Recommender Systems on Amazon Data

Jaehyuk Choi
Data Science Track
SpringBoard
jaehyuk0325@gmail.com

I. INTRODUCTION (PROBLEM STATEMENT)

E-commerce, an activity of buying or selling items (products) on online services, has been populated. The business transformation has been radical and the COVID pandemic accelerated it. Many companies like JCPenney, J. Crew, CMX Cinemas, etc. filed for bankruptcy 2020. One of the causes could be poor strategies of user browsing. Not to mention that targeting online users was not the main run, users of these companies may have explored thousands of the items that they are not interested in, wasting time. Attracting online users is crucial to business since the pandemic. Users may want a better virtual shopping environment for time efficiency and convenience. To solve this issue, many of commerce websites are using recommender systems to maintain customer satisfaction and revenue. In this report, I introduce the recommender systems and its impact on business to optimize a business reach for its full potential.

Recommender systems enhance E-commerce sales in three ways; Browsers into users, Cross-selling, and Loyalty.

Browsers into users: Everyone has different preferences. Increasing options is not always great for users' decision-making. Recommender systems can help users find the items that they want to buy depending on personalization and customization.
Cross-selling: From the customers meta data, the user-item interaction patterns can be learned by recommender systems. This will improve cross-selling by suggesting the additional items that users were positively thinking of.
Loyalty: Tracking and predicting user behavior maintains old users with presence of relevant and helpful information.

Fig. 1. Impact of Recommender System on Business

Inspired by what recommender systems can bring to your business, I applied recommendation algorithms to Amazon dataset which contains items, customer reviews, and ratings accompanying metadata.

II. DATASET

The dataset[1] includes reviews "ratings-only" (ratings, text, helpfulness votes) as a csv file and item metadata "5-core" (descriptions, category information, price, brand, and image features) as a json file. The dataset is categorically divided; for this project, category of "Video Games" was chosen. For the practical purpose, item metadata of json file was loaded.

NOTE: "5-core" (i.e., dense subsets): These data have been reduced to extract the 5-core, such that each of the remaining users and items have 5 reviews each.

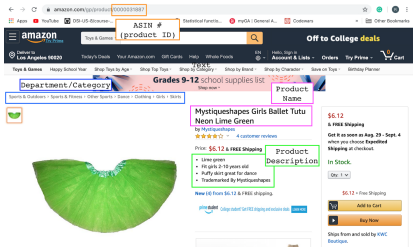

Fig. 2. Example of Key Words from json file on the Amazon webpage[2]

Sample Review:
{
"reviewerID": "A2SUAM1J3GNN3B",
"asin": "0000013714",
"reviewerName": "J. McDonald",
"vote": 5,
"style": {"Format:": "Hardcover"},
"reviewText": "I bought this for my husband who plays the piano. He is having a wonderful time playing these old hymns. The music is at times hard to read because we think the book was published for singing from more than playing from. Great purchase though!",
"overall": 5.0,
"summary": "Heavenly Highway Hymns",
"unixReviewTime": 1252800000,
"reviewTime": "09 13, 2009"
}

Fig. 3. A Sample Review of Amazon Dataset

where

- reviewerID: ID of the reviewer, e.g. A2SUAM1J3GNN3B
- asin: ID of the product, e.g. 0000013714
- reviewerName - name of the reviewer
- vote: helpful votes of the review
- style: a disctionary of the product metadata, e.g., "Format" is "Hardcover"
- reviewText: text of the review
- overall: rating of the product
- summary: summary of the review
- unixReviewTime: time of the review (unix time)
- reviewTime: time of the review (raw)
- image: images that users post after they have received the product

A. Data Wrangling

The main purpose of Data Wrangling for this project was extracting the valid data. The extracted data was over 5 years (2014 - 2018); assumption that data over 5 years is sufficient to find the user behavior on items. Merging the metadata and rating data with data imputation was needed for the EDA process. Data type-conversion and categorization was executed for a better analysis.

The following steps are done for the preprocessing.

1) Merge Metadata and Rating data
2) Data Imputation
3) Valid Data Extraction
4) Data Type-conversion
5) Data Categorization

B. Exploratory Data Analysis

The number of reviews for Amazon game data over 5 years was analyzed from 2014 to 2018. Fig 4 and Fig 5 show the yearly trend of reviews.

