A Movie Recommendation System

Rasmus Munk

April 21, 2017

1 Approach

The user-based collaborative filtering [7] approach was selected to make movie recommendation for a set of users. This meant that recommendations for an individual user would be based on what similar users have liked that the current user have not yet rated/seen. This is based on the theory that users with similar taste are likely to provide good recommendations to each other. The key here lies in how is similarity between users determined, in this instance the Pearson correlation coefficient measurement [9] was selected as the primary method to evaluate similarity, it measures similarity of users by making a linear correlation between what user x and y have rated in terms the score they applied to a set that contains the intersection of movies rated by x and y. The inspiration for this approach was drawn from [8] where this was applied in conjunction with a weighted similarity score that makes sure that users you agree most with are rated higher in terms of movie recommendations.

This technology was selected based on multiple factors. This included the suggestions made by [2] about movie recommendation systems and their domain features including the low risk value of the recommendation, the homogeneous nature of the system and the stability in terms of preference that users have in terms of how much can you rely on previous received information e.g. user's rating history to make future predictions. In terms of technology variation, collaborative filtering provides multiple approaches, e.g. user-based or item based filtering [10]. Both of these have merits, the user-based approach increases in computational cost as the number of users increases. The item based however grows whenever new items are added. In this instance the number of users/items is stable since it is not a live system a choice was made to use the user based approach since the dataset contains 671 users and 9125 movies.

2 Required Knowledge

The knowledge required for this approach to work include user histories of movie ratings from which user similarity can be based upon. To achieve this a number of public available datasets containing user movie ratings were evaluated. This included the Netflix Prize dataset [1] , the IMDb academic dataset [4] and the MovieLens [3] dataset. From these the MovieLens 100K dataset was chosen as being the most appropriate. The reason for this was that the original Netflix dataset only contained data about when a user rated a movie and not how it was rated. The reason for this was that it was the task for the algorithm to predict the number of reviews and not the actual rating a movie would receive in 2006. Similar to this the IMDb dataset contained rating of movie's but they weren't tied to any users, meaning that user-based collaborative filtering wasn't feasible. The MovieLens dataset on the other hand doesn't have any of these missing features, meaning that it contained users which had committed at least 20 ratings with a value between 1.0 and 5.0 to different movies. Because of this the MovieLens dataset was selected as the most appropriate to develop this movie recommendation system.

However it was discovered later in the process that there has been developed an altered version of the Netflix dataset which provides actual ratings of movies [6], this was discovered after the MovieLens dataset had been applied.

3 Algorithm

The pseudo code for the similarity and recommendation functions can be seen in Algorithm 1 and 2. The similarity function uses Pearson correlation coefficient to determine user similarity on a shared set of movies. The similarity is calculated by basically drawing a line through the rating values of the shared movies. If two users agree on the rating of their shared set of movies the result would be a positive correlation between them, meaning that they are expected to have similar taste. After the similarity has been calculated the recommendation function is run 2. This function finds movie recommendations for each user by ranking the unseen movies that similar users have seen. The proposed rating for the unseen movies is proportional to how similar the user who has seen is to the current user, i.e. *weighted_sum[rating.movie_id] + = [rating.value * similarity].* After this has been completed for every user the proposed movies are ranked based on their estimated rating.

Upon completion the correlation is stored in the user/user *df_similarities* matrix and later persisted in the underlying database for later retrieval when a user requests movie recommendations. See [5] for the full implementation.

Algorithm 1 Recommender.sim_pearsons()

```
$df\_movies\_rated[user] \leftarrow$ Series containing movies rated by user
$df\_similarities \leftarrow$ DataFrame consisting of user Id's on the x and y axis
for all user_x in users do
  for all user_y in users do
    if df_similarities[user_x][user_y] is null and user_x != user_y then
      $intersection \leftarrow$ shared movies between user_x and user_y
      if $intersection \geq 5$ then
        $x\_ratings \leftarrow$ list of user_x rating values
        $y\_ratings \leftarrow$ list of user_y rating values
        $pearson \leftarrow personr(x\_ratings, y\_ratings)$
        $df\_similarities[user\_x][user\_y] \leftarrow pearson$
        $df\_similarities[user\_y][user\_x] \leftarrow pearson$
      end if
    end if
  end for
end for
$Recommender.users\_ratings\_similarities \leftarrow df\_similarities$
```

