TV Recommender System

Aarushi Karnany - 2012004, Pulkit Arora - 2012082

16th April 2015

1 Introduction

Today, most of the consumers go through an exhaustive task of finding TV shows that might be of interest to them. The gradual increase in the options available has made this task even more tedious. To automate this task of finding shows that might be of user's interests, we try to use some machine learning techniques and build a tv recommender system for users.

2 Dataset description

The dataset used in testing and training had a collection of approximately eleven thousand anonymous facebook user profiles. The profile data included hashed user id, gender, location, about, tv shows liked, music liked, movies liked, books liked, activities, interests and sports of the user. This profile data has a unary rating system(that is, we have the information only about what the user likes, and not about what he doesn't) for TV shows, movies, books and music that is liked by the user. Either the user likes it, or the user dislikes or ignores it. We used a 1 to indicate that the user likes this show or movie, while a 0 to indicate that there is absence of liking for this show.

3 Pre-processing

To start with, we first normalised the data by converting all the data into lower case character. We then used stop words removal, so that "The Big Bang Theory" and "Big Bang Theory" are considered to be the same show.

We then used stemming, so that the the titles “Game of thrones” and “Game of throne” are not considered as different tv shows.

4 Evaluation Metric

Usually recommender systems are evaluated on the basis of RMSE, the Root Mean Squared Error. However, this evaluation is better for systems in which users rate items (TV shows, movies, books, etc.) in a certain discrete range, e.g., on a scale from one to five stars, in Amazon or ImDb etc. When Facebook data is considered, ratings are unary: either a user likes an item (thereby making a positive association), or not. The absence of a like can be taken either as a dislike or ignorance. We therefore chose to use the Precision–Recall metrics because the unary rating scale makes the task of recommending TV shows more like a classification problem. We define Precision and Recall as follows

$$\text{Precision} = \frac{Number\ of\ recommended\ shows\ that\ user\ likes}{Total\ number\ of\ recommendations}$$

$$\text{Recall} = \frac{Number\ of\ recommended\ shows\ that\ user\ likes}{Total\ number\ of\ shows\ that\ user\ actually\ likes}$$

5 Classifiers used

5.1 Genre Based Prediction

In this approach, we use the fact that a user might like TV Shows based on his/her liking of certain genre. In this approach, we make the recommendations to a user based on the genre he likes the most. The data was divided such that 50% of the likes of each user was taken as training data, and the rest as testing data. From the training data, we first calculate the probability of user liking a genre. We then use this probability distribution to calculate the score for a TV show S for a user U, using the formula given below. Score of a user U for a show is the summation of probability of U liking genre G for each genre G that the TV show S belongs to.

$$\text{Score(S, U)} = \Sigma P(G, U) \forall G \in S$$

5.2 K-means clustering

Our next approach was running the K-Means clustering algorithm to identify clusters of similar users in the data. The users were selected in such a way that 70% of the users that had liked at least two TV shows were part of the training data set. The rest of the 30% who had liked at least two TV shows were part of the testing data set and only half of their ratings were used. The other half was removed. We varied the number of clusters K, which gave us different results. Using the standard iterative implementation of the algorithm, we were able to partition the data into eight clusters. Thereafter, different values of K ranging from eight to hundred were tried and results were obtained accordingly. Recommendations made to a test user were those that others in the same cluster had collectively liked the most.

5.3 Collaborative Filtering

5.3.1 User based

In this approach, we make use of the fact that users with similar interests tend to have more shows in common than otherwise. Here, we used the k nearest neighbours algorithm to define a neighborhood of users similar to the user who will be recommended TV shows. The neighborhood is defined on the basis of the cosine similarity measure, which is defined as follows:

$$\cos(\text{u1, u2}) = \frac{u1.u2}{|u1||u2|}$$

where u1.u2 represents the common likes of the users and $\|u1\|and\|u2\|$ represents the number of likes by user u1 and u2 respectively. The vectors u1 and u2 represent the user's likes and dislikes in the unary-based system. If the similarity measure is high, it means that those two users are more likely to watch similar interest of shows.

5.3.2 Item based

This approach is similar to the user-based one. The only difference being, instead of using similarities between users, we use similarities between different items. The fact employed in this approach is that if two items are liked by similar set of users, they tend to be similar, and the users are expected to have similar likeness towards similar set of items.

5.4 Content based filtering

This approach aims at finding the similarity scores between users using the entire profile information of the users, and using these similarity scores to recommend TV shows to a user. In this we used the pre-processed data to first generate profiles based on all the information available to us through user's facebook profile. We then computed the tf-idf scores for the all the training data which was 70% of the dataset. The rest of the 30% of the data set was used as the testing data, for each of which ten similar users were calculated based on the cosine similarity measure. Recommendations were made from the set of TV shows liked by those ten similar users. We used only those users who had liked at least one TV show, to generate better results.

6 Results

The results obtained from different approaches are shown in the figure below:

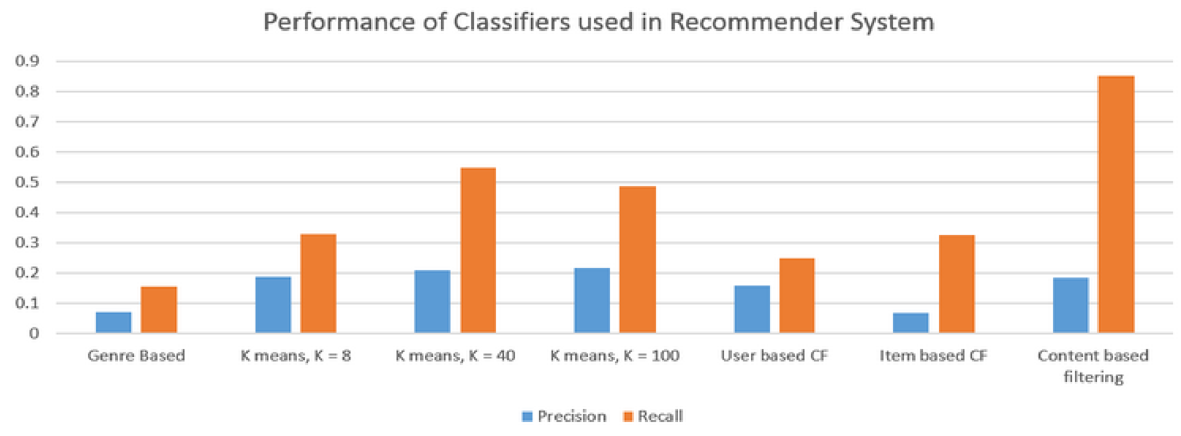

Figure 1: Results

References

[1] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms.

[2] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[3] http://infolab.stanford.edu/ ullman/mmds/ch9.pdf

[4] http://ls13-www.cs.uni-dortmund.de/homepage/ITWP2010/slides/semeraro.pdf

[5] http://homepages.abdn.ac.uk/advaith/pages/teaching/abdn.only/AIS/lectures/abdn.only/Col