Fig. 4. Yearly Reviews over 5 years

Overall, rising trend is seen between 2014 and 2015, but falling trend from 2015 to 2018. The number of reviews over winter season tends higher than that over summer season. It is insightful that weather and holiday season affects the number of reviews.

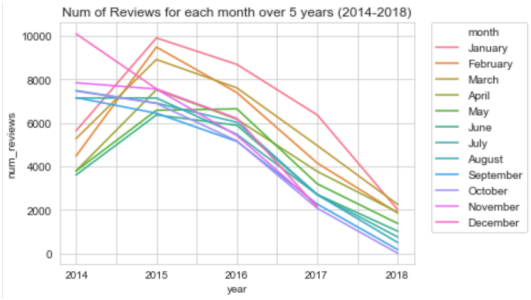

Fig. 5. Yearly Reviews per Month over 5 years

The number of reviews over time was nomthly analyzed from 2014 to 2018; Fig 6.

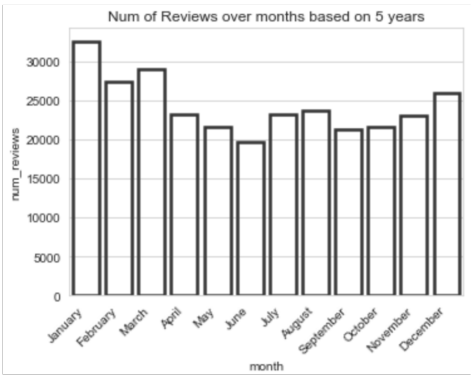

Fig. 6. Yearly Reviews per Month over 5 years

As expected, winter season has more reviews, but compared to that, the academic years has less reviews cause it improved during summer time.

Rating is a discrete range from 1 to 5 in unit 1. It is important to see the trend of ratings over real purchasers and non-purchasers and filter out reviews of non-purchasers because the reviews from real purchasers could be more credential. Fig 7 shows the number of ratings in two categories; verified real purchasers or not.

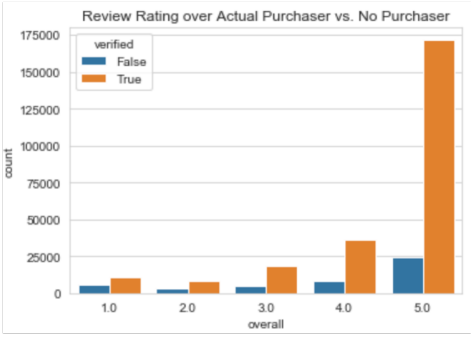

Fig. 7. Reviews of Real purchasers vs. No purchasers distributed in Rating values over 5 years

Every user has different criteria of giving a rate on

an items. It may be accurate if the mean of ratings per user is accounted for recommendations. Fig 8 shows the number of mean of ratings on rating scale.

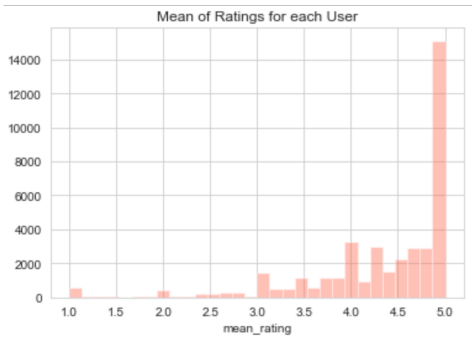

Fig. 8. Yearly Reviews per Month over 5 years

The number of ratings over rating values is shown in 7 and the number of mean of ratings over rating values is seen in 8. This data distribution over number of ratings and mean of ratings are seen in 9.

