COMP9417 Machine Learning and Data Mining Project Report: Book Recommendation system using Collaborative Filtering

Chenyi Cao (z5192583)

Han Yue (z5151083)

Haoling Wu (z5139842)

Zanlai Hu (z5136463)

Date of submission: 2019 – August - 11

INTRODUCTION

Have you ever wondered how Spotify could recommend the songs precisely to your taste, how Amazon could push the product exactly you interested in? The techniques behind are recommendation systems. Recommendation systems are information filtering systems targeting to predict end users' preferences and generate recommendations accordingly (Ricci, Rokach and Shapira, 2015). In general, collaborative-based and content–based approaches are the basic methods to produce recommendation systems. Hybrid systems combining of two or more approaches are commonly used for current recommendation systems. The primary objective of this project is to design a book recommendation system using collaborative filtering which recommends users appropriate books according to their book-rating records. The data set this project worked on is the BookCrossing (BX) dataset collected by the GroupLens ("Book-Crossing Dataset", 2004). Model-based collaborative filling with machine learning techniques are built and ranked with prediction accuracy. Final recommendations are made refer to prediction result made by Funk SVD (Funk, S. (2016)) algorithm in Surprise (NicolasHug, Surprise), with optimized parameters using Grid Search, as well as strategies to deal with cold-start issue.

KEY METHODS

This project utilizes collaborative filtering method to generate recommended items to a user which had been chosen by other users with similar taste (AGGARWAL, 2018). There are mainly two methods of collaborative filtering: memory based and model based (COMP9417, 2015).

Memory -based collaborative filtering

Memory-based collaborative filtering predicts unknown rating $r_{c,s}$ for user *c* for item *s* by aggregating the ratings of *N* users c' of highest similarity to *c* and *s* ratings (COMP9417, 2015) with weighted sum, represented as:

Equation 1. Memory-based CF (p7, COMP9417, 2015)

$$r_{c,s} = k \sum_{c' \, in \, C} sim(c, c') \times r_{c',s'}$$

At most k similar users c' of user c are used for predicting $r_{c,s}$ are top-N users based on similarity. Memory-based collaborative filtering uses the rating matrix according to user-IDs, item-IDs and ratings. The recommendations are made by recommending top-n rated items by top-k similar users(user-based) or items (item -based).

Model – based collaborative filtering

Model-based collaborative filtering uses other machine learning methods to build a model for predicting unknow rating $r_{c,s}$ for user *c* for item *s* (COMP9417, 2015). Collaborative filtering prediction algorithms with machine learning are commonly k-NN based algorisms, Matrix Factorization-based algorithms, Deep Learning algorithms.

Surprise python scikit is used to apply and test machine-learning prediction algorithms, the module and algorithm backgrounds are as following ("Getting Started — Surprise 1 documentation", 2019). The information of Surprise built-in modules and their algorithms are provided as below.

K-NN based collaborative filtering algorithms (KNNBasic, KNNWithMeans, KNNWithZScore) are directly derived from the basic nearest neighbors' approach (Koren, Y. (2019)). All KNN modules are **user-based** collaborative filtering by user-similarities.

Matrix Factorization-based collaborative filtering algorithms (SVD, SVD++) are derived based on factorizing a rating matrix with different approaches. SVD stands for FunkSVD Singular Value Decomposition (Funk, S. (2016)) algorithm. NMF stands for Non -negative Matrix Factorization (Luo, Zhou, Xia & Zhu, 2014).

Co-clustering collaborative filtering algorithm is a directly implementation of clustering method from the referenced paper (Thomas, G & Srujana, M. (2005)) by generating bi-clusters/co-clusters as a subset of rows which exhibit similar behavior across a subset of columns from Matrix Factorization (Co-Clustering, ML Wiki).

Cross validation function from Surprise is implemented by running a cross validation procedure with specified folds for a given algorithm, reporting accuracy measures and computation times. Parameter optimization is applied by using GridSeachCV, which computes accuracy metrics from Grid Search algorithm on various combinations of parameters, over a cross-validation procedure (Surprise, (2018)).

The basic process used for prediction algorithm modules in Surprise is: Read data to its required format (df [‘user_id, ‘item_id’, ‘ratings’]), compute similarity matrices (only if KNN called) or dataset to algorithms. Run algorithms. Predictions are made by training with whole data and predict ratings from resulting of algorithm computed. Or train and test the algorithm with train and test sets split from whole data with specified sizes (a single fold of cross validation), accuracy tests of RMSE, MAE are also built-in.

