YONATHAN KIDANEMARIAM
HRISHI MUKHERJEE
COMP 4601
ASSIGNMENT 3
Due: March 21st, 2017

Analysis

1. Migrate Data to Mongo

First, the raw data was extracted from the archive, including: Users, Movie, and Reviews. The raw data was then migrated, and used to create the custom models, including: User Model, Movie Model, and Review Model. These models were stored in Mongo, a collection for each. The archive given came with a dataset of movie reviews along with the users making reviews, therefore it seemed appropriate to create models for each of dataset.

User Model - Name, Optimism, Reliability

Movie Model - Name

Review Model - UserName, MovieName, Score, Helpfulness

2. Feature Computation

After thoroughly analyzing the datasets, a decision was made to use the scores and helpfulness of the users to create the recommender system. The Score and Helpfulness values, are accurate preferences made personally by the users, therefore it made sense using values that were personal decisions made by the users themselves. This way an accurate recommendation of advertising can be made and displayed to each user.

In the dataset provided, "real" users have made reviews on a list of "real" movies. When a user reviews a movie, he gives a Score for how much they liked/disliked the movie. Also, if other users find a review another user made was helpful, they can give a thumbs-up or thumbs-down rating for the Helpfulness of the review.

The features chosen for the user model, used the Scores that the user gave on each movie they reviewed, and the helpfulness rating they received, from other users. For every review a user has made, the Scores value and Helpfulness value were extracted, and combined to create a total Score and a total Helpfulness value. These values were then divided by the total number of reviews the user has made, to calculate their average Score and average Helpfulness. The average values for each field, were used as the FEATURES for each user.

Features:

Average Score -> Optimism

Average Helpfulness -> Reliability

3. Determining Communities

The communities were created using the K-means algorithm. Four clusters were made for each possible outcome that a cluster of users can have. Below are the four possible outcomes.

Cluster Outcomes (explained later):

Optimism -> HIGH && Reliability -> HIGH

Optimism -> LOW && Reliability -> LOW

Optimism -> HIGH && Reliability -> LOW

Optimism -> LOW && Reliability -> HIGH

Using the features, Optimism and Reliability, the users were placed into their respective clusters. Using the K-means algorithm the clusters were found. A Cluster model was created once the clusters had been formed. Each cluster was stored in mongo, in Cluster Collection. Also, a UserCluster model was created, which stores a user id and the cluster they belong to. This was also stored as a collection in the mongo database.

4. Classification of Communities

Next, the communities were classified, using a custom classifier. The average Optimism and average Reliability of each community (or cluster) was calculated. The averages are then compared to a custom threshold for each feature. The threshold places a cluster in a specific category. If the cluster's average Optimism is below the threshold, it is given a LOW ranking. However, if the clusters average is above they are given a HIGH ranking. If the clusters average Reliability is below or above the threshold, they are also given HIGH or LOW classification as before.

5. Creating Advertisements

The advertisement was created based on the classification of the communities. A community has two fields Optimism and Reliability. Based on where a community fell on the threshold, it is given either a HIGH or LOW ranking it the respective category. The advertisements were made fitting the ranking of each field. For example, for HIGH Optimism, a set of advertisements were created for highly optimistic users, such as a Job opportunity, or a Blockbuster movie. So, if an employer is looking for positive people to fill an empty position, the employer can use advertise content to that specific type of user. Each

advertisement created is stored in a database, to be retrieved when necessary. This is done for each field. Therefore, when the web service /fetch/{user}/{page} is called it will take the user, find the community it belongs to, and retrieve the HIGH or LOW ranking of the Optimistic and Reliable fields. Using the rankings, it will augment different advertisements on the specified page, according to the ranking of the user's community.

Advertising Content:

High Reliability:

-> Management Role Hire

-> Police Force Hire

Low Reliability:

-> Reliability Training

High Optimism:

-> Indian James Bond

-> Sham Wow

-> Hike Fuji

Low Optimism:

-> Adopt Pet

Communities Found After Clustering:

(Note: The number of communities remains the same, but their properties vary every time the clustering k-means algorithm is run)

Community 1:

-> Optimism: LOW -> Reliability: LOW

Community 2:

-> Optimism: HIGH -> Reliability: LOW

Community 3:

-> Optimism: HIGH -> Reliability: HIGH

Community 4:

-> Optimism: HIGH -> Reliability: HIGH

SUGGEST ALGORITHM

Assumptions:

- Page is a movie
- The New User has connections within the graph
- Communities have been determined
- Optimism and Reliability of the users have been computed
- Community specific advertisements have been created

Suggest Algorithm Overview:

The Suggest Algorithm has two components to it as follows.

Suggest Pages

- This is a two-part algorithm, taking a New User as input, and outputs the list of suggested pages. The first part of the algorithm calculates the average score of ALL movies, and retrieves the top five. The second part of this algorithm finds the top five movies reviewed by every connected user. It then combines the two top five lists, and returns a Top 10 suggested movies for the new user.

Advertisement

- This algorithm takes a New User as input, and outputs a list of Advertisements. It achieves this by finding the most recurring cluster id(community) of the new user's connections in the graph. Whichever, community is dominant in the new user's connections, the corresponding advertisements will be outputted from the algorithm.

Suggest Algorithm

Suggest Pages

Input: New User, U
Output: List of Suggested Pages

For Each Movie
  | Calculate Average Review Score

All_Movies <-- Retrieve Top 5 Movies

For Each Connection C of New User
  | Find Top 3 Reviews Made By C
  | Top_Reviews <-- Add Top 3

Connected_Movies <-- Retrieve Top 5 of Top_Reviews

Create New List Suggested Movies
Suggested Movies <-- **All_Movies + Connected_Movies**

Output Suggested Movies

----------X

Advertise

Input: New User, U
Output: List of Advertisements

For Each Connection C of New User
  | C_ID <-- Retrieve Cluster Id
  | Add C_ID to Cluster_Id List

Recurring Cluster <-- Find Most Recurring Element of Cluster_Id

Output Recurring Cluster's Advertisements