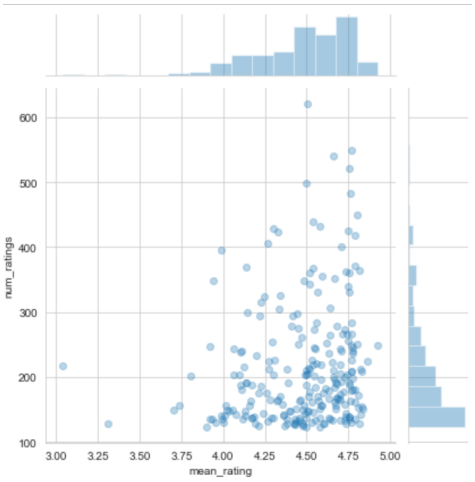

Fig. 9. Yearly Reviews per Month over 5 years

Every review is voted if they are helpful for other users. There are some outliers greater than 1500 votes, but mostly lie below 250 votes. Fig 10 shows the number of votes for reviews.

Fig 11 shows popularity trend over the items. Here popularity is defined by number of ratings per each item. Based on popularity definition, items were reranked in the descending order of popularity. Popularity itself does not help for personalization or customization. Personal preference is added later with recommendation algorithms later.

Then, Pareto Principle 20% was applied to filter out the unpopular items such that *top*20% of items

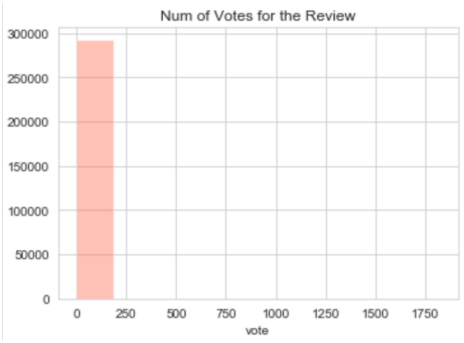

Fig. 10. Yearly Reviews per Month over 5 years

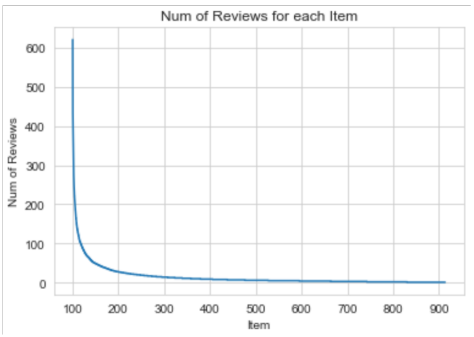

Fig. 11. Yearly Reviews per Month over 5 years

would give a better pattern of user behavior. The items that have more than 124 reviews were about 20%. That is total 49535 data with 23789 users and 241 items.

Fig 12 shows top-20 popular items with number of review ratings each item has.

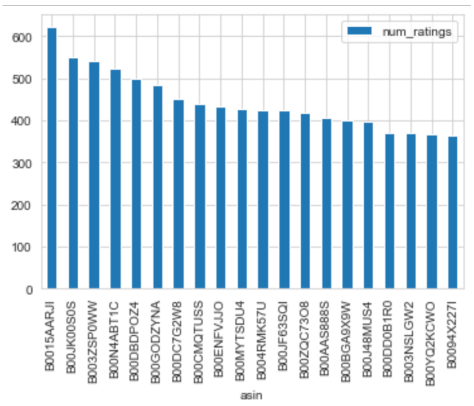

Fig. 12. Popularity Trend of Amazon Game items

III. MODEL PERFORMANCE COMPARISON

The CF algorithms that I used is basically to predict the empty cells (rating of the item that current user has not rated) in the user-item matrix. Then, the problem is defined as a regression task. The following list shows

the CF models used for this project.

A. Models

1) KNN (K-Nearest Neighbor)
2) KNN with Means
3) KNN with Z-score
4) SVD (Singular Value Decomposition)
5) SVD++
6) NMF(Non-negative Matrix Factorization)
7) customized NMF1 (Neural Matrix Factorization)
8) customized NMF2
9) customized NMF3

B. Model Evaluation Metrics

To compare the model, evaluating method is important. Depending on a recommendation task, the task can be either regression or classification. If prediction of user's potential ratings is the point, then the task is regression. If user likes an item or not, the the task becomes binary classification. In this project, we want to see the level of user preference on each item. So, a regression task comes.

For regression task, RMSE or MAE is popular evaluation metric. However, In recommendation, recommending items for user preference within top-k recommendation list is more important recommending 5,000 items. That's why precision@k and recall@k metrics come.