IMPLEMENTATION

The implementation of recommendation system to Book-Crossing dataset is performed with respect to Knowledge Discovery in Database (KDD) process (COMP9318, 2019).

Data Selection

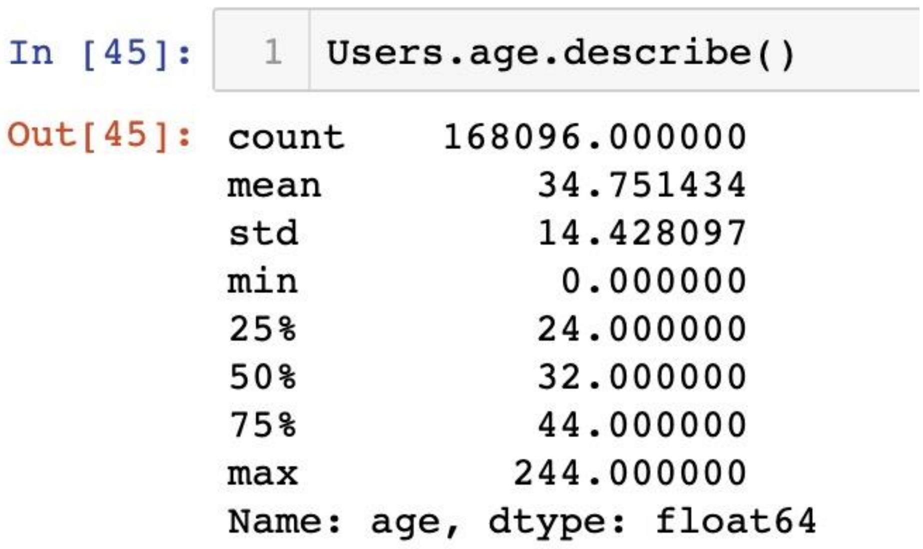
This project builds book recommendation systems from Book-Crossing dataset collected in 2004 ("Book-Crossing Dataset", 2004). There are three tables: BX-Users with attributes 'User-ID', 'Location', 'Age' information of BX users; Bx -Books contains attributes 'ISBN'(bookID), 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher' information for books, 'Image-URL-S', 'Image-URL-M', 'Image-URL-M' URLs for book cover images; and BX -Book-Ratings contains attributes 'Book-Rating' on a scale **explicit rating** 1-10, and **implicit rating** of '0', as well as 'User-ID' and 'ISBN'. Unavailable data is represented as np.NaN.

The tables are downloaded with csv format. Pandas, Numpy, Matplotlib, Seabon are imported for reading, plotting and processing datasets. The tables of users, books, and ratings are read to Pandas data frame accordingly.

Data clean and preprocessing

It is necessary to preprocess these data frames with data clean by: 1. Check invalid and missing data. 2. Remove/replace invalid and missing data; 2. Rescale data

