COMP4601 - Assignment 2

Alexander Nguyen (100908039)
Redwan Wadud (100873111)

04/05/2020

System Documentation

CommunityAnalyzer

The purpose of this class is to analyze the user feature data and group users into clusters. Each resulting cluster is in sense a community. We have limited the number of clusters to 3. This gives us 3 communities of users. The Kmeans algorithm is used to generate the clusters. The feature used in the calculations is the user's preference towards a particular genre (action, comedy, horror). The resulting clusters are somewhat accurate of the user's preferred genre.

PreferenceAnalyzer

The purpose of this class is to determine the genre preference of a user. For each user, the sentiment of each review is taken into account towards the analysis. If the sentiment of the review is positive, a user is then given preference towards the genre of the movie. The highest preference count towards a genre is then determined to be the users preferred genre.

SentimentAnalyzer

Using the Stanford corenlp library, this class analyzes a review and returns its sentiment scores.

CorpusParser

This is a modified version of Prof. White's CorpusParser. This class is used to determine the genre of movies. Each review is analyzed and the term frequency of 3 predefined genres (action, comedy, horror) is used to assign movies a genre. Some keywords are used to assist in better detecting a genre. For example, scary is used to detect the horror genre and funny is used to detect comedy.

DataLoader

This class is used to download and extract archived data, and build the database. The data is downloaded from Prof White’s server and extracted into a temp location. It is crucial to have the data placed in a temporary location outside the eclipse workspace directory. If it is not, it may cause eclipse and the web server to lag and be unresponsive oftentimes.

Kmeans

The purpose of this class is to classify the given data set into multiple clusters based on the feature calculations. It is used to group users into clusters to form a community.

/context

A RESTful web service that utilizes GET to analyze web pages and return an HTML representation of user profiles. The HTML presents the features that are being used as part of the profile for a user.

/community

A RESTful web service that utilizes GET to return an HTML representation of the communities computed for the users. The HTML contains a table in which each row contains the community name (C-X, X = 1, ..., *m*) followed by a cell containing a comma delimited list of user names.

Should the community service be run before the context service an error will be generated.

/fetch/{user/{page}

A RESTful web service that utilizes GET to retrieve a page from the context and augments it with advertising. The HTML contains two frames, one containing the content of the requested page, and the other containing advertising.

/advertising/{category}

A RESTful web service that utilizes GET that returns an HTML representation of the advertising category. Categories are C-X, X = 1, ..., *m*. There is one advertising category for each user community.

Advertisements are stored on a remote repository and are all accessible through endpoints using html string tags (img scr=***).

There are 3 ads per category, the category options are: **action**/ **horror**/ **comedy**.

Advertisements shown are actual movies from the database, a link will be provided along with the advertisement image for users to view the recommended movie page.

Suggest Algorithm

Based on the data we have, it is very easy to generate an algorithm to suggest content from one user to another. Each user has a collection of reviews and each review is attached a rating towards the movie. Using the social graph, the algorithm can use the “friends” of a user to calculate the suggestions. Using a collaborative user based filtering algorithm this can be achieved with ease.

For each user that are friends, calculate their similarity and store it in a matrix.
Calculate the predictions from one user to another using the similarity matrix.
Create a list for each user to store the suggested movies.
Using the predictions matrix, take the highly predicted scores of movies and add them to the users suggestion list.

Now when a user visits a page, a movie is taken from the users suggestion list and presented as an advertisement instead of the standard method of having predefined categories. This method would be highly effective as users would more likely click the suggested movie since it is tailored to be in the users interest. Whereas showing a random movie/page based on a broad community/category would not be much of interest to the user.

When a user has watched a suggested movie from the suggestion list, it would then be removed and the list would have to be recalculated to prevent users from being suggested the same movie repeatedly.