1) Rank-less Recommendation Metrics
   - Precision@k
   - Recall@k

   Rank-less evaluation metrics does not account for ranking (popularity). There are some other metrics based on ranking. It is called Rank-aware evaluation metrics.

2) Rank-aware Recommendation Metrics
   - MMR (Mean Reciprocal Rank)
   - mAP@k (mean Average Precision@k)
   - nDCG@k (normalized Discounted Cumulative Gain@k)

   These evaluation metrics penalize lower ranks. Since I filtered top 20% popular items, I did not use these metrics.

The following metrics shows how a model can be evaluated.

- RMSE: Root Mean Squared Error. Lower value is better.
- MAE: Mean Absolute Error. Lower value is better.
- Precision@k: number of recommended items that are actually rated within top-k over number of top-k. Higher value is better.
- Recall@k: number of recommended items that are relevant over number of recommended items. Higher value is better
- HR: Hit Rate; how often we are able to recommend a left-out rating. Higher is better.
- cHR: Cumulative Hit Rate; hit rate, confined to ratings above a certain threshold. Higher is better.
- Coverage: Ratio of users for whom recommendations above a certain threshold exist. Higher is better.
- Diversity: $1 - S$, where S is the average similarity score between every possible pair of recommendations for a given user. Higher means more diverse.
- Novelty: Average popularity rank of recommended items. Higher means more novel.

$$Prec@k = \frac{\text{\# of Relevant Items Recommended in top-k}}{\text{k (\# of Items that Recommended)}}$$

$$Recall@k = \frac{\text{\# of Relevant Items Recommended in top-k}}{\text{\# of Relevant Items}}$$

$$HR = \frac{\text{\# of Hit}}{\text{\# of Users}}$$

$$cHR = \frac{\text{\# of Hits of ratings above threshold}}{\text{\# of Users}}$$

$$Cvg = \frac{\text{\# of Users above threshold}}{\text{\# of Users}}$$

$$Dvr = 1 - S$$

$$Nvl = \sum_{i=0}^{k} \frac{\text{Rating(i) that current User gave}}{\text{Rank(i)}}$$

Where *hit* is the item in the recommendations that was actually rated by current user, S is the average similarity score between every possible pair of recommendations, Rating(i) is the rating of i-th item in the recommendations, and Rank(i) is the rank of i-th item.

Now let's look at the performance compared on the evaluation metrics introduced above. The comparison in RMSE and MAE is graphically shown in Fig 13.

Based on RMSE measurement, SVD and SVD++ are the best models (0.935 and 0.938 respectively).

For MAE, except high value of NMF, performances of other models are similar; around 0.635 to 0.655. The best one in MAE is KNN with means as 0.638.

Fig 14 shows the comparison of models in Precision@10 and Recall@10 respectively.

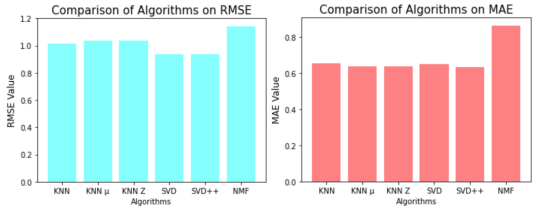

Fig. 13. RMSE and MAE for Model Performance Comparison on Amazon Game Data

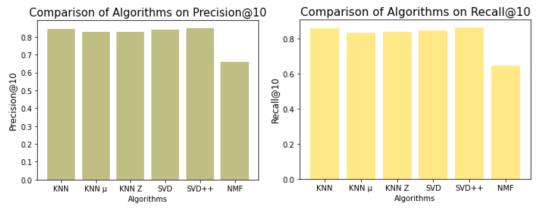

Fig. 14. Precision@10 and Recall@10 for Model Performance Comparison on Amazon Game Data

The trend is the same for both graphs; except low value of NMF, there are small gaps in performances of other models; range from around 0.830 to 0.850. The best one in Precision@10 and Recall@10 is SVD++ as 0.850 and 0.864 respectively.

Fig 15 shows the comparison of models in HR (Hit-Rate) and cHR (cumulative Hit-Rate).

