# Book Recommendation Engine
# DS-GA 1004, Spring 2020

Parthvi Sanjay Shah
pss434@nyu.edu

Gunjan R. Desai
grd285@nyu.edu

Anusha R. Patil
arp624@nyu.edu

May 11, 2020

**Abstract**

This project describes a Recommender Engine that filters the data using algorithms and attempts to predict the preferences of users thereby, making suggestions based on their preferences. It does this by learning the user's past behavior and recommending the most relevant items to the users.

## 1 Overview

We implemented the recommendation engine using the Alternating Least Squares (ALS) Matrix Factorization method. ALS works by trying to find the optimal representation of a user and item matrix. The best part of ALS is it alternates between finding the optimal value for user matrix and item matrix. We evaluated our recommendations based on MAP and Precision at k. We have used spatial data structure to implement accelerated search at query time by using the annoy package and carried out data exploration using t-SNE.

## 2 Data Processing

Exploring the dataset, we begin with the user-book interaction file which contains the columns user_id, book_id and rating that we will be using to build the model.

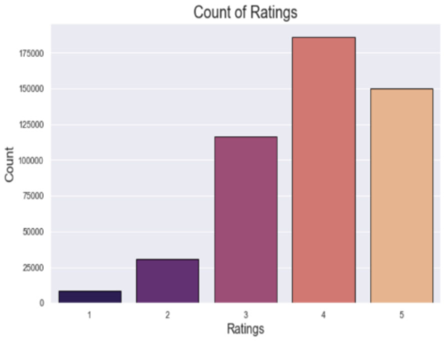


Figure 1: Distribution of Ratings

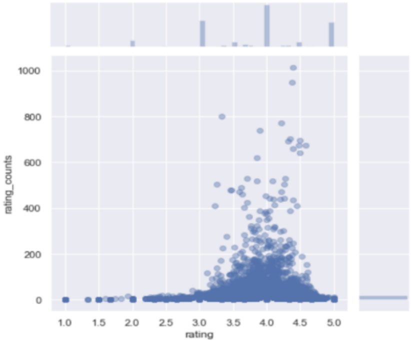


Figure 2: Ratings in general

## 2.1 Data Splitting and Sampling

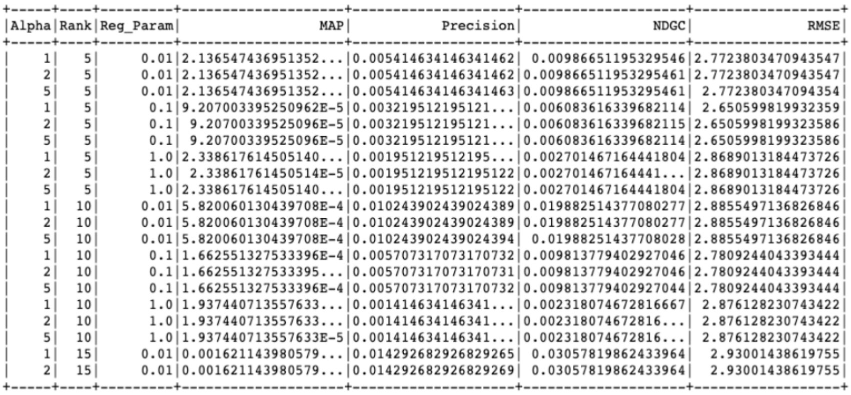
After filtering the users with more than 10 interactions, we collected the distinct user_id along with their interactions from the dataset and distributed them in 60-20-20 percent proportion to train, validation and test sets respectively. Then, We split the users into validation and test sets such that they contain interactions that are present in the training set. Next, we collected the user_id in validation set and sampled 50% of their interactions by using sampleby() and similarly for test set. Since ALS models only accept integer values as parameters, we converted the string valued columns (user_id, book_id) by using String Indexer. Also, Because of the limited computing resources, We sampled 1% of users and took all their interactions to make a subset of the data.

# 3 Model and Experiments

Implemented ALS from Spark's ML library. After fitting the model, we made top 500 predictions for validation and test sets and then retrieved latent features.

## 3.1 Hyper-Parameter Tuning:

*Precision at k*: How many of the predicted items are truly relevant.
*Mean Average Precision (MAP)* : How many predicted also appear in the true label set.
We hyper-tuned the parameters using cross-validation.
- Rank: [5 , 10, 15] - the dimension of latent factors.
- Alpha: [1 , 2 , 5] - the scaling parameter for count data.
- Regularization Parameter: [0.01 , 0.1 , 1] - the parameter which controls overfitting.



