Project 2 Report

Adam Miller
Professor Fang
Web Information Management

Problem 2: User-Based Collaborative Filtering

For the first part of this assignment, I implemented several user-based collaborative filtering algorithms, including modifications such as cosine similarity, Pearson correlation, inverse user frequency, and case modification.

Results using Cosine Similarity:

- MAE of GIVEN 5: 0.855945979742403
- MAE of GIVEN 10: 0.791666666666667
- MAE of GIVEN 20: 0.769171409279444
- OVERALL MAE: 0.803193235921852

Results using Pearson Correlation:

- MAE of GIVEN 5: 0.86619982493435
- MAE of GIVEN 10: 0.812166666666667
- MAE of GIVEN 20: 0.797916465708498
- OVERALL MAE: 0.823838450172385

Results using Inverse User Frequency:

- MAE of GIVEN 5: 0.904214080280105
- MAE of GIVEN 10: 0.834833333333333
- MAE of GIVEN 20: 0.823767724510466
- OVERALL MAE: 0.852897717944508

Results using Case Modification:

- MAE of GIVEN 5: 0.914968113042391
- MAE of GIVEN 10: 0.8865
- MAE of GIVEN 20: 0.855117198803897
- OVERALL MAE: 0.88249055984239

Problem 3: Item-Based Collaborative Filtering

Here I implemented a basic item-based collaborative filtering algorithm. This algorithm bases estimated results off of similar items, or movies, rather than similar users like the previous algorithms.

Results using Item-based Collaborative Filtering:

- MAE of GIVEN 5: 0.894335375765912
- MAE of GIVEN 10: 0.807166666666667
- MAE of GIVEN 20: 0.798881064917527
- OVERALL MAE: 0.832252503693975

Problem 4: My Own Algorithm

For my own algorithm, I used cosine similarity as a foundation, since I achieved the best results with it so far. Then, I took all cases where the similarity matrix was empty, and rather than fill with 0 (which gets constrained to 1), I filled these values with the average movie rating among all users. Failing this, I simply used the mean value of all ratings, 3.

Results using my own algorithm:

- MAE of GIVEN 5 : 0.855570839064649
- MAE of GIVEN 10 : 0.792166666666667
- MAE of GIVEN 20 : 0.769074949358541
- OVERALL MAE : 0.803152191758332

Results Discussion

After comparing results from all four filtering algorithms, cosine similarity emerged as the best performing similarity algorithm. Pearson Correlation was the second most effective, with the two modified versions (Inverse User Frequency and Case Amplification) yielding even worse results. However, all algorithms were able to produce an MAE of less than 0.9.

I presume the reason that cosine similarity outperformed Pearson Correlation across the board is because the rating format is uniquely partitioned into only 5 possible answers. The fact that Pearson Correlation uses an average value to calculate predictions is negated or even hindered because every answer must be rounded after estimation anyways. Additionally, I assume that inverse user frequency yielded a higher MAE because of its presumption that popular movies are less impactful on the final answer. In reality, there were only ever about 5 or less similar users with requisite ratings in the similarity matrix. This means that all similar results had a large impact on the final answer, and any adjustment to the weights of these results would only reduce the accuracy of the final answer. Case modification had a similar issue, in which all answers are multiplied by a power in order to amplify more similar results and nearly eliminate less similar results. Since the k *nearest* neighbors are used for calculations anyways, it's hard to see how this modification would improve the final MAE at all.

Finally, I decided to use cosine similarity for my own algorithm since it yielded me the best results thus far. Then, I used the knowledge that some movies came up with few or no similar users, and I know that these values would eventually get constrained to a value of 1, which may not reflect the actual answer as closely as possible. I put in a conditional check for these particular values at the end of my own algorithm, and replaced them with the average movie rating among *all* users instead. In the case where there were no ratings at all for that particular movie, I filled the score with the mean value of all the scores, which came out to 3. This did help me marginally improve the accuracy of the algorithm compared to pure cosine similarity.