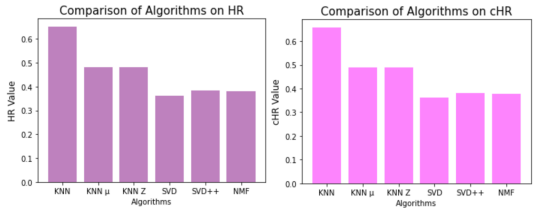

Fig. 15. HR and cHR for Model Performance Comparison on Amazon Game Data

In HR and cHR, KNN series are better than other models. The best one in HR and cHR is naive KNN as 0.652.

Fig 16 shows the comparison of models in Coverage, Diversity, and Novelty

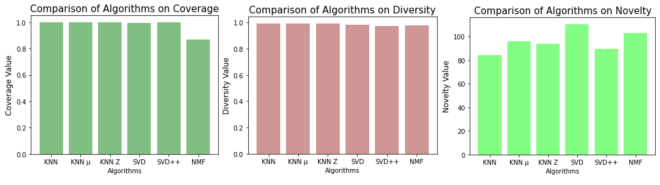

Fig. 16. Coverage, Diversity, and Novelty for Model Performance Comparison on Amazon Game Data

For Coverage, most models have high values of Coverage even though NMF is slightly lower. Similarly, most models have high values for Diversity metrics as well. It means that pair of items has no strong similarity because of low average similarity score between every possible pair of recommendations. For Novelty, KNN is the lowest and SVD is the highest. It was expected that low Novelty for KNN due to its high HR and high Novelty for SVD due to its low HR.

Regarding only RMSE, the model that shows the best performance was SVD and SVD++. Since the parameter space of a learning model may include unbounded value spaces for certain parameters, manually set bounds and discretization were executed with GridSearch on those two models. Then, SVD and SVD++ resulted 0.924 and 0.925 respectively for RMSE (0.647 and 0.648 respectively for MAE) with the parameter of 25 epoches, 0.01 learning rate, and 0.4 regularization rate (λ).

Model with GridSearch	RMSE	MAE
SVD	0.924	0.647
SVD++	0.925	0.648

TABLE I
EVALUATED MODEL PERFORMANCE BASE ON F1-SCORE

We have been explored CF models, and best models in RMSE were based on Matrix Factorization. Then, what about comparing the best CF model in RMSE with N-MF (Neural Matrix Factorization)? By using Keras, I created three N-MF.

N-MF1 is a simple matrix factorization. This initiated with user-latent and item-latent. These latent matrices were transformed into embeddings. Finally flattening them and dot product of these two were the end of this model. Fig 17 shows the Graphical representation of N-MF1.

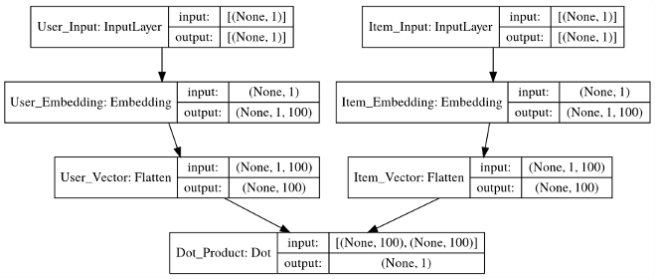

Fig. 17. Neural Matrix Factorization 1 by Keras

Fig 18 shows the graph of the training loss vs. test loss over the number of epochs. This results 3.254 of RMSE on testset; 0.311 of MSE for training loss and 10.588 of MSE for test loss. Poor performance on both trainset and testset. Moreover, it worsened on testset, outputing over 10 of MSE.