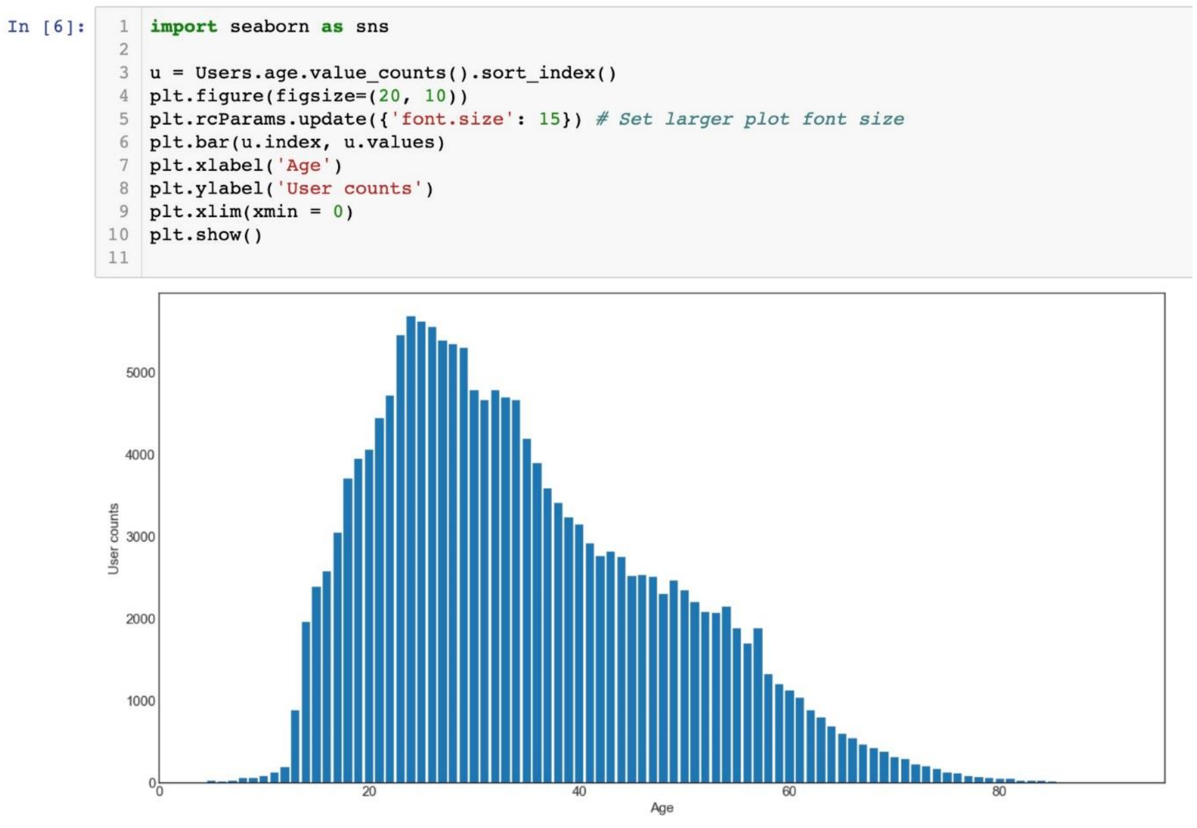
Data frame of users is cleaned by scale attributes to unique user IDs so that all attributes get the same number of data entries. Invalid data are replaced by np.NaN, such as, ages of users larger than 100 years old or 0 years old(Figure 1). The majority ages of users are found by making a 0.05% to 99.5% interval of cumulated user numbers to age is 3 years old to 76 years old. The ages outside 5 to 90 years old are considered as invalid ages. Invalid ages and missing ages are replace using normalization (Figure 2). To clean data for location attribute, in form of 'city, state, country' is split accordingly, so that spitted attributes all represents users from an area instead of a specific place. Missing data of city, state, country is replaced by 'other'.



```
In [45]:  Users.age.describe()

Out[45]:  count    168096.000000
          mean         34.751434
          std          14.428097
          min           0.000000
          25%          24.000000
          50%          32.000000
          75%          44.000000
          max         244.000000
          Name: age, dtype: float64
```

Figure 1 Age describe of original data



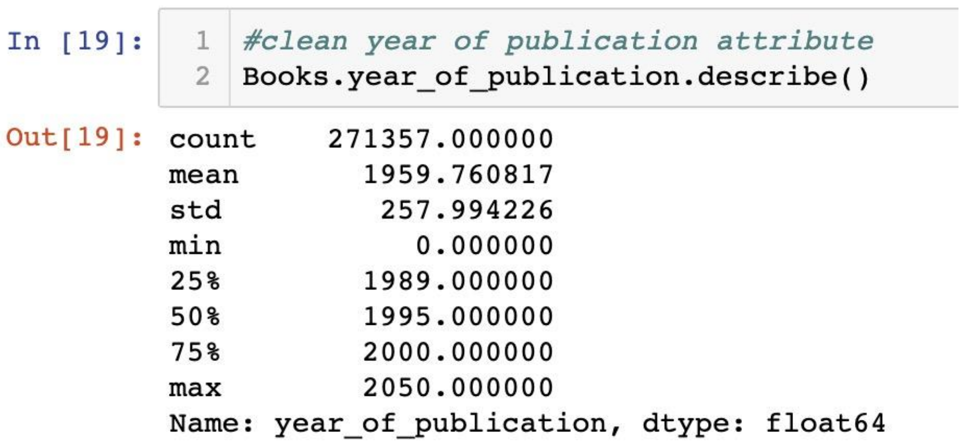
```
In [6]:  import seaborn as sns

         u = Users.age.value_counts().sort_index()
         plt.figure(figsize=(20, 10))
         plt.rcParams.update({'font.size': 15}) # Set larger plot font size
         plt.bar(u.index, u.values)
         plt.xlabel('Age')
         plt.ylabel('User counts')
         plt.xlim(xmin = 0)
         plt.show()
```

Figure 2 Age distribution after normalization

Data frame of books is also scale by unique book IDs (ISBN). Missing data are replaced by np.NaN. Although this data set was collected in 2004, there are books scheduled to publish after 2004, and very old books published in 1300s or even earlier (Figure 3). The year interval which majority books published (0.05% - 99.5% cumulative book counts), is 1963 to 2003.

