COMP4601 Avery Vine (100999500) and Maxim Kuzmenko (101002578)

Assignment 2 Analysis Document

How we computed **user profiles**:

User profiles contains basic information such as a unique id, the URL from which they were created, a name, and a list of web pages (movies) that they have reviewed. We also included the following attributes:

- preferredGenre: This attribute represents the preferred genre of the user, which is obtained by getting the sentiments of all of the reviews that the user has written, sorted by genre, and finding the genre with the highest score. More under "Communities".
- sentimentScores: This attribute is a list that contains a user's sentiment towards each genre. If the sentiment score for a genre is < 0, then the user tends to dislike that particular genre. If the sentiment score for a genre is > 0, then the user tends to like that particular genre. A sentiment score of 0 indicates that the user is neutral -- typically meaning that the user has not written a review on a a single page belonging to that genre. More under "Communities".

How **/reset/{dir}** works:
/reset/{dir} crawls the data for users and pages under directory {dir}, stores that data in the database, and then analyzes the genres for all the webpages. We chose to use the Naive Bayes algorithm to classify web pages into genres, based off of a few training articles that we hand-selected and classified ourselves. We originally tried using Latent Dirichlet Allocation to determine genre, but the size of the libsvm file generated by the analysis of the webpages was so massive that it was causing issues during runtime, so we were forced to switch to the Naive Bayes implementation.

How **/context** works:
/context calculates the preferred genre of each user in the database. This is done by getting the reviews on any given webpage and the user corresponding to each of those reviews, then assigning each user a sentiment score for each review. Once again, we used the Naive Bayes algorithm, this time to generate a number representing how negative or positive their review is. At the end, each user's sentiment scores are averaged out for each genre and are stored as sentimentScores. The genre corresponding to maximum value in the sentiment scores becomes the user's preferred genre. The HTML page returned is an HTML profile of the users, which shows the user ID, Name, URL, Preferred genre, and sentiment values for each genre.

How **/fetch/{user}/{page}** works:

/fetch/{user}/{page} retrieves and displays a web page, augmented with advertising related to both the web page in question and the user who would be hypothetically accessing the page. More specifically:

1. Advertisements related to the genre the user viewing the page prefers
2. The genre of the movie corresponding to the webpage being visited.

Each advertisement shown is a div containing the genre of the hypothetical advertisement being displayed, and is located at the very top of the page. The regular content from the web page in question follows the advertisements.

It is important to note that there is a slight disconnect between the fetch functionality on the client and the /fetch/{user}/{page} REST endpoint. When the client's fetch functionality is invoked, a list of all of the web pages is retrieved from the /fetch REST endpoint. When one of those web pages is clicked, it prompts the person using the client to enter the name of the user who would be hypothetically accessing the web page. Finally, the /fetch/{user}/{page} REST endpoint is invoked.

Communities:
Since the webpages represent movies and movie reviews, we decided to create communities of users based off of genre. This means that there would be a community for every defined genre. To group users into these communities, all that needs to be done is finding a user's preferred genre, do this for all users, and then group these users into communities based on their preferred genre. This was done by first calculating the genres of all webpages/movies, and then using that data to help calculate a user's preferred genre. In our implementation, we used the sentiment analyzer for two essential components:

1. Calculating user's preferred genre
    - To calculate the preferred genre, we used the sentiment analyzer with two classifications, positive and negative. The training data used were many files with movie reviews that were located in two directories (in /training/negative/ (negative reviews) and in /training/positive/ (positive reviews). After calculating posterior probabilities, the positive and negative probabilities were stored as one number representing user's review sentiment. The higher the number, the more positive that review is seemed, and vice-versa.
    - After looping through all of the webpages and gathering all of the review scores for all of the users, the preferred genre for the user is calculated by finding the genre as a string associated to the maximum value of the average of all of the review scores for every genre for the user. This is done in SentimentAnalyzer.java.
2. Calculating the genre of the movie associated to the webpage
    - Finding the genre of the movie was very similar to calculating the user's preferred genre. We used the genre analyzer with five classifications, with each classification representing a genre ("comedy", "thriller", "romance", "action", "drama"). The training data used were five folders labelled with the name of the genre, with each folder containing three html files of the webpage in which we predetermined the genre.

Finally, after calculating the posterior probabilities, the genre associated to the maximum posterior probability in the set of five posterior probabilities is then set as the genre of the webpage/movie. This is done in GenreAnalyzer.java.

Stats:

Category stats using the training data:

- Total # of users crawled: 1252
- Total # of web pages crawled: 1079

	Users		Page	
Community (Genre)	# of Users in Community	% of the total users	# of Web Pages with corresponding genre	% of the total Web Pages
Comedy	49	3.91%	60	5.55%
Thriller	309	24.68%	534	49.49%
Romance	194	15.49%	107	9.92%
Action	166	13.26%	172	15.94%
Drama	534	42.65%	206	19.09%

How **/advertising/{category}** works:

Our advertising content is represented as just a div that includes the genre name. When navigating to /advertising/{category}, if {category} does not match one of the predefined genres ("comedy", "thriller", "romance", "action", "drama"), we return an invalid genre message, otherwise the page returns the valid genre.

How **/suggest** would work:

Because we store individual review scores for each user, the SUGGEST functionality will not be difficult to implement.

For a given user u_a from the collection of users U, we can determine provide a way to suggest pages to user u_a based off the preferences of u_a's friends, known as users u_0, u_1, ... , u_x. The steps of algorithm would be as follows:

1. For each user u_i from u_0 to u_x on user u_a's friend list:
    a. Retrieve u_i's favourite genre
    b. For m_j from $j = 0 \ldots 2$, retrieve u_i's movies from the movie collection M_i based off sentiment score, and add them to u_a's collection of movies, M_a
2. From the collection of movies M_a, retrieve the top three movies based off sentiment score to make a new collection of movies, M_b

3. Suggest the movies from M_b to the user u_a

After implementing this algorithm, the advertising system could now augment a user u_a's advertisements based off the preferred genres of the users that user u_a follows.