CSE 515 Multimedia and Web Databases

Phase #3

(Due November 29^{th}, midnight)

Description: In this project, you will experiment with

- vector models and
- graph models.

This project phase will be performed as a group. You will be provided with sample MovieLens+IMDB data in the form of CSV files. 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. For PCA, SVD, LDA, and CP decomposition you can use existing packages.

- **Task 1:** Implement a program which given all the information available about the sequence of movies a given user has watched, recommends the user 5 more movies to watch,
  - **Task 1a:** using SVD or PCA.
  - **Task 1b:** using LDA.
  - **Task 1c:** using tensor decomposition.
  - **Task 1d:** using Personalized PageRank.
  - **Task 1e:** using a measure that combines all the above.

  The order in which the movies are watched and the recency shoudl also be taken into account.

  The result interface should also allow the user to provide positive and/or negative feedback for the ranked results returned by the system to enable Task 2.
- **Task 2: Relevance feedback task (content):** Implement a probabilistic relevance feedback system to improve the accuracy of the matches from Tasks 1a through 1e. The system should also output the revisions it suggests. See
  - Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science. 41, pp. 288-297, 1990.

  User feedback is than taken into account (either by revising the query or by re-ordering the results as appropriate) and a new set of ranked results are returned.
- **Task 3: Multi-dimensional index structures and nearest neighbor search task:**
  - Implement a program which maps each movie in the system into a 500 dimensional latent space.
  - Implement a Locality Sensitive Hashing (LSH) tool, which takes as input (a) the number of layers, L, (b) the number of hashes per layer, k, and (c) a set of movie vectors as input and creates an in-memory index structure containing the given set of vectors. See

    "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions" (by Alexandr Andoni and Piotr Indyk). Communications of the ACM, vol. 51, no. 1, 2008, pp. 117-122.

  - Implement similar movie search using this index structure: for a given movie and r, outputs the r most similar movies (also outputs the numbers of unique and overall number of movies considered).

    The result interface should also allow the movie to provide positive and/or negative feedback for the returned movie returned by the system to enable Task 4.

- **Task 4: NN-based relevance feedback:** Implement a r-nearest neighbor based relevance feedback algorithm to improve the nearest neighbor matches. The system should output the revisions it suggests in the relative importances of different parts of the query.
- **Task 5: Movie classification:** Implement
  - a r-nearest neighbor based classification algorithm
  - a decision tree based classification algorithm, and
  - an n-ary SVM based classification algorithm

  which takes a set of labeled movies and associates a label to the rest of the movies in the database.

Every result should be presented in decreasing order of weights!

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.