CMPE 256 - Kindle Book Recommendation System

Name: SNEHA .

SJSU ID: 013850174

Kaggle ID: Sneha Patil

RMSE(kaggle): 0.43323

Load datasets:

1. Created Pandas dataframes using training.csv and test_with_asin_reviewerID.csv as training dataset and test dataset respectively. The training dataset will be used to learn the model and the test dataset will be used for final predictions.

Data Preparation and Preprocessing:

Training Data:

1. Checked for the null/NaN values in the training data to be able to perform data preprocessing steps like data cleaning, dimensionality reduction, data transformation etc. Used isnull() and sum() functions to obtain results.
2. For the problem statement at hand, the attributes with least significance were reviewText, reviewerName, summary and unixReviewTime.
3. Out of the remaining attributes the important ones were reviwerID, asin and overall which map to userID, itemID and the rating respectively.
4. Created final training dataframe using the above three attributes.

Testing Data:

1. The test dataframe consists of reviewerID and asin attributes. Added a third column “ratings” to this dataframe with initial value=0 as the parser Surprise reader and predict method requiree 3 attributes.

Used the Surprise library for processing. Surprise provides various classes that help us handle huge datasets with ease. Surprise is great at handling missing values and needs minimal preprocessing to work on the input data unless otherwise specified in the problem statement.

Parsed the training dataframe using Surprise library Reader class.The dataset thus obtained, will act as the raw dataset from which training and validation sets will be obtained.

After parsing the raw data, I used the build_full_trainset method from the Trainset class to build the final training set. Using the same raw data, I created a validation dataset to infer how well the model is predicting the known data(for local testing). Used train_test_split method to obtain the validation set. The measure to determine the error rate is RMSE(Root Mean Square Error). The lower the RMSE, the better the predictions. Worked with care, so the model does not overfit to the training data.

Prediction models tried:

1) Baseline Prediction Model: Algorithm predicting the baseline estimate for a given user and item.
   The primary prediction model used is BaselineOnly from the prediction_module package in Surprise. Configured the bsl_options with “als” and “sgd” as the method for computation. Both yielded similar results.
   Using the baseline estimates the RMSE obtained was between 0.80 to 0.82 initially. When the regularization parameters for user and item were changed from default to 3, the RMSE reduced to 0.6781. But for further iterations, there wasn’t any significant improvement in score.

2) KNN: Using Pearson- similarity options, tried working with KNN but resulted in memory error. So discarded KNN from model consideration.

3) NMF: A collaborative filtering algorithm based on Non-negative Matrix Factorization. After implementing the NMF model, the prediction RMSE actually increased to 0.96.

4) **SVD:** SVD listed as one of the top models in Surprise worked well for the current problem statement. An initial implementation using default parameters yielded RMSE 0.80. After hyper-parameter tuning, like increasing the number of latent factors(n_factors=500), increasing the number of iterations(n_epochs=50), modifying the default reg_all and lr_all parameters, the model performed better. The lowest RMSE obtained is 0.43323.

5) SVDpp: SVDpp being another good prediction model could prove to be the one. But owing to its high computation time, the model did not fit my requirements. For one single run, SVDpp took around 1 hour to compute the results yielding 0.83 as the RMSE

Parameter tuning:

GridSearchCV: GridSearchCV is an algorithm that helps identify which parameter combination yields the best results. Used grid search to find optimal parameters for SVD. Parameters included n_epochs,reg_all, lr_all but the RMSE obtained(0.8367) was not really an improvement from baseline.

Final predictions and creation of output file:

Since SVD provided better results, I chose SVD as my prediction model. Used the test data to obtain final predictions and loaded them into a csv file using file write operation.

References used for the code and report:

https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html
Surprise_demo.ipynb from the class demo file.