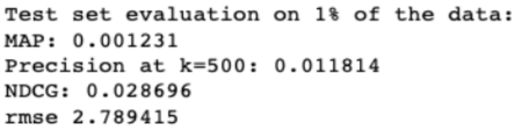
```
+-----+----+---------+--------------------+--------------------+--------------------+------------------+
|Alpha|Rank|Reg_Param|                 MAP|           Precision|                NDGC|              RMSE|
+-----+----+---------+--------------------+--------------------+--------------------+------------------+
|    1|   5|     0.01|2.136547436951352...|0.005414634146341462| 0.00986651195329546|2.7723803470943547|
|    2|   5|     0.01|2.136547436951352...|0.005414634146341462|0.009866511953295461|2.7723803470943547|
|    5|   5|     0.01|2.136547436951352...|0.005414634146341463|0.009866511953295461| 2.772380347094354|
|    1|   5|      0.1|9.207003395250962E-5|0.003219512195121...|0.006083616339682114| 2.650599819932359|
|    2|   5|      0.1| 9.20700339525096E-5|0.003219512195121...|0.006083616339682115|2.6505998199323586|
|    5|   5|      0.1| 9.20700339525096E-5|0.003219512195121...|0.006083616339682114|2.6505998199323586|
|    1|   5|      1.0|2.338617614505140...|0.001951219512195...|0.002701467164441804|2.8689013184473726|
|    2|   5|      1.0| 2.33861761450514E-5|0.001951219512195122|0.002701467164441...|2.8689013184473726|
|    5|   5|      1.0|2.338617614505140...|0.001951219512195122|0.002701467164441804|2.8689013184473726|
|    1|  10|     0.01|5.820060130439708E-4|0.010243902439024389|0.019882514377080277|2.8855497136826846|
|    2|  10|     0.01|5.820060130439708E-4|0.010243902439024389|0.019882514377080277|2.8855497136826846|
|    5|  10|     0.01|5.820060130439708E-4|0.010243902439024394| 0.01988251437708028|2.8855497136826846|
|    1|  10|      0.1|1.662551327533396E-4|0.005707317073170732|0.009813779402927046|2.7809244043393444|
|    2|  10|      0.1|1.662551327533395...|0.005707317073170731|0.009813779402927046|2.7809244043393444|
|    5|  10|      0.1|1.662551327533396E-4|0.005707317073170732|0.009813779402927044|2.7809244043393444|
|    1|  10|      1.0|1.937440713557633...|0.001414634146341...|0.002318074672816667| 2.876128230743422|
|    2|  10|      1.0|1.937440713557633...|0.001414634146341...|0.002318074672816...| 2.876128230743422|
|    5|  10|      1.0|1.937440713557633E-5|0.001414634146341...|0.002318074672816...| 2.876128230743422|
|    1|  15|     0.01|0.001621143980579...|0.014292682926829265| 0.03057819862433964|  2.93001438619755|
|    2|  15|     0.01|0.001621143980579...|0.014292682926829269| 0.03057819862433964|  2.93001438619755|
+-----+----+---------+--------------------+--------------------+--------------------+------------------+
```

Figure 3: Ranking Metrics

After fitting our model with permutation of these parameters, we found the best setting on the validation set based on Precision at k and MAP. The best model was with parameters: Rank - 15, Alpha - 2 and Regularization Parameter - 0.01; where Precision at k is 0.014 and MAP is 0.0016.

### 3.2 Evaluation Results

After hyper-parameter tuning with respect to the validation set, we fit our best model to the training set and evaluate our results on the test set. The results are listed on the side:

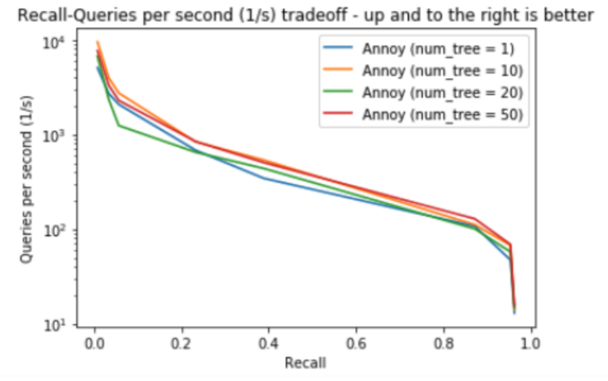


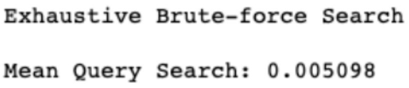
```
Test set evaluation on 1% of the data:
MAP: 0.001231
Precision at k=500: 0.011814
NDCG: 0.028696
rmse 2.789415
```

