Matthew Koken

COEN 169

Project 2

|  |  |
| --- | --- |
| Algorithm | MAE |
| Cosine Similarity | 0.877442127729437 |
| Pearson | 0.823838450172385 |
| Pearson IUF | 0.818051223116073 |
| Pearson Case Amplification | 0.829051058939419 |
| Item-Based | 1.25890658348383 |
| Custom – Cos Euclidean Hybrid | 0.888277786898703 |
| Pure Euclidean Distance | 0.840215071416844 |

**Results**

For all of the algorithms, with the proper optimizations, the MAE should be able to break 0.8. Unfortunately, that wasn't possible with the optimizations I used. Additional fine tuning with the number of k similar neighbors to utilize and the rho for the case amplification could further refine the results and improve accuracy. In terms of speed, additional class based optimizations could be utilized in order to improve runtime and reduce the number of extraneous calculations that must be performed again in the current approach. Here, cosine similarity takes the baseline, and other algorithms should be able to improve upon it. However, cosine is the fastest to run so that is also a consideration that must be made. Pearson gives a good improvement without too much of a runtime tax, and the modifications on Pearson give further refinements at the cost of speed. IUF and Case Amplifications take into account how popular movies are and handle the cases of universally loved movies, etc. Both perform similarly to Pearson, as expected. Of the two modifications, IUF performed slightly better. For all of the algorithms, as the number of initial, user rated movies increased the performance decreased significantly. This shows that much more can be improved by taking a more fully object-oriented approach to save and reduce the amount of calculations. The current use of many nested for loops is harder to follow and is poor on performance.

For Item-Based, we'd expect similar results to cosine similarity. By looking at additional movies and using other algorithms, such as Pearson, the accuracy and performance of the item-based approach could be drastically improved as well.

For the custom algorithm, I decided to use a hybrid of Cosine similarity and Euclidean distance. Cosine similarity fails to account for the distance between two vectors, so there is a possibility of a loss of precision here. Instead, it only looks at the angle of the two vectors. To counteract that shortcoming, Euclidean distance can be used to get a different rating by looking at different components. Euclidean distance can give the distance between the two vectors, taking into account of the lengths of the vectors. Together, averaging the retrieved ratings should be able to overcome the shortcomings of both algorithms while performing well. The results show not quite so much, and tweaking the weights of each rating may be able to give a better result. Performance-wise, it tends to run faster than any of the Pearson based algorithms. Here, this may be an advantage.

In terms of the best balance between performance and accuracy, then the Pearson algorithm tends to work as the current best. With small performance hits, IUF increases the accuracy further. Looking at future potential given better performance optimizations, then Item-based may be able to give the best balance between performance and accuracy.