Recommendation System, Models and Applications

Recommender System is a system that utilizes information filtering to predict the user's preferences of given items according to the user's former choices. Recommender systems have applications in a variety of areas including movies, books, research articles, music, and products in general. In our study, we used different kinds of models to implement recommendations on books datasets to figure out the most appropriate

Computing

Python Libraries Surprise, Sklearn, BM25, NLTK, and LightFM are used in this work.

SI671 Data Mining

YueyangZ, Xinyi Zhao & Ryan Deng

Data

- This Books Datasets mainly include 1) reader-book ratings and 2) book-tags dataset.
- This ratings dataset includes 600,0000 ratings of 10000 books by 53424 readers.

Methods

- **Sub-task1: Collaborative filtering**
- ***Based on Explicit rating values***
- 1) Item-based or User-based KNNWithMeans algorithm;
- 2) Model-based dimension reduction algorithm SVDpp.
- Evaluation: RMSE, MAE
- **Sub-task2: Content-based**
- ***Based on Books Text Features***
- 1) TFIDF(cosine similarity)
- 2) BM25+
- **Sub-task3: Hybrid System**
- ***Based on Implicit feedback & Books features***
- 1) Pure CF model
- 2) Hybrid model
- Evaluation: AUC score

Results

- **Collaborative filtering**
- SVDpp was selected to construct final model
- Test RMSE 0.8617, MAE 0.6743
- **Demo**: The books with a larger predicted rating will be recommended to users.

- **Content based recommendation**
- TF-IDF model performs better than Count model
- **Demo**: Recommend similar books based on book author and tags features.

- **Hybrid Model**
- Hybrid model beats pure CF model
- Hybrid test auc 0.95 v.s. CF test auc 0.92
- **Demo**: The books with a predicted higher rank will be recommended to users.

Recommendation System, Models and Applications

Yueyang Zhang, Xinyi Zhao, Ryan Deng
School of Information
University of Michigan
yueyangz@umich.edu, xinyiz@umich.edu, zydeng@umich.edu

Abstract

Recommender system or recommendation engine, a subclass of information filtering system, is a system that predicts the preferences of users and provides a list of recommendation items, aims to help users make appropriate decisions. In our study, we built three different kinds of recommendation engines and implement recommendations on books datasets. The three different strategies we used to build our models are collaborative filtering based on explicit rating interaction, content-based filtering based on books features, and a hybrid recommender based on both implicit rating interaction and books features. Specific evaluation methods have been used to judge the accuracy of each of these methods. Our result shows that both collaborative filtering and content-based filtering can provide good prediction, and hybrid recommender system could give higher quality recommendation.

1 Introduction

Recommender System is a system that utilizes information filtering to predict the user's preferences of given items according to the user's former choices. This system usually gives users a list of recommendation products they might be interested in, or informs users of how much the user would be in favor of these items, that would help users to find and decide on appropriate items. Recommender systems have applications in a variety of areas including movies, books, research articles, music, and products in general. With a single input or multiple inputs across platforms such as search queries, these systems will operate and return the specific recommendations for each unique user. One of the examples is the DVD rental provider Netflix displayed the predicted rating for each movie in order to help customers to decide with movies to order. Another example is the book retailer Amazon, the predicted book ratings and a list of other books that are bought by customers who buy a specific book are provided to users.

In our study, we used different kinds of models to implement recommendations on books datasets to figure out the most appropriate recommendation strategies. Using python library *Surprise*, We first applied user-based and model-based collaborative filtering based on K-Nearest Neighbours(KNN) and model-based collaborative filtering based on the Singular Value Decomposition(SVD), SVD extension and NMF (Non-negative Matrix Factorization). For these models, we used RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) to evaluate the performances and we found that KNNWithMeans method should be the best KNN algorithm and SVDpp should be the best model-based algorithm. Then we built content-based filtering recommendation engines by extracted features including books' titles, authors, tags, number of ratings and the average ratings. TF-IDF and Count methods were applied to vectorize the text information for the following cosine similarity calculation. We also added the BM25+ model to address the limitation of the search engine for this recommendation system. Finally we built the hybrid model based on python library *LightFM* based on transformed implicit rating interactions and books features. We compared pure collaborative filtering and hybrid model constructed with LightFM and drew to the conclusion that hybrid model yields higher AUC score on test data.

In this project, for CF strategy, we finally built a recommendation demo which can return recommended books for every users. For Content based model, we built a recommendation demo which can return self defined number of recommended books for a given books based on similarity. For hybrid model, we built a recommendation demo which can recommend new books to a given user id based on both rating interactions and books features.

2 Related Works

Recommender systems are currently found in lots of modern applications that provide the user with a large number of candidate products or items. Recommendation services are diverse based on their tasks. The most common take is to predict the users' opinion (eg. ratings, stars, etc) on some items, and recommend the items with higher scores to users. The collaborative filtering is the commonly used method for this task.[9] The second task is to recommend a set of interesting or useful items to users based on the genre, brand and other features of the user's order history. This is called content-based recommendation. For some complicated cases, either the collaborative filtering or the content-based recommendation are not sufficient to completely solve the problem, then the hybrid recommendation can be used to combine the collaborative filtering and content-based recommendation together. The third task is to provide a set of recommendations that will optimize the companies' revenue, this task is highly important for some e-commerce websites. We don't plan to utilize this method because it does not fit our case.

