Book Recommender System and Analysis Using Matrix Completion Algorithms

Ananya Jegannathan

1 Abstract

Recommender systems are used to make qualitatively useful product suggestions to users based on their previous usage information. There are multiple methods to do this, and one common method uses user-product interactions, such as ratings, to select recommendations. However, as users will only rate a small ratio of the products, the matrix containing the interactions will be sparse and will need to be filled, resulting in the need for an efficient and accurate matrix completion algorithm. This project aims to build a book recommender system while simultaneously comparing two different matrix completion algorithms to determine which performs better.

2 Introduction

Recommendation systems have existed for a while now, and one of their primary uses is in the sphere of entertainment. Systems that provide suggestions for what to watch on Netflix or what to listen to on Spotify are commonplace. While they are not perfect, the quality of recommendations that a good system can provide is astonishing. This project aims to create a recommendation system for books in Python.

3 Literature Review

The primary facets of this project are building a recommender system, and evaluating the performance of different matrix completion algorithms. Building a recommender system is a popularly explored topic, but there is a standard approach to it. This primarily involves matrix completion or factorization, as explored in several blog posts and online sources[8][11][9]. Of the various matrix completion algorithms that exist, the two that are used in this project are Non-Negative Matrix Factorization[2] and Iterative Singular Value Thresholding[7][6], in particular the Soft Impute algorithm, first introducted by Mazumdar et al.[12]

4 Types of Recommender Systems

There are two main types of recommender systems, which differ based on the information they use to make recommendations [8][11][9].

4.1 Content-Based Filtering

These types of systems recommend items to users based on the attributes and properties of the items themselves. The system observes the items that a user has liked in the past, and recommends to that user items which have similar attributes. In order for this system to work, it needs information about the items. For example, a content-based book recommender system would require meta data about the books such as genre, length, target age group etc. The advantage of such a system is that adding new items is straightforward - all that's required are the item's attributes. The main drawback, however, is that users will only be recommended the same type of items - that is, there will be no diversity or scope to branch out and try new things.

4.2 Collaborative Filtering

These types of systems recommend items to users based on items liked by users with similar tastes. The system observes the interactions between users and items, such as ratings assigned to the items by the users. Based on these interactions the systems finds similarities between users, and recommends items to users based on items that have already been liked by similar users. This ensures that users are recommended a wide range of items, with a high chance of these items suiting their tastes. These recommender systems do not require any information about the items as such. The main drawback of these systems is that it is difficult to add new items, as no users will have interacted with them and so the system will not work on them.

This project builds a book recommender system using collaborative filtering methods. It uses the ratings assigned to books by users to make recommendations.

4.2.1 History of Collaborative Filtering

Collaborative filtering methods are the most commonly used methods in popular recommendation systems. For example, Netflix and Spotify both use collaborative filtering methods to recommend new movies/shows and songs respectively to their users. These methods became well-known due to the Netflix prize challenge, which was an open competition held by Netflix to find a collaborative filtering algorithm which has a lower root mean squared error than its existing system[1][13].

One of the first matrix completion algorithms, known as Funk MF, was proposed by Simon Funk in 2006[3]. This is a basic matrix factorization algorithm. Another very popular algorithm is SVD++, which adds user and item bias terms to the factorization used in Funk MF.

5 Matrix Completion

The collaborative filtering methods discussed above, all require the use of user-item interactions, or ratings, which take the form of a matrix. Each row consists of the ratings given by a particular user to the items, and each column consists of all the ratings given to a particular item. In general, users will never rate all the existing items in a system - often, the proportion of rated to unrated items per user is quite low. This results in a high level of sparsity in the user-item ratings matrix. However, the system can only make recommendations if there is some rating for every

item, that is, if the ratings matrix is filled. Hence, there is a need for matrix completion algorithms.

5.1 Matrix Factorization

One of the most common methods of matrix completion is factorization. This involves decomposing the user-item ratings matrix.

The ratings in the user-item ratings matrix are explicitly known values. However, that is not the only information implied by the matrix. There are some features which are not explicitly defined, but are important in predicting ratings. These are known as latent features. Matrix factorization involves the decomposition of the ratings matrix into the product of two rectangular matrices with lower dimension. These two matrices are latent-factor matrices. The inner product of these two matrices gives the filled ratings matrix.

