Final Project Report - Group 88

https://github.com/nyu-big-data/final-project-group_88

Susan Chen
Computer Engineering
NYU Tandon School of Engineering
New York, New York
xc689@nyu.edu

Garima Gupta
Computer Engineering
NYU Tandon School of Engineering
New York, New York
gg2612@nyu.edu

Abstract

In this project, we are comparing and evaluating the recommendation given by the popularity baseline model, alternating least squares(ALS) model, and lightfm. Garima is responsible for the original dataset partitioning as well as the extension (Lightfm) implementation and evaluation. Susan is responsible for the popularity baseline model and ALS model implementation and evaluation.

1 Approach

This section describes the approach followed for the project.

1.1 Data Partitioning

The original dataset of rating.csv is split into train, validation, and test sets in approx—60/20/20 ratio.

Figure 1 demonstrates the process where the data is divided into three divisions (Div01, Div02, Div03), and further, Div02 and Div03 are divided based on each user's rating history.

The dataset is divided into three sets based on an equal number of users. The first part is added to the training set directly. Each user's history in the second and third divisions is divided into an 80/20 ratio, where aggregation of the 80% (user's history) for both the divisions goes to the train set. Further, the 20% (user's history) for the second and third divisions goes to the test and validation set. The users in both parts are grouped and ranked according to their timestamp. Both the train and test/ train and validation sets have shared users, but there are no common users in the test and validation set. All the three training parts are combined at the end. The process is implemented using PySpark.

After the splits are generated, they are saved into CSV files to be used to evaluate the models later with ease. The rating.csv file was chosen out of all the files provided by the MovieLens dataset since it has all the information needed for the models, including userId, movieId, and rating.

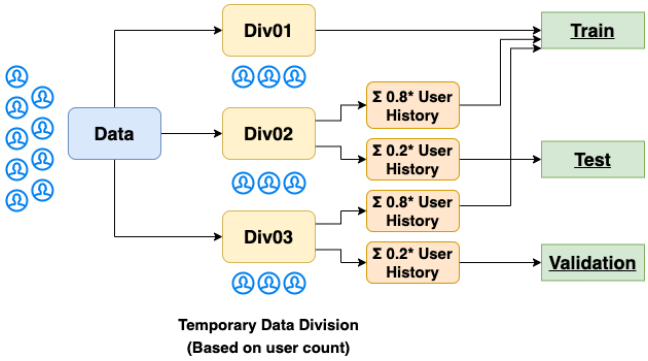

Figure 1 Data Partitioning Workflow

1.2 Evaluation Metrics

- ➔ Mean average precision @ 100 - returns the mean average precision (MAP) at first 100 ranking of the recommendation [4]
- ➔ NDGC @ 100 - computes the average NDCG value of the recommendations, truncated at ranking position 100.[3]
- ➔ Root Mean Squared Error (RMSE) - measures the average magnitude of the errors and is concerned with the deviations from the actual value [2].

With MAP @ 100, we could analyze the performance of our model by comparing the predicted movies by the models to a user with the ones ranked highly by this specific user in the test set to get a score as an indication of recommendation accuracy. The higher the score, the more accurate the model is.

NDGC is a measure of ranking quality. We chose to use it on top of MAP @ 100 because besides measuring the accuracy of our models, we also want to know how relevant the recommended results are. NDGC score ranges from 0 to 1, the higher the score, the more relevant the recommended results are.

RMSE indicates the absolute fit of the model to the data. Therefore, we look at the RMSE score when we tune the ALS model hyperparameter as well as

comparing among different models. Low score indicates higher level of accuracy.

2 Experimentation and Results

In this section we discuss the implementation and results.

2.1 Popularity Baselines on small and full datasets

➔ Small dataset -
- **Total popularity baseline model query execution time:** 0.1392209529876709 seconds
- **Mean average precision at 100 =** 4.77592966931986e-05
- **NDCG at 100 =** 0.0005998836997623643
- **RMSE =** 1.3863170633011772

➔ Full dataset -
- **Total popularity baseline model query execution time:** 0.12665081024169922 seconds
- **Mean average precision at 100 =** 5.107539918783622e-08
- **NDCG at 100 =** 1.0593979809091906e-06
- **RMSE =** 2.881591821992044

The advantage of the popularity baseline model is that it is simple to implement, the query runs very fast. However, the NDCG score is very low, which means the recommendation is almost irrelevant. Also, it is very inaccurate.

