Web Search & Information Retrieval Recommendation Systems Project#2

By

Mitali Sahoo
SCU ID: W1632271
msahoo@scu.edu

COEN 272

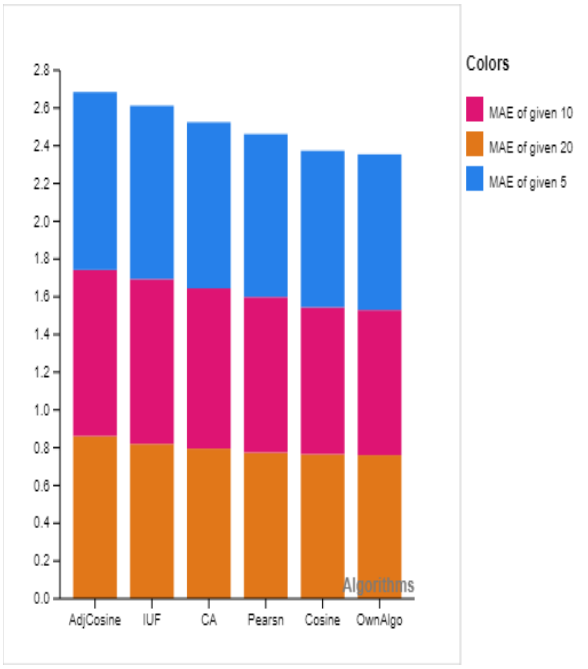
Under Prof. Yi Fang
Santa Clara University

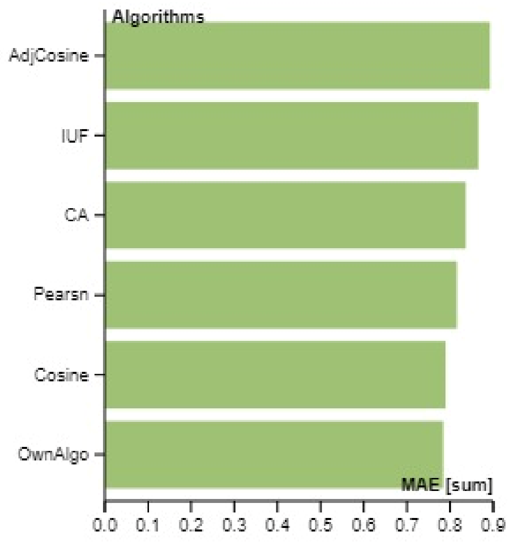
Algorithm Report:

Collaborative Filtering Algorithms MAE report:

Algorithms	**MAE**	MAE of Given 20	MAE of given 10	MAE of given 5
Cosine	0.789812838614	0.76598823188	0.778666666666	0.8290608978
Pearson	0.8158758824495	0.774766084591	0.8241666666	0.862948605727
IUF	0.865580364472172	0.8201987074370	0.8738333333333	0.918219332249
Case Amplification	0.836439008373009	0.795408507765024	0.849333333333	0.879954983118
Adjusted cosine (Item based CF)	0.892833689049049	0.8620623131089	0.88033333333	0.942103288733
Own Algorithm	0.782301756690199	0.760682936239	0.769166666666	0.824934350381

Visualization:





Analysis of report:

Best algorithm: Own Algorithm

Worst algorithm: Item based Adjusted cosine Similarity

Result Justification:

According to me, the prediction performance result of various algorithms is fairly reasonable as -

Analyzing the observed table data, I can say, user-based cosine similarity performed better among all the user-based similarity model as its weighted average method provided accurate user ratings within the rating range. Whereas in Pearson-correlation method, prediction method used (average + deviation) got us prediction out of the 1-5 rating range and needed normalization. Due to this, I assume we lost the prediction accuracy.

In cosine, not limiting ‘k’ [nearest neighbors] value helped in getting better prediction rather than limiting prediction based on few nearest neighbors.

Modified version of pearson correlation, that are IUF and Case Amplification did not get as good results as pearson correlation itself. IUF checks for importance/controversiality of the movies and assigns them more weightage than the popular ones. So as the sample data set given was not large enough, finding rarity of movies is difficult and might be the reason for bad performance.

Case Modification of pearson correlation algorithm, does not predict better than original pearson, and might be due to user X movies dataset was not that large to provide amplification to greater and diminish weights of lower ratings. Also noticed, lower the value of amplification power, **ρ ,** the algorithm performed better. I tried with **ρ=2.5, 2 and 1** and observed the prediction were better for ρ=1 and were getting worser when ρ = 2.5. At ρ=1, it gave exact rating as original pearson correlation algorithm.

Here Item based adjusted cosine similarity did not perform well as compared to user-based similarity methods in terms of rating prediction and runtime efficiency. According to me, that could be due to dataset not being enough large.

Details of own algorithm :

Implementing an own algorithm gave better results than all of the above methods. As user-based algorithms were performing better as compared to item based, tried modifying the way the similar users are picked. While finding similarity, I only fetched users who have rated either exact rating or Δ(delta)1 difference of rating given by users to the movies rated by the active user. This helped, in finding more precise similar users than taking nearest neighbors by defining k-value. By this method I was able to find users who had very similar rating as active user. Example: if active user had rated 2, we only choose a neighbor if they have rate 1/2/3. In the second step, while finding rate prediction, filtered users (from previous step) who rated the target movie, and averaged their target movie rating to assign rating to active user.

Read Me:

Instructions to run:

1. Place the train.txt, test5.txt, test10.txt, test20.txt, train and test input files in same location as the python code.
2. In the recommender.py code, go to the end of the file and uncomment the function call (line 438 - line 442) , for the algorithm you want to execute. Run only one function at once, as all algorithms are programmed to write output to the same file names – named result5, result10, result20.
3. Command to run on terminal:

```
$ python3 recommender.py
```

4. Output file generated:

    result5, result10, result20 files will be generated in the same path as the code file