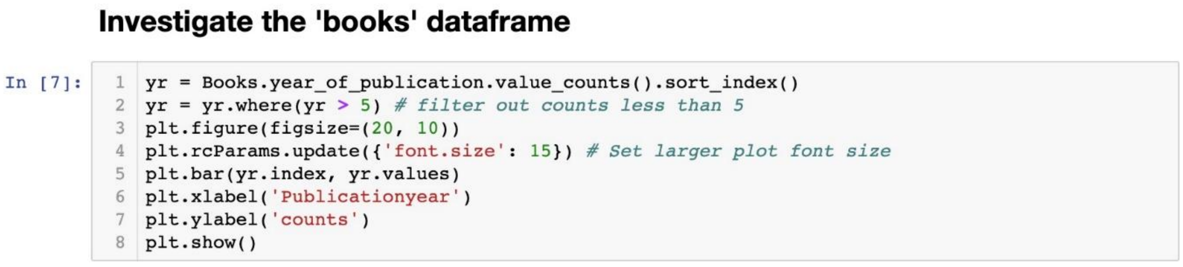
Only 0.1% of books published in year outside this interval, but this 0.1% affected the concentration of book counts according to year of publication significantly. To minimize potential impact that may affect prediction accuracy, invalid and missing years (original 'NaN' transferred to 0 as a result of 'to_numeric') are replaced by normalization. (Figure 3)



```
In [19]:  #clean year of publication attribute
          Books.year_of_publication.describe()

Out[19]:  count    271357.000000
          mean       1959.760817
          std         257.994226
          min           0.000000
          25%        1989.000000
          50%        1995.000000
          75%        2000.000000
          max        2050.000000
          Name: year_of_publication, dtype: float64
```

Figure 3 year of publication describe before cleaning



Investigate the 'books' dataframe

```
In [7]:  yr = Books.year_of_publication.value_counts().sort_index()
         yr = yr.where(yr > 5) # filter out counts less than 5
         plt.figure(figsize=(20, 10))
         plt.rcParams.update({'font.size': 15}) # Set larger plot font size
         plt.bar(yr.index, yr.values)
         plt.xlabel('Publicationyear')
         plt.ylabel('counts')
         plt.show()
```

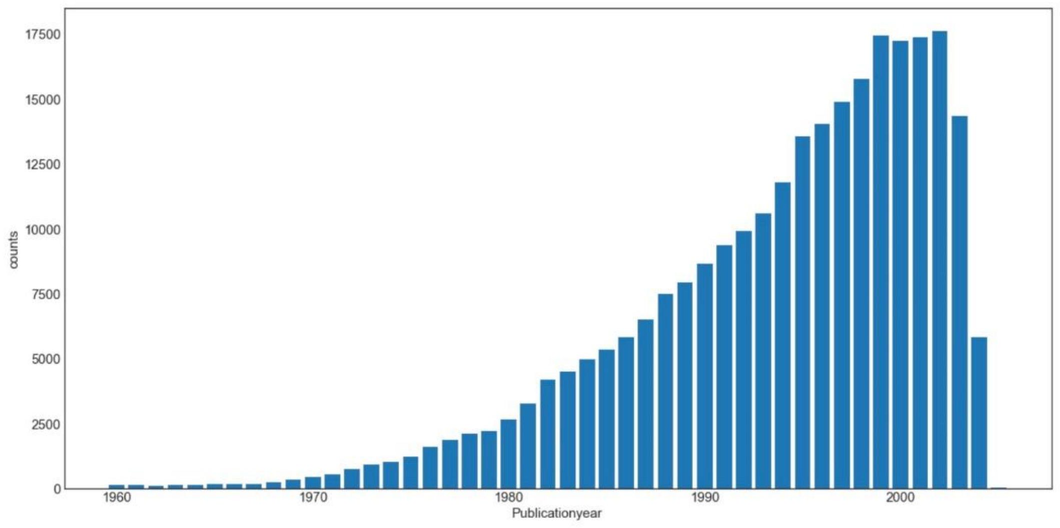


Figure 4 year of publication distribution after normalization

To clean the data frame of book ratings, for the reason implicit ratings where user does not sure how to rate. These implicit ratings need special method to deal with (eg. ALS, Bayesian). So, we only focus on explicit ratings for the algorithms we chosen. SVD++ specifically predict implicit rating from explicit rating, but it takes 10 times fit time of other algorithms.Regrading to the issue, some ISBN in book dataset, are actually pointing to the same book, books of same content published by different publishers. However, books from different publishers may affect the ratings of book, since some books were translated into different languages from publishers in different countries. We decided not to deal with the multiple ISBN to books of same content until the recommendation.

