# Georgia Tech
# CSE6242 / CS4803-DVA: Data and Visual Analytics
# Spring 2013, Polo Chau

# Homework #1, Due: January 29, 2013, 1:30PM

**Submission details:** Submit each of the deliverables as a separate attachment on the t-square submission site with the *specified file name*. In case you have collaborated with other students on this assignment, please specify the name(s) of your collaborator(s) in the text box present at the t-square submission site.

## Part 1: Collecting and visualizing Rotten Tomatoes data

1. **[30 pt]** Use the Rotten Tomatoes API to download data about movies and, for each movie its 5 related movies.
    a. Information regarding the RottenTomatoes API:
        - Register at http://developer.rottentomatoes.com/page to access the API
            - Put the course name "CSE6242" as the application name and course website (http://poloclub.gatech.edu/cse6242/) as the application website for registering the application (this seems required). However, you may choose another name or website.
        - After you have registered and got an API key, look at the documentation (http://developer.rottentomatoes.com/docs) to find out how to use the API.
    b. [4 pt] Search for movies with the keyword "**star**" and retrieve the first 300 movies.
        - The documentation for keyword search: http://developer.rottentomatoes.com/docs/read/json/v10/Movies_Search
    c. [8 pt] For each of the 300 movies, use the API to find its 5 most similar movies.
        - The documentation for obtaining similar movies: http://developer.rottentomatoes.com/docs/read/json/v10/Movie_Similar
    d. Create the following two data files, from the obtained data
        - [8 pt] **`movie_ID_name.txt`**: each line describes one of the 300 movies, in the format: <movie-ID, movie-name>
        - [10 pt] **`movie_ID_sim_movie_ID.txt`**: each line describe one pair of similar movies obtained in step c, in the format: <movie-ID, similar-movie-ID>. Note that for each movie, there should be 5 or fewer similar movies. Remove all duplicate pairs, that is, if both the pairs `(A, B)` and `(B, A)` are present, only keep one of them (you can choose either one). For example, if movie A has three similar movies X, Y and Z; and movie X has two similar movies A and B, then there should only be four lines in

the file

```
A, X
A, Y
A, Z
X, B
```

**Deliverables:**

- **Code:** you may use (1) Google Refine or (2) write your own program/script to generate the two data files that step d asks for; you can use any languages you like (e.g., shell script, python, Java);
  For either case, you must write a README file that include: (1) which approach you are using, (2) in case you write your own program/script, which platform you tested it on (e.g., Linux, Windows 8, etc.), and (3) instructions on how to run your code/script. Submit a **single zipped file**, called "`HW1.Q1.CODE.<extension>`", that contains your source code/script (unless you use Google Refine), and the **README** files.
- **Two data files**: "`movie_ID_name.txt`" and "`movie_ID_sim_movie_ID.txt`" produced in step d

2. **[20 pt]** We are going to view the data from question 1 as an undirected graph (network): each movie in **`movie_ID_name.txt`** as a node in the graph; and each movie pair in **`movie_ID_sim_movie_ID.txt`** as an undirected edge. You will visualize the graph of *similar movies* (based on the top 300 movies with the keyword ``**star**'') on Rotten Tomatoes using Gephi (available for free download at http://gephi.org).
   (*Clarification:* Please import all the edges in **`movie_ID_sim_movie_ID.txt`.** You have to check the "create missing nodes" option while importing the edges in Gephi since the many of the IDs in **`movie_ID_sim_movie_ID.txt`** will not be present in **`movie_ID_name.txt`**).
   a. Guide to getting started with Gephi: https://gephi.org/users/quick-start/
   b. [6 pt] Visualize the graph and submit a snapshot of a "visually meaningful" view of this graph. This is fairly open-ended and left to your preference. Experiment with Gephi's features, such as graph layouts, changing node size and color, edge thickness, etc.
   c. [6 pt] Using Gephi's built-in functions, compute and report the following metrics for your graph:
      - Average node degree
      - Diameter of the graph
      - Average path length
   d. [8 pt] Run Gephi's built-in PageRank algorithm on your graph
      - Submit an image showing the distribution of the PageRank score (*Clarification:* the "distribution" is the one generated by Gephi's built-in PageRank algorithm where the horizontal axis is the PageRank score and the vertical axis is the number of nodes with a specific PageRank score)
      - List the top 5 movies with the highest PageRank score

**Deliverables:**

- **Result for part b:** An image file named "`movie_graph.png`" containing the snapshot of the graph of the data you obtained from the rotten-tomatoes API.
- **Result for part c:** A text file named "`movie_graph_metrics.txt`" containing the three metrics
- **Result for part d:** An image file named "`movie_pagerank_distribution.png`" containing the distribution of the PageRank scores and append the movies with the 5 highest PageRank scores in the file "`movie_graph_metrics.txt`" from part c.

## Part 2: Using SQLite

3. **[30 pt]** Elementary database manipulation with SQLite (http://www.sqlite.org/):
   a. [1 pt] *Import data:* Create an SQLite database called **rt.db**.
   Import the movie data at
   http://poloclub.gatech.edu/cse6242/hw1/movie-name-score.txt
   into a new table (in rt.db) called **movies** with the schema:
   `movies(id integer, name text, score integer)`
   Import the movie cast data at
   http://poloclub.gatech.edu/cse6242/hw1/movie-cast.txt
   into a new table (in rt.db) called **cast** with the schema:
   `cast(movie_id integer, cast_id integer, cast_name text)`
   Provide the SQL code (and SQLite commands used).

   *Hint*: you can use SQLite's built-in feature to import CSV (comma-separated values) files: http://www.sqlite.org/cvstrac/wiki?p=ImportingFiles

   b. [3 pt] *Find cast size:* How many distinct cast members does this database contain? Provide the SQL code.

   c. [3 pt] *Critically acclaimed movies:* How many movies have '`score`' greater than 90? Provide the SQL code.

   d. [4 pt] *Prolific cast members*: List the top 5 cast members with the highest number of movies appearances. Provide the SQL code.

   e. [6 pt] *Successful cast members*: Using the two tables above, list the top 5 cast members with the highest average scores. Provide the SQL code. A cast member's score for a movie is the score of the movie he/she was in and the average is over all the movies they were involved with.

   f. [7 pt] Create a view (virtual table) called `` `co_cast_members` `` that contains one row for each pair of cast members. The view should have the format

`(cast_member_1_id, cast_member_2_id)`. Exclude self-pairs `(cast_member_1_id == cast_member_2_id)` but keep mirror pairs (for example, `(A, B)` and `(B, A)` should both be in the set). Provide the SQL code.

g. [6 pt] (Collaborator) List the names and distinct `co_cast_member` counts for the top 5 cast members with the highest number of co-cast members. Provide the SQL code.

**Deliverables:**

- **Code:** A text file named "`HW1.Q3.SQL.txt`" containing all the SQL commands and queries you have used to answer questions a-f in the appropriate sequence. We will test its correctness in the following way:
  - `$ sqlite3 rt.db < HW1.Q3.SQL.txt`

  We will assume that the data files are present in the current directory.
- **Answers:** A text file named "`HW1.Q3.OUT.txt`" containing the answers of the questions a - f. This file should be created in the following way:
  - `$ sqlite3 rt.db < HW1.Q3.SQL.txt > HW1.Q3.OUT.txt`

  We will compare your submitted text file to the text created in the above manner.