original / spoken languages of movies that a particular person was part of their cast. (2 methods)
5. Find the top-K directors with regard to largest average movie runtime.
6. Find the top-K producers with regard to largest budget / revenue.
7. Find the actors that have co-acted in more than one movies released in a given single year.
8. Find those that have directed and produced a movie in a particular year.
9. Find those that have acted, directed and written a movie in a particular year.
10. Find the actors that co-acted with an actor that has acted with a given actor, but have not co-acted with the given actor.
11. Find the directors that a given actor has worked with.
12. Find the top-K directors that a given actor has not worked with, with regard to most co-operations with actors that the given actor has worked with.
13. Find the pairs of people that have directed each other in at least one movie.
14. Find the directors of consecutively released movies with more than a given amount of years between them.
15. Find the movie with the most ratings.
16. Find the movie with the best / worst average rating. (two methods)
17. Find the director with the best / worst average rating. (two methods)
18. Find the pair of actors that co-acted in movies with regard to the largest count of user ratings.

In case some of the above queries do not have any results using the data of your database you are free to create *bogus* information to test your queries.

You should use indices and constraints wherever you believe is appropriate to do so.

COOPERATION:

You do not have the option of forming a team in this project. However, discussions on most aspects of the project in the classroom's forum are encouraged.

REPORTING:

The final *typed* project report (brief report) must consist of:

1. Code for populating the database using the provided files. In case you limit the data used, this should include your three selections of movies outside the top-K.
2. Your Cypher queries.
3. Code for your API.
4. Curl commands with examples for all queries of your API (these can be easily exported from API clients such as Insomnia[3])
5. Results for each of your queries.

Finally, you will have to demonstrate your work.

[3]`https://insomnia.rest/download/`