## 4 Extensions

### 4.1 Extension: Fast Search

By using spatial trees, we can accelerate the query by recursive partitioning on the dataset, reducing the time complexity from O(n) to O(log n). Annoy is extremely efficient for memory storage. After extracting latent features of the user and items matrix, we search for nearest-neighbours for a query based on these latent-factors. We compared a number of trees built in the Annoy Index. Each point on the graph denotes the total number of nodes searched at K. After plotting recall vs query-time search plot, we realised that, to increase accuracy we need to compromise on query search time if we keep the number of trees fixed. An interesting thing to notice is that the query search time almost remains the same for any number of trees. Although, with increase in number of trees, we achieve a higher recall at the same search_k value.





```
Exhaustive Brute-force Search

Mean Query Search: 0.005098
```

Figure 4: Mean query search using annoy

It takes a while for Annoy to reach an exact score of 1 for recall compared to the brute force method. This could possibly be because of the additional overhead of going through intermediate nodes as the search_k increases. Otherwise, Efficiency gain in Annoy is better than Brute Force.

### 4.2 Extension: Exploration

We use the learned representations to visualize the items and users using t-SNE.
t-SNE is a machine learning algorithm that is used for visualizing high-dimensional data.

The main idea is to take a set of points in a high-dimensional space and find an accurate representation of those points in a lower-dimensional space, typically the 2D plane. The algorithm is non-linear and adapts to the underlying data, performing different transformations on different regions.

We tune the perplexity parameter, which describes how to balance attention between local and global aspects of data. The parameter is a rough estimate about the number of close neighbors each point has.

To visualize our data, we have tuned the perplexity parameter, taking values [2, 30, 50]

At perplexity = 2, local variations dominate.

At perplexity = 30, the basic topology is displayed correctly

At perplexity 50, the outer group becomes a circle, as the plot tries to depict the fact that all its points are about the same distance from the inner group.

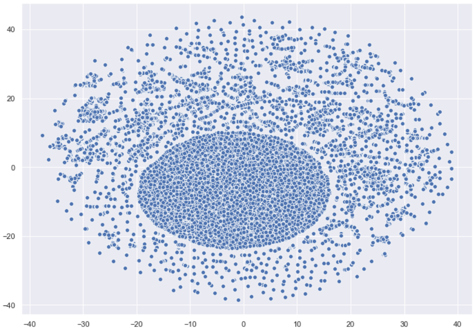


Figure 5: Item, perplexity = 50

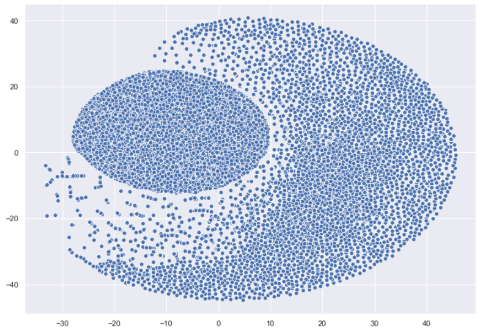


Figure 6: Item, perplexity = 2

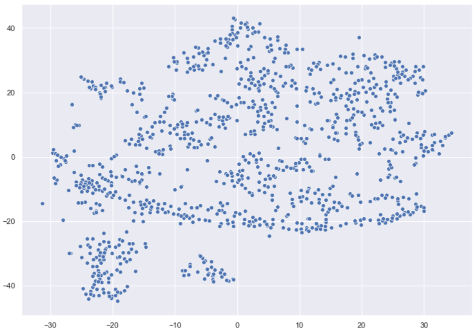


Figure 7: Users, perplexity = 30 (default)

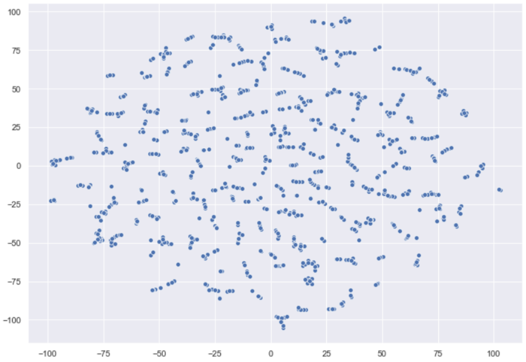


Figure 8: Users, perplexity = 2

## 5 Student Contribution

Gunjan worked on Data Processing, Splitting and ALS model. Parthvi worked on Hyperparameter Tuning and Annoy. Anusha worked on Exploration and LaTeX documentation.

All members contributed equally towards the content in the report.

## 6 References

1. https://towardsdatascience.com/my-journey-to-building-book-recommendation-system-5ec959c41847
2. https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pcaand-t-sne-in-python-8ef87e7915b
3. https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1
4. https://spark.apache.org/docs/latest/api/python/pyspark.ml.html