Collaborative filtering is proved to be one of the most effective approaches by its simplicity in rationale and implementation.[4][18] There are two basic collaborative filtering, user-based CF and item-based CF.[8][6] Use-based CF assumes that the good way to recommend items to the user is to find the orders of other users with the similar preferences of this user. So it will find the user's neighbors by similarity and predict the user's rating on some new items based on the similarity matrix.[16] The item-based CF analyzes a set of items and their ratings by the active user, then a new set of items will be predicted by the similarity matrix.[20] One of the challenges for collaborative filtering is sparsity. Although there are so many active users, there are still large amounts of null ratings in the user-item database, which leads to the inaccuracy of the prediction. One of the solutions is dimensional reduction.[2] This method is also used in our study.

Content-based filtering is based on the features of an item in a user's order history or past activity, the items with similar features can be recommended to users. The recommender system needs to decide between two ways for information delivery, the Exploitation and Exploration. Exploitation means that the recommender selects documents similar to the document that the user has already expressed a preference for. Exploration means the recommender selects documents which the user record does not provide evidence to predict the user's reaction. In addition, there are two methods to vectorize the text information, TF-IDF and Count vectorizer. Term Frequency (TF) is the count of a word appearing in a document divided by the total number of words in the document, and Inverse Data Frequency (IDF) is the log of the number of documents divided by the number of documents that include the word.[5] Count Vectorizer count the number of times a word appears in the document which results in biasing in favour of most frequent words.

A hybrid recommender specifically uses both collaborative and content based filtering for making recommendations. It is proved to be more effective in many complicated cases, eg. E-Commerce.[14][13] There are multiple hybrid recommender methods, in this study , we mainly talk about LightFM model. LightFM is a hybrid matrix factorisation model representing users and items as linear combinations of their content features' latent factors. LightFM model learns embeddings for users and items through a way that encodes user preferences on items. When multiplied together, these representations produce scores for every item for a given user; items scored highly are more likely to be recommended to the users.[11]

3 Exploratory Data Analysis

The 10,000 Books Datasets include five csv files with names as ‘ratings’, ‘to_read’, ‘books’, ’tags’ and ‘book_tags’. The ‘ratings’ dataset includes ‘user_id’, ‘book_id’ and ‘rating’, which is the main data source for the recommendation by collaborative filtering. There are totally 53424 unique users and 10000 unique books in this dataset and the rating of ’4’ and ’5’ (in the scale of 0 to 5) take 35.8% and 33.2% of the entire data. The ‘book’ dataset contains the metadata for books and the distribution of ‘average_rating’ also shows that rating of ’4’ and ’5’ take most proportion of the entire dataset.

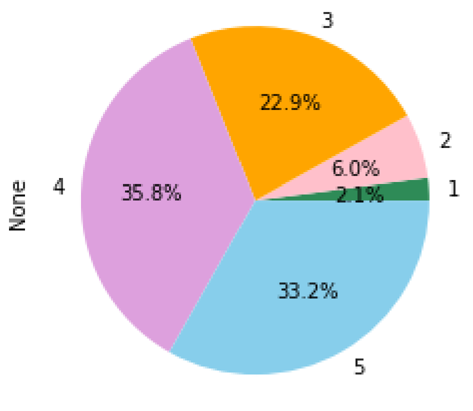


Figure 1: Rating Count

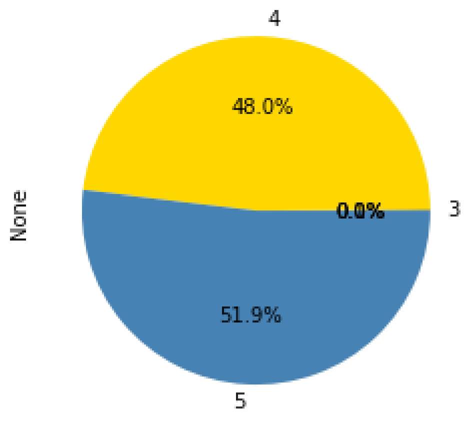


Figure 2: Book Average Rating Count

The ‘tag’ dataset includes the tags (a kind of text messages) of the books given by users. The tag that was used the most often is ‘to_read’. According to the dataset ‘to_read’, 91% of the users have some books marked to read and 99.9% books have been marked ‘to_read’ by some users.

By the count of tags in ‘book_tag’ dataset, we got the occurrence of each tag, the WordCloud of the top 500 and bottom 500 most often used tags indicate that the tags that were used the most often by users include the messages like "fantasy", "fiction","literature" and "romance" etc., which tends to be typical and formal literature and adult books. The tags that were used occasionally by users include the messages like "comic’, "manga", "cookbook" and "movie" etc, which is more likely to be casual and daily readings.