Data mining functions and algorithms

With the definition of model-based collaborative filtering, user-based and item-based models both requires pivot matrix with rows and column of user IDs and item IDs, with value of ratings. However, with Surprise built-in models for prediction, a Pandas data frame of format [‘u_id’, ‘i_id’, ‘ratings’] is required, which is the cleaned rating data frame.

Using built-in collaborative filtering algorithms of KNNBasic, KNNWithMeans, KNNWithZScore, SVD, SVDpp, NMF with default settings to train on our dataset, rank with cross validating result of RMSE, the best two algorithms will be passed to parameter-optimization using Grid Search. The algorithm with better RMSE, will be used in our recommendation system. From the resulted average RMSE, SVD is the best algorithms to use for our dataset.

Knowledge Representation

A particular example of knowledge representation is shown as below for before and after we remove the ‘Lazy Users’ who have less than 10 ratings for books.

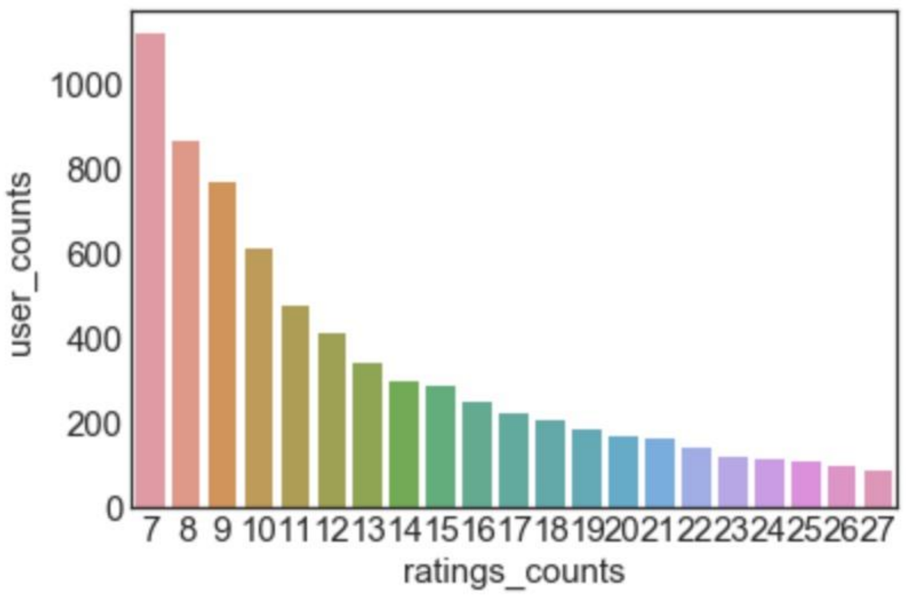


Figure 5 Rating count vs User count before cleaning

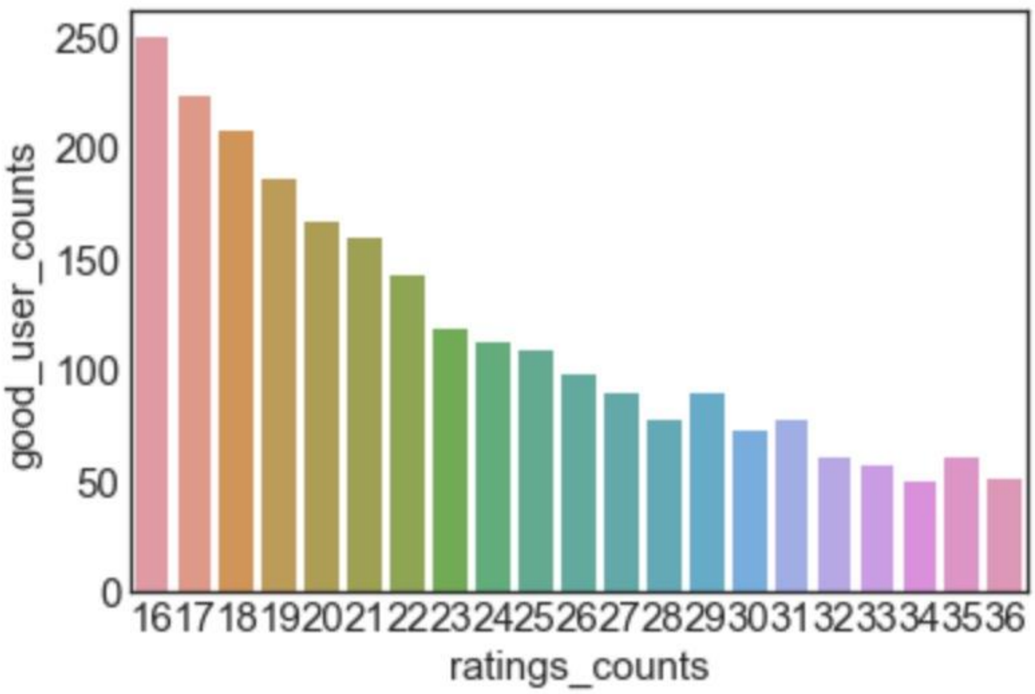
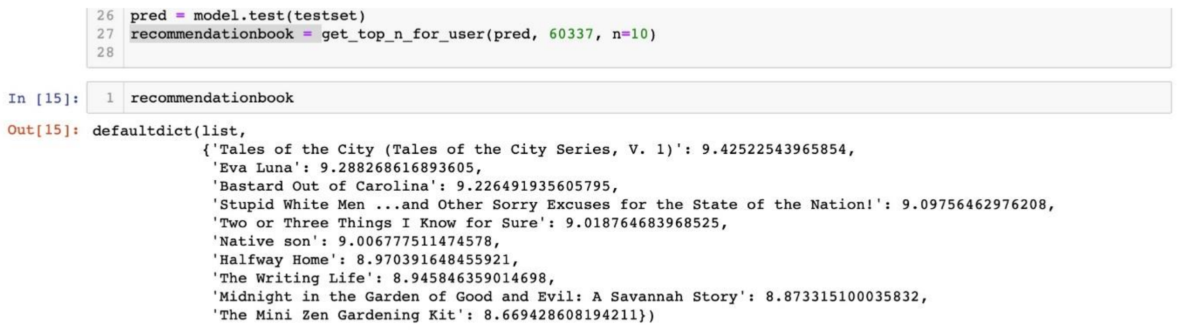


Figure 6 Rating count vs User count after cleaning

Recommendations are made based on top-10 ISBNs computed by SVD algorithm. If new user entries system, he/she will be first recommended with top-10 books according to top-10 high rated books of all books rated by users.



```
pred = model.test(testset)
recommendationbook = get_top_n_for_user(pred, 60337, n=10)

In [15]: recommendationbook

Out[15]: defaultdict(list,
                     {'Tales of the City (Tales of the City Series, V. 1)': 9.42522543965854,
                      'Eva Luna': 9.288268616893605,
                      'Bastard Out of Carolina': 9.226491935605795,
                      'Stupid White Men ...and Other Sorry Excuses for the State of the Nation!': 9.09756462976208,
