Assignment 3

Versions

Spark - 2.2.1

Scala - 2.11

Python - 2.7

Task 1 : Jaccard Based LSH

Python command
spark-submit Piyush_Umate_task1_Jaccard.py <rating_file_path>

Scala command
spark-submit --class JaccardLSH Piyush_Umate_hw3.jar <rating_file_path>

Python output -
Time: 113.05479598 sec
Precision 1.0
Recall 0.815018491384

Scala output
Time: 102sec

Description -
I used 8 hash functions with b = 4 and r = 2. My hash function was of the form (7*row_index) + (3*i) % 671 . The value of "i" varies from 1 to 8 . A good hash function is one that contains prime numbers thus 7 and 3 were used.

Task 2.1 : Model Based CF

Python command -
spark-submit --driver-memory 4g --executor-memory 4g Piyush_Umate_task2_ModelBasedCF.py <rating_file_path> <testing_file_path>

Scala command -
spark-submit --driver-memory 4g --executor-memory 4g --class ModelBasedCF Piyush_Umate_hw3.jar <rating_file_path> <testing_file_path>

ML-Latest - Small data Python output
>=0 and <1: 13761
>=1 and <2: 4149
>=2 and <3: 714
>=3 and <4: 103
>=4: 6
RMSE: 0.95073628922
Time: 10.8795609474 sec

ML-Latest - Small data Scala output
>=0 and <1: 13826
>=1 and <2: 4091
>=2 and <3: 708
>=3 and <4: 102
>=4: 6
RMSE: 0.9479834931653777
Time: 7sec

ML-20m - Big data Python output
>=0 and <1: 3232631
>=1 and <2: 723767
>=2 and <3: 82068
>=3 and <4: 7655
>=4: 210
RMSE: 0.817364633871
Time: 1286.95861316 sec

ML-20m - Big data Scala output
>=0 and <1: 3232743
>=1 and <2: 723742
>=2 and <3: 82029
>=3 and <4: 7611
>=4: 206
RMSE: 0.8172113310320842
Time: 545sec

Description -
For Model Based CF , I used rank as 10 , 12 as the number of iterations and 0.1 as the regularization factor of ALS. I used ParamGridBuilder in tuning different values and then used Regression Evaluator to extract the best parameters.

Task 2.2: User Based CF

Python command -
spark-submit Piyush_Umate_task2_UserBasedCF.py <rating_file> <testing_file_path>

Scala command -
spark-submit --class UserBasedCF Piyush_Umate_hw3.jar <rating_file> <testing_file>

ML-Latest - Small data Python output
>=0 and <1: 14965
>=1 and <2: 4088
>=2 and <3: 1039
>=3 and <4: 153
>=4: 11
RMSE: 0.988432740607
Time: 30.9461770058 sec

ML-Latest - Small data Scala output
>=0 and <1: 15070
>=1 and <2: 4000
>=2 and <3: 1023
>=3 and <4: 154
>=4: 9
RMSE: 0.982381870797202
Time: 54sec

Description -
For User Based CF, I used Pearson correlation as the similarity metric. The outliers (outcasts) whose predicted ratings were not within 0 to 5 were fitted in that range by normalization. For certain predicted ratings where Pearson correlation fails, I used imputation boosting to handle them.

Task 2.3 Item Based CF with LSH

Python command - spark-submit Piyush_Umate_task2_ItemBasedCF.py <rating_file_path> <testing_file_path> <jaccard similar movies>
Scala command -
spark-submit --class ItemBasedCF Piyush_Umate_hw3.jar <rating_file_path> <testing_file_path> <jaccard similar movies>

ML-Latest - Small data Python output
>=0 and <1: 13944
>=1 and <2: 5133
>=2 and <3: 959
>=3 and <4: 208
>=4: 12
RMSE: 1.00286340067
Time: 8.94653177261 sec

ML-Latest - Small data Scala output
>=0 and <1: 13924
>=1 and <2: 5152
>=2 and <3: 955
>=3 and <4: 214
>=4: 11
RMSE: 1.0049722266060221
Time: 9sec

Comparing the result with CF without LSH and answering how LSH could affect the recommendation system?

While LSH compares the similarity between movies by taking into consideration only whether the user rated the movie or not and not the weight i.e. rating given by the user, LSH could severely affect the recommendation system. LSH in turn causes loss of data in terms of the ratings made by the user. For example, while Pearson correlation takes into consideration the similarity between the ratings of user to predict the rating, LSH only takes into consideration whether a user rated a movie or not but not how good was the rating.