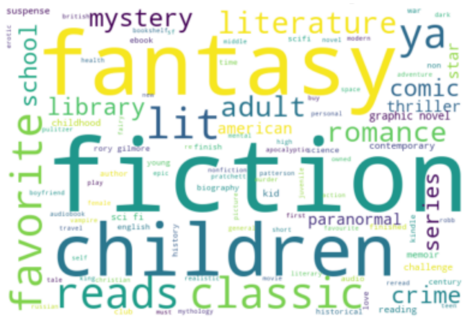


Figure 3: Most Often Used Tags

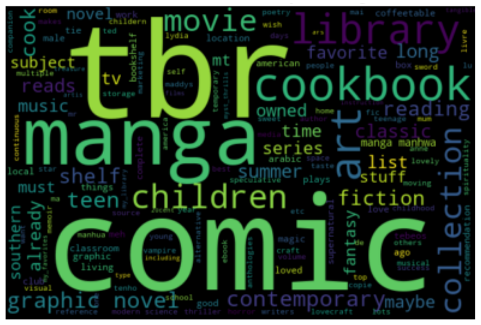


Figure 4: Occasionally Used Tags

4 Collaborative Filtering

4.1 Methods

In this study, user-based, item-based and model-based collaborative filtering have been tried and evaluated. User-based or user-user collaborative filtering means that a rating matrix is used to find similar users based on the ratings they give. The users with similarity of observed preferences with the active user are set as the neighbours. Then, items that were preferred by users in the neighborhood will be recommended to the active user. Item-based or item-item collaborative filtering means using the rating matrix to find similar items based on the ratings given to them by users. This method recommends items also prefered by users that prefer a particular active item to other users that also prefer that active item. Model-based collaborative filtering involves a step to reduce or compress the large but sparse user-item matrix. If the matrix is mostly empty, dimension reduction can improve the performance of the algorithm for both space and time.

We used the python library *Surprise* to implement these algorithms. For user-based and model-based collaborative filtering, the algorithm based on K-Nearest Neighbours is the most regular model. KNN uses a dataset in which the data points are separated into several clusters to make inference for new samples.This algorithm is very close to the centered cosine similarity formula, which is available in the library as KNNWithMeans and other KNN algorithms. For model-based collaborative filtering, the SVD and SVDpp are the very powerful algorithms to conduct the dimension reduction, which are the commonly used models for many studies.

4.2 Evaluation

Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are the two evaluation methods that are often used on collaborative filtering. RMSE is the standard deviation of the residuals. Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are.

$$RMSE = \sqrt{\frac{1}{\hat{R}} \sum_{\hat{r}_{ui} \epsilon \hat{R}} (r_{ui} - \hat{r}_{ui})^2}$$

MAE is a measure of errors between paired observations expressing the same phenomenon.

$$MAE = \frac{1}{\left|\hat{R}\right|} \sum_{\hat{r}_{ui} \epsilon \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

4.3 Results

The ‘rating’ dataframe has 5976479 rows with different user and book combinations and the rating scores to the books given by users. We randomly selected 25000 unique users and 5000 unique books from this data set as the train set, then use the rest dataset as the test set. Thus, the train and test set don’t have any overlapping user-book combination. Then the train and validation set has been obtained by train-test split and the validation set will be used to evaluate the model.

Before we built the recommendation engines, we did a basic trial to find the best algorithm based on the MAE. We used a small set of sample data to feed different models in the *Surprise* library, and found that KNNWithMeans should be the best KNN algorithm and SVDpp might be the best model-based algorithm. We plan to mainly use these algorithms to do the recommendation.

First of all , we used the BaselineOnly model to fit the train set in order to set a baseline to our model building. This model gives the mean MAE 0.6855 and the mean RMSE 0.8666 for the train set and the MAE 0.6859 and RMSE 0.8676 for the validation set.

We can select user-based or item-based method by KNNWithmeans model. By tuning the KNNWithmeans model, we found that the “msd similarity” is better than “cosine similarity” in this case and “min_support”should be 3. So we used these parameters to train the model and then do the evaluation. For the train set, the item-based KNNWithMeans model gives the mean MAE 0.6685 and mean RMSE 0.8628 for the train set. Then we did the evaluation on validation set and got the MAE of 0.6690 and the RMSE 0.8641. So, the KNN model is better than the baseline model. In addition, we also tried the user-based KNN model by setting the parameter to "user_based": True. Not surprisingly, this model also gives good results with mean MAE of 0.6623 and mean RMSE of 0.8579 for train set, and MAE of 0.6633 and RMSE of 0.8592 for validation set.

Then we tried the SVDpp algorithm to do the dimension reduction. We first tuned the hyper parameter. The tuning result suggests 20 for n_factors, 20 for n_epochs, 0.007 for lr_all, which are the same as the default values, and 0.1 for reg_all. The model fitting result gives the mean MAE 0.6738 and the mean RMSE 0.8603 for the train set and the MAE of 0.6743 and the RMSE of 0.8617 for the validation set, this result is also better than the Baseline model.

Based on the evaluation result and considering the sparse feature of the dataset, the SVDpp model has been used to do the final recommendation for the users in the test set. The books with the predicted rating larger than or equal to 4.0 will be recommended to this user. Because the test set includes the original rating values, which can be used as the ground truth data, the predicted values and the ground truth values are compared to evaluate the accuracy of the prediction. The MAE is 0.7517 and the RMSE is 0.9533. We also did the evaluation on the test set by KNN models and found that SVDpp gives the best prediction. The example recommendation demo result is shown below:

```
user_id
      2    book_id    260
           book_id   2318
           book_id    315
           book_id     33
           book_id    301
         ...
  35336    book_id    317
  21879    book_id   5674
           book_id   6720
  49925    book_id    330
           book_id    528
Length: 1210242, dtype: int64
```

Figure 5: Recommendation Results

5 Content-Based Filtering

5.1 Methods

For the purpose of getting all the necessary information, we merged the 'book_tag', 'tag', and 'book' datasets by their corresponding shared id columns. After we extracted the title, author and all the tags for each book, we cleaned them by basic text data preprocessing steps including turning all the text into lowercase and adjusting the whitespaces. Then we combined the author and tags together and generated the desired corpus of the books.