                      'Two or Three Things I Know for Sure': 9.018764683968525,
                      'Native son': 9.006777511474578,
                      'Halfway Home': 8.970391648455921,
                      'The Writing Life': 8.945846359014698,
                      'Midnight in the Garden of Good and Evil: A Savannah Story': 8.873315100035832,
                      'The Mini Zen Gardening Kit': 8.669428608194211})
```

Figure 7 Example recommendations for user

RESULTS AND DISCUSSIONS

Evaluation

The most popular and basic recommendation models are collaborative filtering, which can predict things that people might like, based on the similar things or similar users. Collaborative filtering can be roughly separated into two types: Memory based approach and Model based approach. This project focus on the model-based approach.

Model based collaborative filtering system can also be separated into three types:
1. Clustering based algorithm: KNN
2. Matrix factorization-based algorithm: SVD, Probabilistic Matrix Factorization and Non-ve Matrix Factorization
3. Deep Learning: Mutli-layered neural nets (including embedding layers)

(Grover, 2017)
By using surprise library, we can directly use the built-in functions to achieve the following model based collaborative filtering:

1. NMF
2. SVD
3. SVDpp
4. CoClustering
5. KNNBasic
6. KNNWithMeans
7. KNNWithZScore

The evaluation of each models is necessary before the start of recommendation system.
By using surprise built-in function, we cross validate these seven models

RMSE	Fold1	Fold2	Fold3	Fold4	Fold5	mean
NMF	2.4747	2.4660	2.4906	2.4752	2.4650	2.4743
SVD	1.5679	1.5740	1.5716	1.5735	1.5801	1.5738
SVDpp	1.6577	1.6476	1.6470	1.6459	1.6439	1.6484
CoClustering	1.8328	1.8306	1.8361	1.8577	1.8209	1.8356
KNNBasic	1.9345	1.9486	1.9524	1.9443	1.9545	1.9469
KNNWithMeans	1.8043	1.7921	1.8082	1.7933	1.7917	1.7979
KNNWithZScore	1.7904	1.7942	1.7978	1.8016	1.7972	1.7962

