MEMORYLEAK

RECOMMENDATION ALGORITHM

To recommend music to the users, we generate some rating data, which represents the preference value of the user on a given music item. Improving the speed of generation process was our objective. In this operation and we came up with a new and more efficient algorithm.

Our previous algorithm inserts the logs to the database, then map-reduces the data from the database and finally applies the math model to the retrieved data to generate the rating values. However, the involvement of the database slowed down our speed. Therefore, we decided to eliminate it by performing the process operation directly on user logs. We first concatenate the logs. Then we sort them with external sort to make the process operation easier and faster. After the sort operation, we iterate on the file to process it with our math model, which we introduced in the 2nd iteration.

To illustrate, the concatenated and sorted logs of 2722475 is given below. Let's calculate that user's rating value on the song 71845.

2722475,41869,3544,36789,2013-09-07 00:05:44,0.63,SI
2722475,41869,3544,36789,2013-09-07 00:05:44,0.63,SI
2722475,41869,3544,36789,2013-09-10 00:05:44,0.63,SI
2722475,71845,5571,36789,2014-04-26 00:05:44,0.63,SI -> 4 days ago
2722475,71845,5571,36789,2014-04-27 00:05:44,0.63,SI -> 3 days ago
2722475,71845,5571,36789,2014-04-28 00:05:44,0.63,SI -> 2 days ago
2722475,71845,5571,36789,2014-04-28 00:05:44,0.63,SI -> 2 days ago
2722475,71845,5571,36789,2014-04-30 00:05:44,0.63,SI -> 0 days ago
2722475,71845,5571,36789,2014-04-30 00:05:44,0.63,SI -> 0 days ago
2722475,91234,2145,14355,2013-09-07 00:05:44,0.63,SI
2722475,95545,5656,65322,2013-09-07 00:05:44,0.63,SI

$$R(u,s) = \frac{min(max, r(u,s))}{max} \times 5$$

$$r(u,s) = \sqrt{\frac{\left(\sum_{i=1}^{n_s} d_{max} - d_i\right)^2}{n_s}}$$

$$max = d_{max}\sqrt{n}$$

d1 = 0, d2 = 0, d3 = 2, d4 = 2, d5 = 3, d6 = 4
dmax = 60 (= last 60 days)
max = 60*sqrt(5) = 134,16
ns = 6
r(u,s) = sqrt((60 + 60 + 58 + 58 + 57 + 56)^2 / 6) = 142,478
R(u,s) = 5, which means that the user 2722475's rating value on the song 71845 is 5 out of 5.
(Please see the report of the 2nd Iteration for more details on the math model)

To make recommendations to a given user, we first calculate the given user's similarity with other users using Tanimoto similarity metric. Then, we use K-Nearest Neighbors algorithm to find the neighbors of the given user, where k is 3. Then, we sort the songs of these neighbors by their rating values and eliminate the songs that the given user has already listened. Finally, we pick the top 5 songs in this sorted list and display them as the recommended songs.

To illustrate, let's use the following example data with Tanimoto Similarity and 2-Nearest Neighbors, and generate 2 recommendations for user 3.

User,Song,Rating:
1,a,4
1,b,5
2,a,1
2,e,5
3,a,4
3,c,3
4,c,3
4,f,1
4,d,1
5,a,4
5,d,5
5,c,3
5,e,2

User 3's similarity with user 1: 1/3 = 0.33,
user 2: 0/4 = 0.00,
user 3: NA,
user 4: 1/4 = 0.25,
user 5: 2/4 = 0.50

$$T(a,b) = \frac{N_c}{N_a + N_b - N_c}$$

2-Nearest Neighbours: user 1 and user 5

Possible Recommendations:

- song b of user 1 with rating 5
- song d of user 5 with rating 5
- song e of user 5 with rating 2

Top 2 Recommendations for user 3:

- Song b
- Song d

EVALUATION

DATA SET

We used the logs of the last 13 days from Argedor's dataset, which contains 5,225,092 logs, 144,292 different users and 109,147 different songs. A user listened to an average of 36 songs.

The logs above are our both train and test data. 30% of that data randomly chosen and hidden from the recommender for testing, which means 70% of randomly selected data is used as the train data.

EVALUATION METRICS

Prediction-recall evaluator hides some (30%) of the selections of the user, and asks the recommender to predict a set of items that the user will listen using the remaining (70%) data. Then, it calculates precision and recall values based on the following formulas;

$$Precision = \frac{Recommended\ Items\ \cap\ Relevant\ Items}{Recommended\ Items}$$

$$Recall = \frac{Recommended\ Items\ \cap\ Relevant\ Items}{Relevant\ Items}$$

"Relevant items" in the above formulas refer the hidden items in our evaluator.
To illustrate, let's consider the user 1000217 in Argedor's dataset. According to the logs in the dataset mentioned above, that user listened to 34 songs (11 different songs). Assume that the evaluator hides the songs {3408894, 3407844, 3408786, 3408899} from the recommender and the recommender recommends {3408894, 3409040, 3408786, 64557, 3409296}. Then, Recommended Items ∩ Relevant Items is equal to {3408894, 3408786}. Therefore, the precision value is equal to 2 over 5 which is 40% and precision value is equal to 2 over 4 which is 50%.

RESULTS

The results of our evaluator were presented in the previous iteration, yet this iteration, we achieved a recent increase in our evaluation result, which is given below.

	Precision	Recall
25-Nearest Neighbor	23%	21%
3-Nearest Neighbor	32%	30%

Previously, we were making recommendations based on the rating data of the 25 nearest neighbors of a user. We decreased the neighbor size to 3. The decrease in the neighbor size increase the accuracy of the recommender, since only the ratings of the “most” similar users are taken into account while recommending music to the user.