We tried two methods to transform the text in the corpus to a matrix of features after removing the stop words for further processing. One of the methods is the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer that balances out the term frequency (how often the word appears in the document) with its inverse document frequency (how often the term appears across all documents in the data set). The other is the Count vectorizer that gives a vector with the number of times each word appears in the document. The formula for TF-IDF is as followed:[17]

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Then we used the vectors from the previous step to calculate the matrix of cosine similarity scores between any two books in the dataset and created corresponding indices for the matrix. In this way, we could find how similar two books are and provide recommendations based on the similarities.

We found a problem that if user input is not exactly a book's title, an error will occur. So in order to address this limitation of the current model, we applied BM25+ which provides a ranking of the search model to evaluate how relevant the documents are to the given user input. BM25+, as an extension of

BM25, deals with a deficiency of the original BM25. To avoid the improper lower-bounded problem that arises from normalizing term frequency by the length of the document, unlike BM25, the scoring formula of BM25+ contains only one extra free parameter δ (default is 1).[15]

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot [\frac{f(q_{i,D} \cdot (k_1+1))}{f(q_{i,D}) \cdot k_1 \cdot (1-b+b \cdot \frac{|D|}{avgdl})} + \delta]$$

In this way, with any input words by the user, the search engine will first find the corresponding most relevant book title, and then the recommendation system will return 10 most similar books based on the matched book title.

A weighted scoring system was also introduced to improved the recommendation system. This system uses Bayes average estimation as the baseline rating:

$$Weighted\ Rating = (\frac{v}{v+m} \cdot R) + (\frac{m}{v+m} \cdot C)$$

where: R (Rating) = average rating for the book, v (votes) = number of votes for the book, m = minimum votes required to be listed in the Top 2.5%, C = the mean rating across the whole corpus.

5.2 Results

Here we will present a simple demo which could recommend numbers of similar books and print out the cover images of the books, given a fuzzy book title. When input is "The Hunger Games", The TFDIF model will give the following result:

Figure 6: Recommendation Results

Recommended Books
The Hunger Games: Official Illustrated Movie Companion
The Hunger Games Trilogy Boxset (The Hunger Games, #1-3)
The Hunger Games Tribute Guide
Catching Fire (The Hunger Games, #2)
Divergent (Divergent, #1)
Monsters of Men (Chaos Walking, #3)

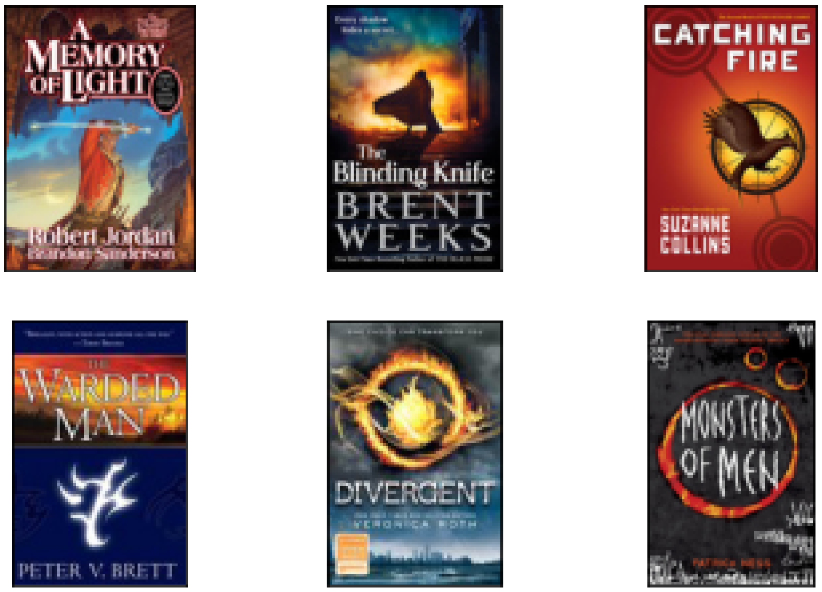


Figure 7: Recommendation Results

With the same input, the Count model would give a result like this:

Recommended Books
A Memory of Light (Wheel of Time, #14)
The Blinding Knife (Lightbringer, #2)
Catching Fire (The Hunger Games, #2)
The Warded Man (Demon Cycle, #1)
Divergent (Divergent, #1)
Monsters of Men (Chaos Walking, #3)

We don't do strict evaluation on this content based model in this project, because there would be unnecessary overlap with the following section. Based on an artificial judging from our team, the recommendation system performs well and the TF-IDF model has a much better performance than the Count model.

