CSE 515 Multimedia and Web Databases

Phase #1

(Due Sept 17th 2017, midnight)

Description: In this project, you will experiment with

- vector models and
- graph models.

This project phase will be performed by each group member; but, you will get group grades. **We will randomly select two submissions for each group for grading**.

- You will be provided with sample MovieLens+IMDB data, including
  - mlmovies(movieId, movieName, genres)
  - mltags(userId,movieId,tagid,timestamp)
  - mlratings(movieId,userId,imdbId,rating,timestamp)
  - genome-tags(tagId, tag)
  - movie-actor (movieId, actorId, actorMovieRank)
  - imdb-actor-info (actorId, name, gender)
  - mlusers(userId)

  in the form of a CSV file.
- Download the MovieLens+IMDB data data. You are free to store the data in a relational database (such as MySql) or create an in-memory data structure to store the provided network.
- **Task 1:** Implement a program which considers all the movies an actor played and creates and stores a weighted tag vector for each actor using (time-weighted) TF as well as TF-IDF models. When combining tag vectors under TF or TF-IDF models, newer movies and newer tags should be given higher weight than old movies and older tags. Similarly, movies where the given actor appears with a lower rank should be given a relatively higher weight.

  Also create a command line interface

  `print_actor_vector` *actorId model*

  which prints the tag vector (as a sequence of $\langle tag, weight \rangle$ pairs, sorted in descending order of weights) for the given actor under the given vector model.
- **Task 2:** Implement a program which considers all movies of a given genre to create a combined tag vector for the genre. When combining tag vectors under TF or TF-IDF models, newer movies and tags should be given higher weight than old movies and tags.

  Create a command line interface

  `print_genre_vector` *genre model*

  which prints the tag vector for a given genre under the given vector model.

- **Task 3:** Implement a program which considers all movies watched by a user to create a combined tag vector for the user. When combining tag vectors under TF or TF-IDF models, newer movies and tags should be given higher weight than old movies and tags.

  Create a command line interface

  `print_user_vector` *userId model*

  which prints the tag vector for a given genre under the given vector model.

- **Task 4:** Implement a program which considers two genres, g_1 and g_2, and explains in what ways the g_1 *differs* from g_2. You will consider three models, TF-IDF-DIFF, P-DIFF1, and P-DIFF2:

  - In the TF-IDF-DIFF model, you will consider the set of movies for the given genre to compute TF, but the set, $movies(g_1) \cup movies(g_2)$, of all movies in genres, g_1 and g_2, to compute IDF.
  - In the P-DIFF1 model, you will identify the weight, $w_{i,j}$, of the tag t_j for genre g_1 relying on a probabilistic feedback mechanism (as will be discused in the class later – See Section 12.4):

    $$w_{1,j} = log \frac{r_{1,j}/(R - r_{1,j})}{(m_{1,j} - r_{1,j})/(M - m_{1,j} - R + r_{1,j})} \times \left| \frac{r_{1,j}}{R} - \frac{m_{1,j} - r_{1,j}}{M - R} \right|,$$

    where:

    - $r_{1,j}$ is the number of movies in genre, g_1, containing the tag t_j
    - $m_{1,j}$ is the number of movies in genre, g_2, containing the tag t_j
    - $R = \|movies(g_1)\|$, and
    - $M = \|movies(g_1) \cup movies(g_2)\|$.
  - In the P-DIFF2 model, you will identify the weight, $w_{i,j}$, of the tag t_j for genre g_1 relying on a probabilistic feedback mechanism (as will be discused in the class later):

    $$w_{1,j} = log \frac{r_{1,j}/(R - r_{1,j})}{(m_{1,j} - r_{1,j})/(M - m_{1,j} - R + r_{1,j})} \times \left| \frac{r_{1,j}}{R} - \frac{m_{1,j} - r_{1,j}}{M - R} \right|,$$

    where:

    - $r_{1,j}$ is the number of movies in genre, g_2, not containing the tag t_j
    - $m_{1,j}$ is the number of movies in genres, g_1 or g_2, not containing the tag t_j
    - $R = \|movies(g_2)\|$, and
    - $M = \|movies(g_1) \cup movies(g_2)\|$.

  Create a command line interface

  `differentiate_genre` *genre1* genre2 *model*

  which prints the differentiating tag vector for a given pair of genres under the given vector model.

Deliverables:

- Your code (properly commented) and a README file.
- Your outputs for the provided sample inputs.
- A short report describing your work and the results.

Please place your code in a directory titled "Code", the outputs to a directory called "Outputs", and your report in a directory called "Report"; zip or tar all off them together and submit it through the digital dropbox.