2.2 Documentation of latent factor model's hyperparameters

- We increased rank from default of 10 to 200, maxIter from 10 to 15 and decreased regParam from 1.0 to 0.01.
- rank is the number of latent factors in the model (defaults to 10)
- maxIter is the maximum number of iterations to run (defaults to 10)
- regParam specifies the regularization parameter in ALS (defaults to 1.0)

When we tuned the hyper-parameter, we mostly looked at rank and maxIter. When we evaluate the hyper-parameters, we also take into account the execution time. If a change in value increases the accuracy of the result by only a small percentage, but significantly increases the execution time, we do not make that change. For example, when we increase the rank from 200 to 300, the RMSE on the validation set only improved by around 0.001, but the execution time increased by almost 20%. As a result, we kept using rank=200.

After many experiments with different combinations of rank, maxIter and regParam, we concluded that rank=200, maxIter=15 and reg=0.01 to produce the best result in terms of time and accuracy.
Here are some of the sample observations for different combinations of hyper-parameters:

- Rank=50 maxIter=10 regParam=0.01 on Small Dataset

Getting top 100 movies with highest ratings
- Total popularity baseline model query execution time: 0.16105914115905762 seconds

With popularity baseline model:
- ➔ **popularity baseline Mean average precision at 100 =** 4.77592966931986e-05
- ➔ **popularity baseline NDCG at 100 =** 0.0005998836997623644
- ➔ **Popularity baseline model RMSE =** 1.3863170633011772

With ALS model:
- ➔ **Total ALS model training time:** 19.406323194503784 seconds
- ➔ **ALS validation set Mean average precision at 100 =** 0.0033835851390158936
- ➔ **ALS validation set baseline NDCG at 100 =** 0.020278472012230448
- ➔ **ALS Validation set RMSE =** 1.148244583014365
- ➔ **ALS test set Mean average precision at 100 =** 0.003858092504687739
- ➔ **ALS test set baseline NDCG at 100 =** 0.023064116551216974
- ➔ **Test set RMSE =** 1.0625371714708476

After we finish running the job and look at the output, we then decide on whether the change in hyperparameter is worth the time increase and the accuracy improvement.

2.3 Evaluation of Latent factor model on small and full datasets

On small dataset
- ➔ **Total ALS model training time:** 247.0239245891571 seconds

- ➔ **ALS testset Mean average precision at 100 =** 0.007239272610763887
- ➔ **ALS testset baseline NDCG at 100 =** 0.03244381793992112
- ➔ **Test set RMSE =** 0.7407424192790064

On large dataset

- ➔ **Total ALS model training time:** 26263.57272815704346 seconds
- ➔ **ALS test set Mean average precision at 100 =** 2.9557697570359662e-05
- ➔ **ALS test set baseline NDCG at 100 =** 0.00016839664564973728
- ➔ **Test set RMSE =** 1.7010586875630123

As we could tell from the result of the ALS model, the disadvantage is that it takes a long time to train on the datasets. However, it produces much better results than the popularity baseline model.

3 Documentation of Extension

As an add-on we implemented the LightFM [1] model with WARP loss function on the small dataset and single core. The dataset for this set of experiments is divided into training and testing sets and keeping in mind all the users belong to both train and test as required in training and testing of the LightFM model. The experiment is done using lightfm and scipy libraries on Google Colab. We present the evaluation metrics for each of them.

- ➔ The results of LightFM:
  - ◆ **Total LightFM model training time:** 4.057746887207031 seconds
  - ◆ **Training Accuracy :** 0.9729295969009399
  - ◆ **Testing Accuracy:** 0.8905578255653381
  - ◆ **Testing Precision:** 0.03329470008611679
- ➔ The results of ALS recommender system on the same dataset as LightFM
  - ◆ **Total ALS model training time:** 20.030635833740234 seconds
  - ◆ **ALS test set Mean average precision at 100 =** 0.004223344182253933
  - ◆ **ALS test set baseline NDCG at 100 =** 0.024254305665244905
  - ◆ **Test set RMSE =** 1.1021707346844034

We majorly compare the training time for both models. Although the training time for LightFM differed each time it ran, the average training time was calculated at 5 seconds. The mean average precision @ 100 for LightFM is greater than that of ALS in this dataset, which follows the trend found in other experiments performed.

Conclusion

After comparing the execution time, ranking metrics and regression metrics of the 3 different models we implemented, we concluded that the ALS model produces the best results in terms of accuracy and relevancy.

REFERENCES

[1] Kula, Maciej. "Metadata embeddings for user and item cold-start recommendations." arXiv preprint arXiv:1507.08439 (2015).

[2] Chai, T. and Draxler, R. R.: Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature, Geosci. Model Dev., 7, 1247–1250, https://doi.org/10.5194/gmd-7-1247-2014, 2014.

[3] Distinguishability, Consistent. "A Theoretical Analysis of Normalized Discounted Cumulative Gain (NDCG) Ranking Measures." (2013).

[4] (2009) Mean Average Precision. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_3032.

[5] Takács, Gábor, and Domonkos Tikk. "Alternating least squares for personalized ranking." Proceedings of the sixth ACM conference on Recommender systems. 2012.