6 Hybrid Recommender System

6.1 Methods

The above two methods we have implemented have their advantages and disadvantages. As a team of 3, we want to make use of both thoughts and build a better and more robust performing model. Also, having already tried taking use of explicit feedback, we want to try using implicit feedback. So next we will introduce a hybrid recommendation system which combines the two approaches. This method is implemented using the open-source library *LightFM*.[12]

6.2 Data Preparation

We mainly need two datasets for this method: one with user rating information, the other one with books tag information. To build a hybrid recommender system, we need an interaction matrix between readers and books, or say users and items. We also need a metadata of items to construct an item features dataset. Here we use authors and books tags as items features, which is the same as the second method this project implemented.

To accelerate the training process and also highlight the effect of comparison in following sections, we filtered the datasets to keep only those readers who have rated more than 50 books and those books which have more than 100 ratings. As for constructing item features, we keep only those book tags data which have more than 210 counts, i.e. Only when a certain book has more than 210 clicks on a certain tag, we think this book tag pair is reliable. After filtering, we still have too much data. We keep all remaining books but sample 2000 readers among all 53424 readers. Finally, we have rating data and tag data for 2000 readers and 9374 books. There are 224190 pieces of ratings and 43588 pieces of unique book features including book id, authors and tags.

6.3 Explicit feedback versus Implicit feedback

In rating dataset, previously we are directly using the values of the ratings and the goal is actually predicting the values of ratings and calculating mean absolute error as evaluation. But there is also another thought: what people choose to rate or not rate expresses a more fundamental preference than what the ratings is.[10] This sort of phenomenon is described as data which is missing-not-at-random in the literature: the ratings that are missing are more likely to be negative precisely because the user chooses which items to rate.[21] So that implicit feedback is far more valuable than explicit feedback in many cases. That's why our group want to try using implicit feedback only in this method.

We tried different ways to construct implicit feedback based on the rating dataset and finally decide to use the following transformation. To remove readers bias from subjective rating, we scaled the rating data first by subtracting the average rate each user gives for each user item rating pair. Then we kept those rates larger than 0.5 and converted to 1, and converted others to 0, which is the same with all unrated ones. For example, reader A has 10 rates on 10 different books, and the average rate is 3, then we converted the rates that are larger than 3.5 to 1, and others to 0. We assume that for this reader, the books rated larger than 3.5 would be positive or liked items, and others would be negative or disliked items.

Using implicit feedback here means that we actually classify the books to "liked" and "disliked" for each reader. And the goal is to rank the liked books and disliked books in a correct order, rather than predicting the values of the rates.

6.4 Model Settings and Evaluation

LightFM model adopt matrix factorization, which decomposes matrix into two user feature matrix and item feature matrix, which are called user and item embedding. The latent embedding produced by pure rating interaction matrix capture latent features of users and items. The *LightFM* library is flexible in combining items features from metadata. Actually, if we include item feature dataset constructed before in the input, then the item embedding would be the sum of itself plus its metadata embedding. So we can produce a hybrid model.

We are also provided several loss functions and we will introduce BPR and WARP here, which are suitable for implicit feedback. BPR denotes Bayesian Personalised Ranking pairwise loss.[19] It maximises the prediction difference between a positive example and a randomly chosen negative example by calculating the difference between two predicted scores and then passing this difference through a sigmoid function and uses it as a weighting to update all of the model parameters via stochastic gradient descent. While WARP is Weighted Approximate-Rank Pairwise loss[22] and it maximises the rank of positive examples by repeatedly sampling negative examples until rank violating one is found. Generally, WARP performs better than BPR but it also takes more time to train. Because if the rank is not violated, it will keep on sampling negative samples until a violation happens, which is harder and harder to achieve during the training process.

The two loss functions are similar in the way that they ignored how well the prediction of the rating for each user item pair is, and focus only on the relative ranking between items for each user. In our project, we tried both loss functions and the final model adopts WARP.

LightFM also provides two learning rate schedules: adagrad[7] and adadelt[23]. We evaluate the performance of both of them and will present the results in following subsection.

We split the dataset as train and test chronologically. The test set contains 20% of interactions and the train contains 80%. They are disjoint sets and no effort is made to make sure that all items and users with interactions in the test set also have interactions in the training set.

We train the model on training set and evaluate the model on test set using AUC as our evaluation.[3] As what we care is ranking the items in correct order, so here we use AUC score as the model evaluation metric. To briefly introduce, AUC score is the area under ROC curve and can measure the probability that a randomly chosen positive example has a higher score than a randomly chosen negative example. A perfect auc score is 1.0, and an auc score of 0.5 could be a random baseline.

6.5 Comparison and Results

We establish both pure collaborative filtering based on interaction matrix and hybrid model based on interaction matrix and item features. We tried both BPR and WARP loss function and WARP takes more time but return a better performance, so we will adopt WARP as our loss function.

We also tried both adagrad and adadelta learning rate schedules. We run each model for a number of epoches, measuring the AUC score on the test set at the end of each epoch. The training process is as followed.

