Music Recommender System Project

Zehui Gu*
Meiyu Li*
zg745@nyu.edu
ml8457@nyu.edu
New York University
New York, New York, USA

ABSTRACT

This report uses the dataset ListenBrainz dataset, which consists of implicit feedback from music listening behavior, spanning several thousand users and tens of millions of songs. We implement popularity baseline model, alternating least square model, and an extension LightFM to recommend songs for each user. Then we use meanAP at 100 items to evaluate the performance.

KEYWORDS

big data, music datasets, recommender system, popularity baseline model, mean average precision, Alternating Least Square Model, LightFM,

ACM Reference Format:
Zehui Gu and Meiyu Li. 2023. Music Recommender System Project. In *Proceedings of 16th.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 DATA PREPROCESSING

We first explore the small dataset and find that in the interaction dataframe, there are 6954 distinct users and 12M distinct tracks (using msid) and around 51M user history. We decide to downsample the dataset for the sake of scalability for both baseline and ALS model by discarding users who listen less than 6 songs and songs with fewer than 6 listeners. If time and resources allowed, we will try different thresholds to decide the size of downsampling. After processing, our filtered dataset keeps 38% of user history with 6557 distinct users and 510K distinct tracks.

We then count the frequency for each user-item interaction by the following PySpark code:

```
interactions_filter
.groupBy("user_id", "recording_msid")
.agg(F.count("*").alias("frequency"))
```

By partition the dataset into the train and validation set, we are able to store the processsed data into parquet format.

Due to the limited resources in NYU HPC, we decide to choose 50% of users in the all dataset. Then we use this dataset to train our baseline model and Alternative Least Sqaure model (ALS), then compare the performance. Furthermore, it is time-consuming to predict the songs using lightFM on large dataset in Greene. Thus, we choose 50% of users on small dataset to train both LightFM and to train the ALS model again to maintain consistency when evaluating LightFM and ALS.

2 DATA PARTITION

During the training and validation set partitioning, the data is splitted into 20% validation set and 80% training set across users (`user_id`).

For instance, if a user has listened to 233 songs, we take 187 (80%) of them into the training set and the rest 46 (20%) songs into the validation set.

3 EVALUATION CRITERIA

In evaluation, we choose Mean Average Precision (MeanAP) at top 100 predicted songs, a measure of how many of the recommended songs are in the ground truth of relevant songs. MeanAP takes into account the entire ranked list of songs and considers the order in which they are presented. The order in which songs are presented in a recommendation list can significantly impact user experience and engagement. Users tend to focus on the top results, and the ranking of songs determines their visibility and likelihood of being listened to. MeanAP considers the rank order of songs and it gives a comprehensive evaluation of the recommendation models in placing relevant songs higher in the list.

4 POPULARITY BASELINE MODEL

4.1 Model Construction

The popularity per each recording ($P[i]$) will be calculated by the sum of interactions for all users in the ith recording divided by the total number of plays for ith recording plus the damping factor β. The popularity for each recording (i) is the formula below:

$$P[i] \rightarrow \frac{\sum_u R[u,i]}{|R[:,i]| + \beta}$$

where u represents `user_id`, $R[u,i]$ represents the number of times that user u has played recording i. $|R[:,i]|$ represents the total number of plays for recording i across all users. β stands for damping factor which is used to reduce the weight of the popularity of less frequently played recordings.

To tune the hyperparameter β, we select a range of damping factors, 0, 10, 20, 30, 40, 50, 60, 70, 80, 90. Then we sort the recordings

*Both authors contributed equally to this research.

16th, May 2023, New York

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

based on popularity values calculated above, and choose the top 100 most popular recordings as the prediction.

4.2 Results

For the popularity baseline model, We conduct experiments using the small dataset and 50% users' interactions in the full dataset. By tuning the hyperparemeter β, we obtain results in the Table 1 below.

Table 1: Evaluation on Baseline Popularity Model

Dataset	best β	MeanAP (Validation)	MeanAP (Test)
small	50	3.34e-05	4.03e-05
50% full	10	6.48e-05	4.45e-05

5 ALTERNATING LEAST SQUARE MODEL

5.1 Model Construction

In this section, we use Alternating Least Square Model (ALS model) to factorize a user-item interaction to build recommendation systems because ALS model can handle large and sparse dataset and implicit feedback.

Before constructing ALS model, we first use `window` in PySpark to convert the string values in recording ID to integer values to accomodate the setting in the model because the ALS algorithm computes dot products between user and item factor vectos, thus rendering an efficient way to perform computatings using integers.

For hyperparmeters tuning, there are 3 parameters will be tuned by comparing performance on training and validation sets. Due to the limited resources on HPC, we only choose a small range of parameter setting to construct our models.