Algorithm 2 Recommender.recommendations() Page 16 [8]

```
$weighted\_sum \leftarrow \{\}$
$similarity\_sum \leftarrow \{\}$
$num\_rows \leftarrow length(users)$
for all user in users do
  $df\_similar\_users \leftarrow Recommender.users\_ratings\_similarities[user.id]$
  $df\_similar\_users \leftarrow$ Remove users that aren't similar.
  $user\_movies \leftarrow$ user rated movie ids
  for all target_user_id, similarity in df_similar_users do
    if $similarity$>0 then
      for all rating in users[target_user_id].ratings do
        if rating.movie_id not in user_movies then
          $weighted\_sum.setdefault(rating.movie, 0)$
          $weighted\_sum[rating.movie\_id] += (rating.value * similarity)$
          $similarity\_sum.setdefault(rating.movie\_id, 0)$
          $similarity\_sum[rating.movie_id] += similarity$
        end if
      end for
    end if
  end for
  Rank the ratings, users who agree alot with the current user will have their ratings affect the final score the most
  $rankings \leftarrow [(w\_score/similarity\_sum[movie\_id], movie\_id)$
      $for\ movie\_id,\ w\_score\ in\ weighted\_sum.items()]$
  $rankings.sort()$
  $rankings.reverse()$
  $Recommender.recoms[user] \leftarrow rankings$
end for
```

4 Evaluation & Reflection

To evaluate whether the recommendations made by the correlation filter was actually usable to the users a series of offline tests were conducted. This included randomly splitting the set of user ratings into a training and testing set in of either 60/40 or 80/20 distribution. This was then used to apply the training dataset to the recommendation engine to evaluate whether the engine would be accurately be able to predict the rating contained in the test data Beyond this the amount of minimum shared rated movies were also varied between either above 0 or equal to and above 5. The hypothesis was that by sharing more movie ratings the algorithm would be able to better predict the missing ratings.

To evaluate the precision of the recommendation predictions the root mean squared error(RMSE) was calculated to determine the accuracy of the predicted ratings. The reason for this was that as described in [7] the RMSE emphasizes large errors and penalizes the performance value by how far each individual prediction is from the true value/rating.

The results of the testing can be seen in 1. The results are from running the recommendation test 20 times for each configuration on the MovieLens 100K dataset and extracting the top 100 recommendations. The reason for multiple runs is that because the testing data is a randomly selected sample that by chance could extract a very good set of users. i.e they have rated more movies than the average.

As Table 1 shows the mean RMSE improves when the number of required shared movies is increased. T-tests at the 1% level also shows that there is a significant difference in the results gathered i.e. (p-value = 0,00004) for the 80/20 configurations and (p-value = 4.43e-13) for the 60/40's. However, when looking the percentage between the training/test distributions it indicates no significant difference in the achieved predictions with (p-value = 0.58) between the 60/40 1 or more shared movies vs the 80/20's configuration or (p-value = 0.13) with 60/40 5 or more shared movies vs the 80/20's.

Table 1: Testing Results

Train %	Test %	Shared Movies	Mean RMSE
60	40	1 or more	1,58840
60	40	5 or more	1,23029
80	20	1 or more	1,54424
80	20	5 or more	1,29246

Reflecting on these results it shows that in terms of the configuration parameters for the user-based collaborative filtering approach using the Pearson correlation coefficient similarity function the results supports the proposed hypothesis. I.e that an increase in shared movies has a significant impact on the RMSE with a lower prediction average which means that the prediction ability of the system improves. Whether the recommended movies are actually engaging and interesting to the users is another matter and can't be claimed based on these results.

In terms of overall drawbacks of the user-based approach, whenever a user adds a new rating that users recommendations has to be recalculated. This could pose a significant query time whenever a user attempts to retrieve recommendations. To mitigate this the current recommendations are pre-calculated when the system is launched, however this does mean that if a user rates a new item it won't be reflected in the recommendations before the system is relaunched. To counter this in general the item-based approach does seem preferable in that the similarity of each item can be precomputed, hence whenever a user commits a new rating the recommendations can be updated subsequently.

References

[1] James Bennett and Stan Lanning. The Netflix Prizes.

[2] Robin Burke and Maryam Ramezani. Matching Recommendation Technologies and Domains. *Recommender Systems Handbook*, 54:367–386, 2011.

[3] F Maxwell Harper and Joseph A Konstan. The MovieLens Datasets : History and Context. 5(4):1–19, 2015.

[4] IMDb. IMDb dataset. http://www.imdb.com/help/show_leaf?usedatasoftware, 2017.

[5] Rasmus Munk. set11521cw2. https://github.com/rasmunk/set11521cw2, 2017.

[6] Netflix. Netflix Prize Data Set. 2009.

[7] J Ben Schafer, D Frankowski, Jon Herlocker, and S Sen. Collaborative Filtering Recommender Systems. *International Journal of Electronic Business*, 2(1):77, 2007.

[8] Toby Segaran. *Praise for Programming Collective Intelligence*. 2007.

[9] Wikipedia. Pearson correlation coefficient, 2017.

[10] Guanwen Yao. User-Based and Item-Based Collaborative Filtering Recommendation Algorithms Design.