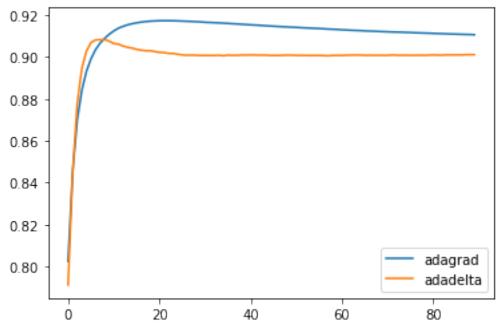


Figure 8: Pure CF adagrad/adadelta

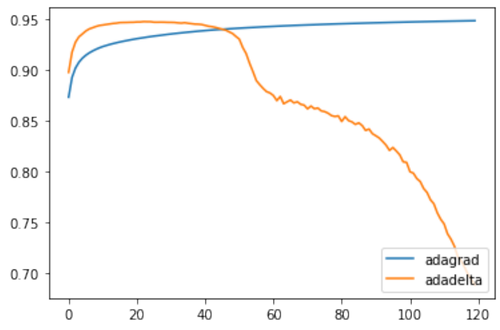


Figure 9: Hybrid model adagrad/adadelta

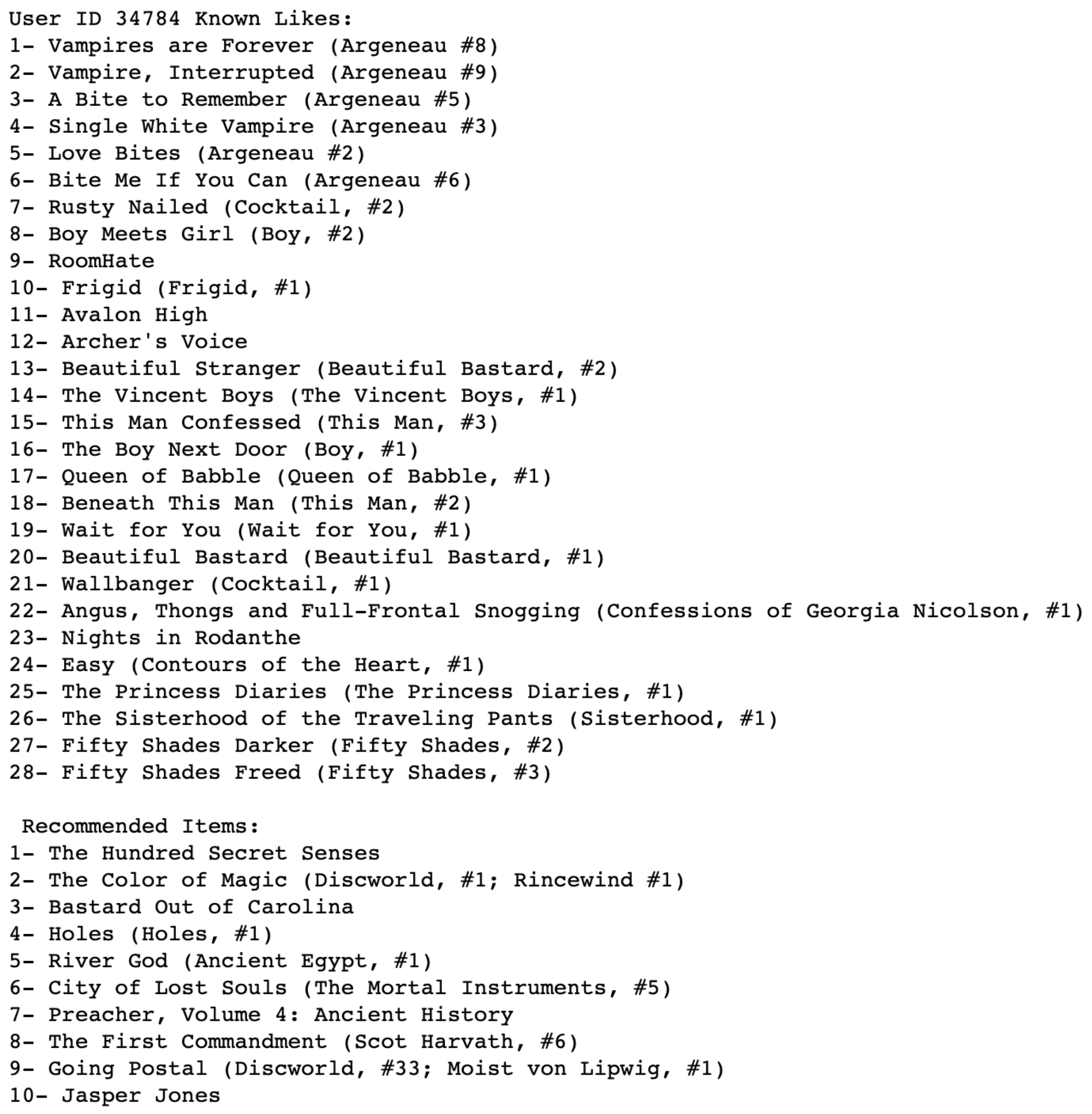
It looks like for both pure collaborative filtering and hybrid model, the adadelta gets better result in the beginning of training. But adagrad converges to a better final result. And comparing pure collaborative filtering and hybrid model, would say hybrid model performs better when generalized to the test set. Pure CF model is quick in training and fit the training set very well, but seems to have more overfitting problems.

To build the final pure CF model, we use adagrad to get a higher auc score. But to build the final hybrid model, we still use adadelta because adagrad would take too much time to get a good performance. Even so, hybrid model performs better than pure CF. The final model performance comparison is as follows.