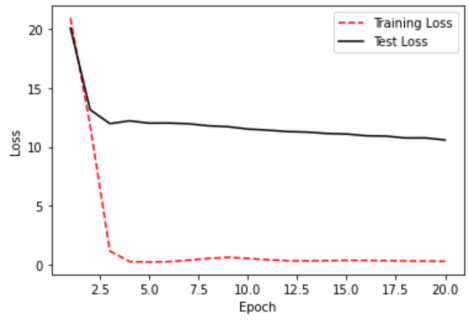

Fig. 18. Training Loss vs. Test Loss for N-MF1

Since N-MF1 was not quite successful (underfitting), I focus on increasing variance of a model. Thus, I set up "He" Normal initialization to prevent the local minimum due to potential non-convex optimization problem for each embedding and lowered regularization rate ($1e-9$ for L2) with bias. The product of two embeddings was concatenated with biases. That was flattened with two additional Fully Connected layers at the end with ReLU and Batch Normalization. Fig 19 shows N-MF2.
NOTE: ReLU works better with "He" initialization.
Fig 19

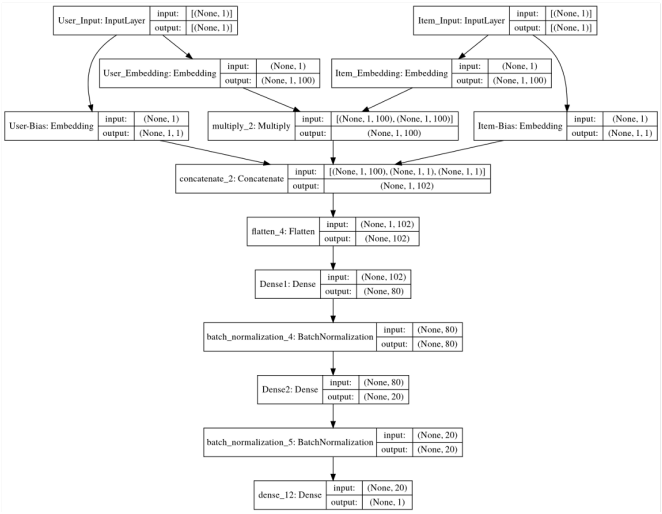

Fig. 19. Neural Matrix Factorization 2 by Keras

As expected, the model was overfitting; This results 1.008 of RMSE on the testset; 0.0358 of MSE for training loss and 1.017 of MSE for test loss. There was a gap of performance between trainset and testset. Fig 20 shows training loss vs. test loss over number of epoch for N-MF2.

With keeping "He" normal initialization and low L2 regularization rate ($1e-9$), I wanted to check with concatenation method for the user-latent and item-latent to contain their information better. I concatenated two embeddings instead of dot product of them and added four Fully Connected layers with ReLU and Batch Normalization. Fig 21 shows N-MF3.