Here are the choices set for the three hyperparmeters:

- rank: the number of latent factors which refers to the dimensionality of the user and item matrix. We choose values of 20, 50, 100 as the rank parameter in the model to avoid overfitting of a too high rank (400, 500, etc.) and avoid underfitting of a too low rank (3, 5, etc.)
- Regularization: a $l-2$ norm to add penalty and prevent overfitting performance. We choose values of 0.1, 1, 10 as the regularization parameter. These values are appropriate because too large regularization will penalize the model more and thus underfitting and too small regularization term will not give a
- α: implicit feedback parameter which reflects how much we value observed events versus unobserved events. We choose values of 1, 5, 10 as the α term.

After setting our parameters, we construct the als model by the following PySpark code:

```
als = ALS(userCol='user_id',
itemCol='recording_msid_numeric',
ratingCol='frequency',
rank=rank, regParam=reg, alpha=alpha,
maxIter=5, coldStartStrategy='drop',
implicitPrefs=True)
```

Then we use grid search method to tune the parameters listed above, and choose the parameter setting that reaches a high accuracy on test set.

5.2 Results

We performed experiments on a subset of the full dataset, specifically focusing on interactions from 50% of the users. To optimize the performance of our model, we perform a grid search over the hyperparameter setting stated above. Due to page limitations, we present a selection of the grid search results in Table 2, showcasing the Mean Average Precision at 100 (MeanAP) scores obtained on both the validation set and the test set.

From the table, we coud see that the highest meanAP of 0.00691 at validation is achieved at rank=100, regularization=1, α=1, with a corresponding meanAP of 0.00513 at the test set. Compared to the results of popularity based model in the previous section, we could see there is an improvement on the Test MeanAP from 4.45e-05 to 0.00513.

Table 2: Evaluation on Alternating Least Square Model

Parameter Setting	MeanAP (Validation)	MeanAP (Test)
$rank = 20, reg = 0.1, \alpha = 1$	0.00396	0.00212
$rank = 20, reg = 0.1, \alpha = 5$	0.00323	0.00254
$rank = 20, reg = 0.1, \alpha = 10$	0.00461	0.00342
$rank = 50, reg = 0.1, \alpha = 1$	0.00577	0.00249
$rank = 50, reg = 0.1, \alpha = 5$	0.00376	0.00304
$rank = 50, reg = 0.1, \alpha = 10$	0.00457	0.00249
$rank = 50, reg = 1, \alpha = 1$	0.00524	0.00438
$rank = 50, reg = 1, \alpha = 5$	0.00329	0.00168
$rank = 50, reg = 1, \alpha = 10$	0.00591	0.00466
$rank = 50, reg = 10, \alpha = 1$	0.00186	0.00160
$rank = 50, reg = 10, \alpha = 5$	0.00364	0.00316
$rank = 50, reg = 10, \alpha = 10$	0.00266	0.00208
$rank = 100, reg = 1, \alpha = 1$	**0.00691**	**0.00513**
$rank = 100, reg = 1, \alpha = 5$	0.00521	0.00435
$rank = 100, reg = 1, \alpha = 10$	0.00488	0.00240

In addition, we can observe that the model with higher rank (= 100) will achieve a precision on test set when the regularization term and the α stays the same. We did not try models with rank $>$ 100 due to the high cost on computation and the increasing chance of overfitting.

6 EXTENSION

6.1 Model Construction

In this section, we implement `lightFm`, known for its flexibility and ability to handle sparse data. It uses matrix factorization to make recommendations on the top 100 songs.

First we use `Dataset.fit` and `Dataset.build_interactions` to transform the dataframe of numeric user ID, numeric recording ID to interaction matrix. Then we train and test the LightFM model on the matrix by using the following code:

```
model = LightFM(loss='warp')
model.fit(interactions)
```

Then we implement evaluation metric to evaluate the performance.

6.2 Results

Due to the time constraints and limited resources, we perform our experiments using a subset of the small dataset, focusing on interactions from 50% of the users.

Table 3: Evaluation on LightFM and ALS

Model	MeanAP at 100	Model Fitting Time
`LightFM`	0.00402	12.38 seconds
`ALS`	0.000179	2.32 seconds

As we can see from Table 3, LightFM takes 12.38 seconds to fit the model using the small dataset, much longer than Spark's parallel ALS model does. This is likely due to the parallel processing capabilities offered by powerful Spark clusters. Also, LightFM is a hybrid recommendation algorith involving both collaborative filtering and content-based approaches, requiring more computational steps. In terms of the resulting accuracy, LightFM achieves a Test MeanAP of 0.00402, which is much better than MAP@100 achieved by ALS model approximately 1.79e-04. This indicates that LightFM provides a notably higher level of recommendation accuracy compared to the ALS model on the given test dataset.

7 GITHUB LINK

https://github.com/nyu-big-data/final-project-group-21