Model Performance		
AUC score	Pure CF	hybrid model
Train set	0.98	0.97
Test set	0.92	0.95

6.6 Recommendation Demo

Here we use a re-usable code for reference[1] and build the Recommendation demo for our dataset. It will print out known liked books for the reader and recommend another top 10 books which are not "liked" before by this specific reader.



```
User ID 34784 Known Likes:
1- Vampires are Forever (Argeneau #8)
2- Vampire, Interrupted (Argeneau #9)
3- A Bite to Remember (Argeneau #5)
4- Single White Vampire (Argeneau #3)
5- Love Bites (Argeneau #2)
6- Bite Me If You Can (Argeneau #6)
7- Rusty Nailed (Cocktail, #2)
8- Boy Meets Girl (Boy, #2)
9- RoomHate
10- Frigid (Frigid, #1)
11- Avalon High
12- Archer's Voice
13- Beautiful Stranger (Beautiful Bastard, #2)
14- The Vincent Boys (The Vincent Boys, #1)
15- This Man Confessed (This Man, #3)
16- The Boy Next Door (Boy, #1)
17- Queen of Babble (Queen of Babble, #1)
18- Beneath This Man (This Man, #2)
19- Wait for You (Wait for You, #1)
20- Beautiful Bastard (Beautiful Bastard, #1)
21- Wallbanger (Cocktail, #1)
22- Angus, Thongs and Full-Frontal Snogging (Confessions of Georgia Nicolson, #1)
23- Nights in Rodanthe
24- Easy (Contours of the Heart, #1)
25- The Princess Diaries (The Princess Diaries, #1)
26- The Sisterhood of the Traveling Pants (Sisterhood, #1)
27- Fifty Shades Darker (Fifty Shades, #2)
28- Fifty Shades Freed (Fifty Shades, #3)

 Recommended Items:
1- The Hundred Secret Senses
2- The Color of Magic (Discworld, #1; Rincewind #1)
3- Bastard Out of Carolina
4- Holes (Holes, #1)
5- River God (Ancient Egypt, #1)
6- City of Lost Souls (The Mortal Instruments, #5)
7- Preacher, Volume 4: Ancient History
8- The First Commandment (Scot Harvath, #6)
9- Going Postal (Discworld, #33; Moist von Lipwig, #1)
10- Jasper Jones
```

Figure 10: Final hybrid model demo

7 Conclusion

In this project, we built three kinds of recommendation systems (collaborative filtering, content-based filtering and hybrid recommender system) to implement book recommendation based on 10,000 Books Datasets.

We first explore the data by many exploratory analysis work. We found that there are totally 53424 unique users and 10000 unique books in this dataset. In the scale of 0 to 5, the rating of 4 and 5 take 35.8% and 33.2% of the entire data. The ratings dataset includes user_id, book_id and rating, which is the main data source for the recommendation by collaborative filtering.

We used the library *Surprise* to implement collaborative filtering. The algorithm KNNWithMeans in this package is a typical algorithm to predict ratings with user-based and item-based approaches, and our analysis shows that user-based approach provides somewhat more accurate prediction on our data. Because of the sparsity of the data, we also tried the model-based algorithm, SVDpp, to carry out the dimension reduction. Comprehensively considering all the factors, we finally selected SVDpp as the best algorithm to do the recommendation. We obtained the MAE of 0.6743 and the RMSE of 0.8617 for the validation set and the MAE is 0.7517 and the RMSE is 0.9533 for the test set.

We built content-based filtering recommendation engines by the titles, authors, tags, number of ratings and the average ratings of books. We calculated the cosine similarity scores between books after vectorizing the text information using TF-IDF and Count methods. Finally we added the BM25+ model to address the limitation of the search engine for this recommendation system. Based on artificial judging from our team, the recommendation system performs well and the TF-IDF model has a much better performance than the Count model.

Finally, we turn to implicit feedback based system and transformed the rating to "liked" and "disliked". We built both pure collaborative filtering model and a hybrid model based on library *LightFM*. The experiment shows that the hybrid model is better than the pure CF model. Our final model achieve an AUC score of 0.95, which is quite high and indicated that this model did a great job in ranking items task. By the way, we cannot simply compare the pure CF model in this setting to the one we previously built. They have different goals and different evaluation metric. If there is a need to compare this two settings: xplicit feedback versus Implicit feedback, ranking performance can be used as a common metric. But this is out of the scope of this project. We'll not discuss this in this report, but will be very interested and happy to explore more in this field in the future.

In addition, we build 3 sample demos for the above three recommendation systems, it may be a good project in another field to make a web version and realize the functions of instant interaction.

References

[1] Aayush Agrawal. *Re-usable code in Python 3 for Recommender systems*. URL: `https://github.com/aayushmnit/cookbook/blob/master/recsys.py`.

[2] Hyung Jun Ahn. "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem". In: *Information Sciences* 178 (Jan. 2008), pp. 37–51. DOI: `10.1016/j.ins.2007.07.024`.

[3] Andrew P. Bradley. "The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms". In: *Pattern Recogn.* 30.7 (July 1997), 1145–1159. ISSN: 0031-3203. DOI: `10.1016/S0031-3203(96)00142-2`. URL: `https://doi.org/10.1016/S0031-3203(96)00142-2`.