Table 1 RMSE Test record

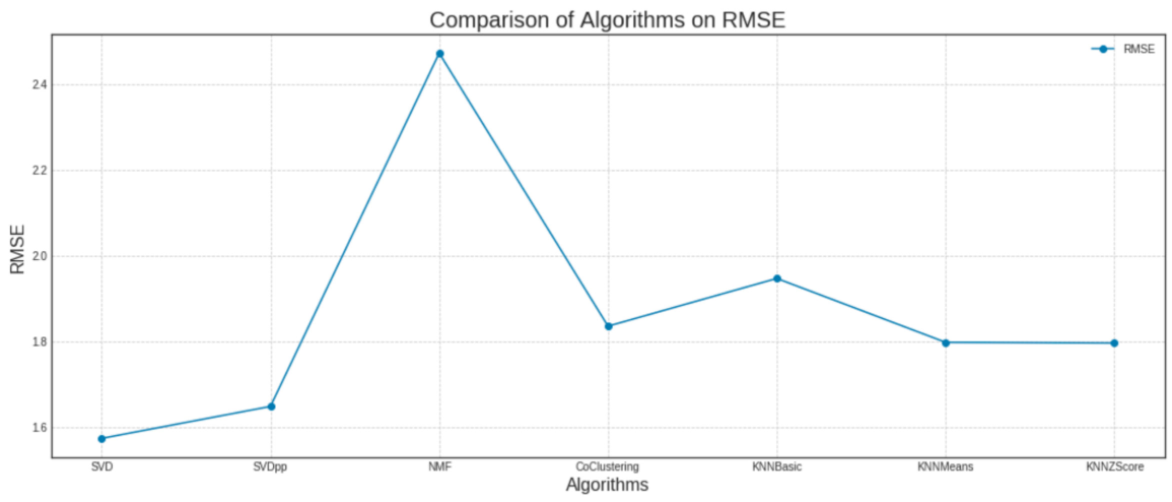


Figure 8 Mean RMSE for algorithms tested

It is clear that SVD owns the lowest RMSE which means this model can provide the most correct predictions compared to other models. Note that the training set and test set are randomly separated from data, which lead to different RMSE results in every test.

Parameters Optimization

Since SVD algorithm has the best RMSE, we began to search the best parameters that can further improve our algorithm's performance compared to default parameters.

We want to find the best pair of parameters from {'n_epochs':[20,30,40], 'lr_all': [0.001, 0.005, 0.01], 'reg_all': [0.01, 0.02, 0.04]}, where "n_epochs" is the number of iteration of the SGD procedure, "Ir_all" is the learning rate for all parameters and " reg_all" is the regularization term for all parameters.

For this project, we used GridSearchCV method from the surprise library. After applying this method on our data, we found the best parameters for our data are {'n_epochs':[30], 'lr_all': [0.005], 'reg_all': [0.04]} with the minimum RMSE (1.6299). And we used these parameters for building our model for recomanding system later.

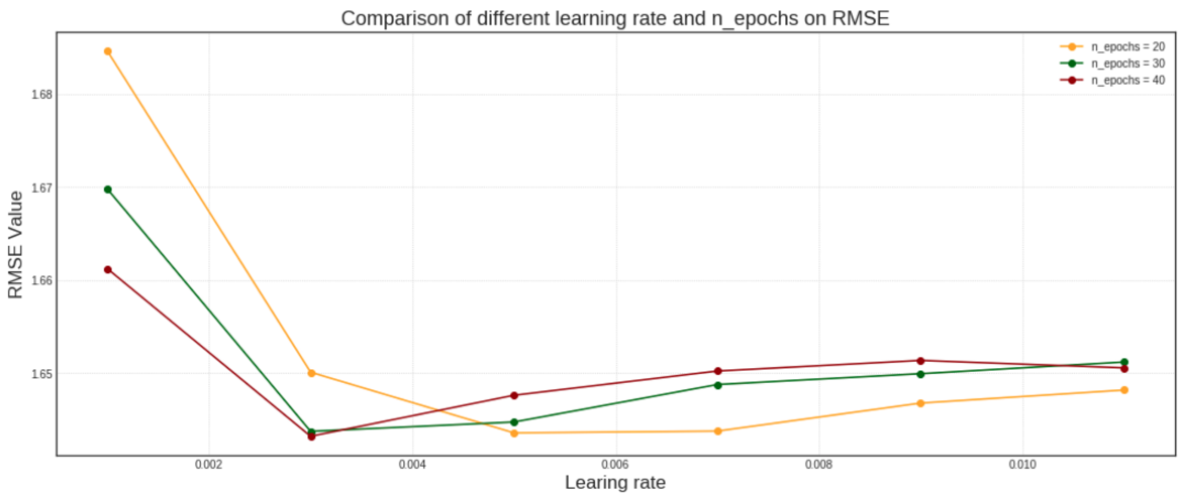


Figure 9 Comparison of learning rate and n_epochs

In the end, we also compared the impact of different learning rate and n_epochs on SVD algorithm performance. And we can clearly see that these 3 lines have similar shapes.