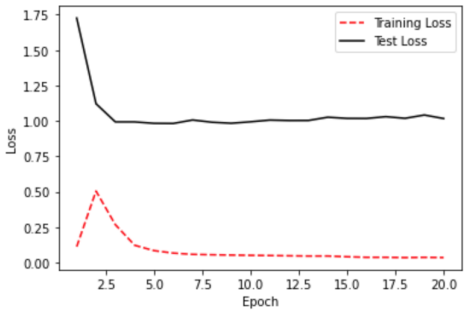

Fig. 20. Training Loss vs. Test Loss for N-MF2

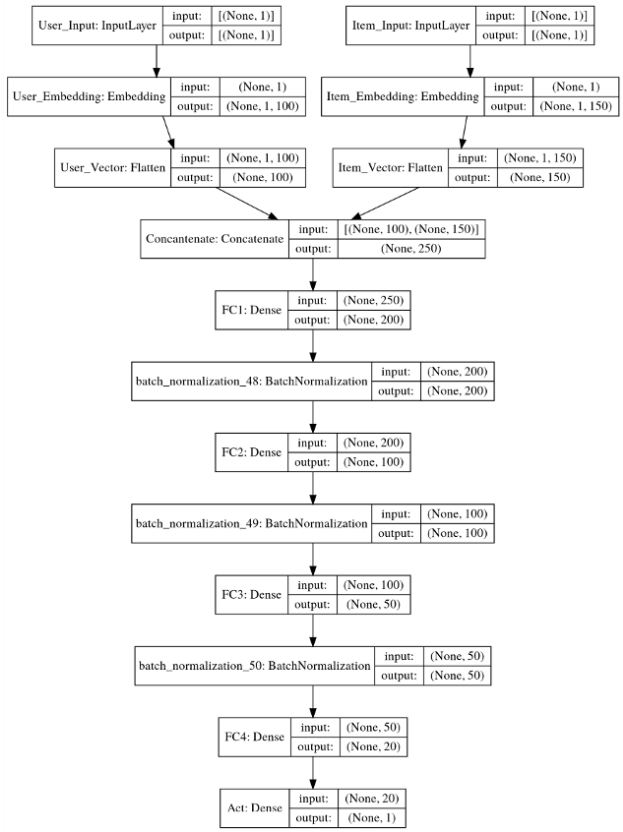

Fig. 21. Neural Matrix Factorization 3 by Keras

N-MF3 results 1.011 of RMSE which was higher than that of N-MF2. However, its training loss was 0.0851 of MSE and test loss was 1.0217 of MSE. The gap of performance between trainset and testset was reduced, so the model was more generalized than N-MF2. Fig 22 shows training loss vs. test loss over number of epoch for N-MF3.

IV. RESULTS / CONCLUSION

The Table II shows evaluated CF model performances with various metrics.

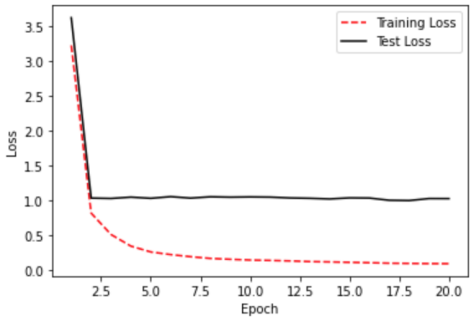

Fig. 22. Training Loss vs. Test Loss for N-MF3

Model	RMSE	MAE	Pr@10	Re@10	HR	cHR	Cvg	Dvs	Nvl
KNN	1.012	0.656	0.843	0.857	0.652	0.659	1.0	0.991	84.2
KNN μ	1.037	0.638	0.827	0.835	0.481	0.488	1.0	0.993	96.0
KNN Z	1.037	0.641	0.829	0.836	0.480	0.488	1.0	0.993	94.1
SVD	0.935	0.651	0.838	0.848	0.360	0.361	0.99	0.980	110.5
SVD++	0.938	0.634	0.850	0.864	0.384	0.381	1.0	0.970	89.6
NMF	1.143	0.866	0.659	0.646	0.382	0.379	0.87	0.977	103.0

TABLE II
COMPARISON OF CF MODEL PERFORMANCE

Among these models, SVD and SVD++ were the best in RMSE and MAE. In Precision@10k and Recall@10k, performance of CF models except NMF(Non-negative Matrix Factorization) were hard to compare. Probably, there are features that must be accounted as real number (including both positive and negative numbers) for users and items. Except that, Matrix Factorization technique is a good way to tackle the problem. In HR and cHR, KNN was the best. Overall, most models have good Coverage and Diversity. But in Novelty, SVD seems too high.

The Table IV shows evaluated model performances with RMSE.

Model	RMSE
KNN	1.012
KNN μ	1.037
KNN Z	1.037
SVD	0.935
SVD++	0.938
NMF	1.143
N-MF1	3.254
N-MF2	1.008
N-MF3	1.011

TABLE III
COMPARISON OF MODEL PERFORMANCE IN RMSE

For further understanding of the impact of MF (Matrix Factorization) on Neural Network, customized MFs were created. Whreas N-MF1 had poor performance on both trainset and testset (underfitting), N-MF2 was overfitting. N-MF3 has similar RMSE value to that of N-MF2, but generally performs better with a lower gap between trainset and testset.

For targeting conservative users, I would choose KNN with the highest HR and cHR. For targeting the users such that best accurate prediction with their own patterns, SVD++ or N-MF3 can be chosen.

In future work, the N-MF models can be evaluated with other metrics including Rank-less and Rank-aware metrics. NOTE: Rank-less with Precision@k, Recall@k, and Hit@k. Rank-aware with MAP(Mean Average Precision) and nDCG (normalized Discounted Cumulative Gain) with penalizing rank. Moreover, comparison of training time can bring more values on this project.

REFERENCES

[1] https://nijianmo.github.io/amazon/index.html
[2] https://towardsdatascience.com/web-scraping-recommender-systems-project-1d360fa678e4