[4] John S. Breese, David Heckerman, and Carl Kadie. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering". In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. UAI'98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., 1998, 43–52. ISBN: 155860555X.

[5] Jorge Castro, Rosa M. Rodriguez, and Manuel J. Barranco. "Weighting of Features in Content-Based Filtering with Entropy and Dependence Measures". In: *International Journal of Computational Intelligence Systems* 7 (1 2014), pp. 80–89. ISSN: 1875-6883. DOI: `https://doi.org/10.1080/18756891.2013.859861`. URL: `https://doi.org/10.1080/18756891.2013.859861`.

[6] G. Cheng and S. Gong. "Mining User Interest Change for Improving Collaborative Filtering". In: *Intelligent Information Technology Applications, 2007 Workshop on*. Vol. 3. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 24–27. DOI: 10.1109/IITA.2008.385. URL: https://doi.ieeecomputersociety.org/10.1109/IITA.2008.385.

[7] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: http://jmlr.org/papers/v12/duchi11a.html.

[8] S. Gong. "The Collaborative Filtering Recommendation Based on Similar-Priority and Fuzzy Clustering". In: *2008 Workshop on Power Electronics and Intelligent Transportation System*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 248–251. DOI: 10.1109/PEITS.2008.99. URL: https://doi.ieeecomputersociety.org/10.1109/PEITS.2008.99.

[9] Katsuhiro Honda et al. "Collaborative Filtering Using Principal Component Analysis and Fuzzy Clustering". In: *Web Intelligence: Research and Development*. Ed. by Ning Zhong et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 394–402. ISBN: 978-3-540-45490-8.

[10] Maciej Kula. *Explicit vs Implicit Recommenders*. 2018. URL: https://maciejkula.github.io/2018/07/19/dont-use-explicit-feedback-recommenders/. (accessed: 01.09.2016).

[11] Maciej Kula. *Metadata Embeddings for User and Item Cold-start Recommendations*. 2015. arXiv: 1507.08439 [cs.IR].

[12] Maciej Kula. "Metadata Embeddings for User and Item Cold-start Recommendations". In: *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015*. Ed. by Toine Bogers and Marijn Koolen. Vol. 1448. CEUR Workshop Proceedings. CEUR-WS.org, 2015, pp. 14–21. URL: http://ceur-ws.org/Vol-1448/paper4.pdf.

[13] Yu Li, Liu Lu, and Li Xuefeng. "A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in E-Commerce". In: *Expert Systems with Applications* 28.1 (2005), pp. 67 –77. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2004.08.013. URL: http://www.sciencedirect.com/science/article/pii/S0957417404000910.

[14] Duen-Ren Liu and Ya-Yueh Shih. "Hybrid approaches to product recommendation based on customer lifetime value and purchase preferences". In: *Journal of Systems and Software* 77.2 (2005), pp. 181 –191. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2004.08.031. URL: http://www.sciencedirect.com/science/article/pii/S0164121204001426.

[15] Yuanhua Lv and ChengXiang Zhai. "Lower-Bounding Term Frequency Normalization". In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. CIKM '11. Glasgow, Scotland, UK: Association for Computing Machinery, 2011, 7–16. ISBN: 9781450307178. DOI: 10.1145/2063576.2063584. URL: https://doi.org/10.1145/2063576.2063584.

[16] Manos Papagelis and Dimitris Plexousakis. "Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents". In: *Engineering Applications of Artificial Intelligence* 18.7 (2005), pp. 781 –789. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2005.06.010. URL: http://www.sciencedirect.com/science/article/pii/S0952197605000825.

[17] Anand Rajaraman and Jeffrey David Ullman. "Data Mining". In: *Mining of Massive Datasets*. Cambridge University Press, 2011, 1–17. DOI: 10.1017/CBO9781139058452.002.

[18] YiBo Ren and SongJie Gong. "A Collaborative Filtering Recommendation Algorithm Based on SVD Smoothing". In: *Proceedings of the 3rd International Conference on Intelligent Information Technology Application*. IITA'09. Nanchang, China: IEEE Press, 2009, 530–532. ISBN: 9781424452125.

[19] Steffen Rendle et al. "BPR: Bayesian Personalized Ranking from Implicit Feedback". In: *CoRR* abs/1205.2618 (2012). arXiv: 1205.2618. URL: http://arxiv.org/abs/1205.2618.

[20] Badrul Sarwar et al. "Item-Based Collaborative Filtering Recommendation Algorithms". In: *Proceedings of the 10th International Conference on World Wide Web*. WWW '01. Hong Kong, Hong Kong: Association for Computing Machinery, 2001, 285–295. ISBN: 1581133480. DOI: 10.1145/371920.372071. URL: https://doi.org/10.1145/371920.372071.

[21] H. Steck. “Training and testing of recommender systems on data missing not at random”. In: *KDD ’10*. 2010.

[22] Jason Weston, Samy Bengio, and Nicolas Usunier. “WSABIE: Scaling up to Large Vocabulary Image Annotation”. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*. IJCAI’11. Barcelona, Catalonia, Spain: AAAI Press, 2011, 2764–2770. ISBN: 9781577355151.

[23] Matthew D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. arXiv: `1212.5701 [cs.LG]`.