When we set a too small learning rate (like 0.001), the algorithm performs worse than those with higher learning rates. However, the performance become worse when we further rise the learning rates since 0.03 or 0.05(depends on different n_epochs). Apart from that, even when we set the same learning rate, the higher n_epochs(iterations of SGD) didn't lead to a better RMSE.

Apparently, we could find that these two parameters are both very important factors for stochastic gradient descent process. And only the good combination of all parameters could lead to best performance of this algorithm.

Methods

SVD is the singular value decomposition (SVD). Matrix factorization is the main part of SVD as showed below:
R=MΣU^T
The SVD will take our matrix 'data' loaded from 'user_item_rating', and gives M, Σ and U (Hug, 2017). Σ is a diagonal m × n matrix with non-negative real numbers on the diagonal. So, when we put all users and books in the matrix, there will be some empty place as none of users can read all books. Then SVD will be used to predict these empty places as the table showen below:

	book1	book2	book3
user1	?	1	1
user2	2	?	?
user3	1	?	1

The SVD model will try to predict the missing ratings.

Discussions

Python Surprise package is a very popular and good open source package. However, it can still have some problems. As a built-in package, the whole functions were written inside, which means it is hard to customize it for some specific demand. Collaborative Filtering also has its limitation. The SVD model prediction for a given data (ratings, books, users) is the dot product of the corresponding embedding vector. Therefore, it is not possible if a new item wants to be added to it to predict. Because it an item is not seen during training, the system cannot create embedding for it and cannot use the model to predict this item. One technique to solve this problem is using WALS projection. This algorithm will be added in the further study of this system.

REFERENCES

Ricci, F., Rokach, L. and Shapira, B. (2015). Recommender Systems Handbook. Boston, MA: Springer US, pp.1 - 35.

AGGARWAL, C. (2018). RECOMMENDER SYSTEMS. [Place of publication not identified]: SPRINGER.

COMP9417 Machine Learning and Data Mining Recommender Systems. (2015).

COMP9318 Introduction to Data Mining. (2019).

NicolasHug, *Surprise Python scikit*. (2019). Retrieved 9 August 2019, from https://surprise.readthedocs.io/en/stable/getting_started.html#

Book-Crossing Dataset. (2004). Retrieved 9 August 2019, from http://www2.informatik.uni-freiburg.de/~cziegler/BX/

Koren, Y. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. NJ 07932: AT&T Labs – Research. Retrieved from https://www.cs.rochester.edu/twiki/pub/Main/HarpSeminar/Factorization_Meets_the_Neighborhood-_a_Multifaceted_Collaborative_Filtering_Model.pdf

Salakhutdinov, R., & Mnih, A.. Probabilistic Matrix Factorization [Ebook]. Department of Computer Science, University of Toronto. Retrieved from http://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf

Funk, S. (2016). Netflix Update: Try This at Home. Retrieved 10 August 2019, from https://sifter.org/~simon/journal/20061211.html

Luo, X., Zhou, M., Xia, Y., & Zhu, Q. (2014). An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems. IEEE Transactions On Industrial Informatics, 10. doi: 10.1109/tii.2014.2308433

Thomas, G & Srujana, M. (2005). A scalable collaborative filtering framework based on co-clustering. 2005.

Co-Clustering - ML Wiki. Retrieved 10 August 2019, from http://mlwiki.org/index.php/Co-Clustering

Grover, P. (2017). Various Implementations of Collaborative Filtering. [online] Medium. Available at: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0 [Accessed 7 Aug. 2019].

McKenzie, T. (2018). tttgm/fellowshipai. [online] GitHub. Available at: https://github.com/tttgm/fellowshipai [Accessed 10 Aug. 2019].

Hug, N. (2017). Understanding matrix factorization for recommendation (part 2) - the model behind SVD. [online] Nicolas Hug. Available at: http://nicolas-hug.com/blog/matrix_facto_2 [Accessed 11 Aug. 2019].