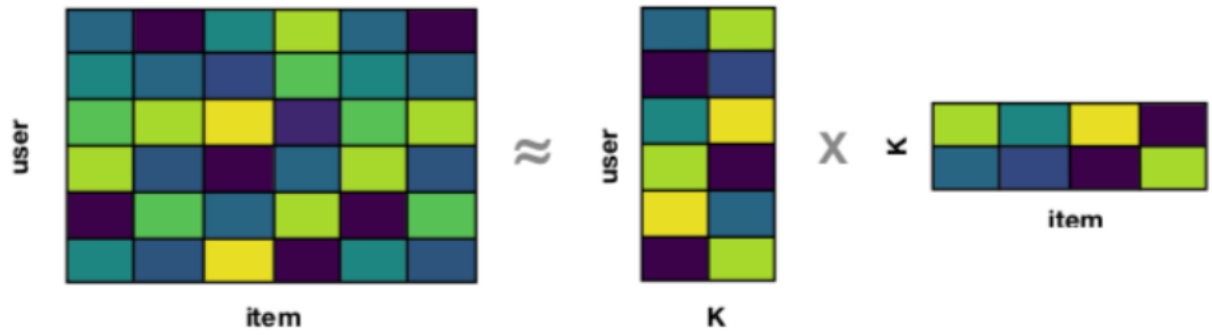

Figure 1: Matrix Decomposition[11]

Most popular training algorithms use stochastic gradient descent to iteratively update and optimize the latent-factor matrices. There are several such algorithms, such as Alternating Least Squares(ALS) and Non-Negative Matrix Factorization (NMF). This project implements the NMF algorithm.

5.1.1 Non-Negative Matrix Factorization (NMF)

NMF is a form of matrix factorization which forces all the values in the latent-factor matrices to be non-negative[2]. The problem formulation involves decomposing a non-negative matrix V into two non-negative matrices W and H, such that

$$V = WH \tag{1}$$

5.2 Iterative Singular Value Thresholding (ISVT) - Soft Impute Algorithm

Another method to perform matrix completion is using the singular value decomposition (SVD) of the user-item ratings matrix. An efficient way to do this is using iterative singular value thresholding.

Matrix completion is achieved by minimizing the rank of the ratings matrix. By minimizing the rank, the dependence of the rows and columns of the matrix increases, which makes it possible to fill in the missing entries. ISVT performs this minimization iteratively.

Consider a ratings matrix X, and its low-rank approximation Z. The optimization problem involved in ISVT is,

$$\begin{aligned} &\min rank(Z) \\ &\text{subject to} \sum_{i,j \in \Omega} (X_{i,j} - Z_{i,j})^2 \leq \delta \end{aligned}$$

where δ is a positive regularization parameter which controls the error tolerance.

However, rank is non-convex function, and so is complicated to directly minimize. Instead, rank minimization can be achieved by minimizing the nuclear norm of the matrix as nuclear norm is the convex envelope of the rank function. Nuclear norm is given by

$$||X||_* = trace(\sqrt{X^*X}) = \sum_{i=1}^{min(m,n)} \sigma_i(X) \tag{2}$$

That is, the nuclear norm of a matrix is the sum of its singular values, obtained by finding its SVD. ISVT iteratively calculates the SVD and so is computationally tractable. Soft Impute is an ISVT algorithm which exploits the problem structure such that it can handle matrices with very large dimensions[12][7][6].

6 Building the Recommender System

The steps involved in building the system are as follows.

1. Data collection
2. Data preprocessing
3. Matrix completion
4. Selecting recommendations

6.1 Data Collection

The datasets for this project were obtained from UCSD Book Graph[5][10][**ucsd2**]. These datasets were collected from Goodreads[4], the world's largest online collection of user-book interactions and book information. The dataset consists of information from around 900,000 users and 2.3 million books. There are several files containing user-book interaction information, book meta-data, fuzzy genre information etc.

6.2 Data Preprocessing

There were several steps involved in converting the raw dataset to a format usable for this project.

1. Running code on all the data was not possible due to its huge size. Due to restrictions in computational capacity, a subset of this dataset was used. The first 1000 users were selected, and the 100 books most frequently rated by these users were found.

2. The ratings given by these 1000 users for these 100 books were converted from the original format shown in Fig.2 into a user-book matrix of size 1000x100 shown in Fig.3.

	A	B	C	D	E	F
1	user_id	book_id	is_read	rating	is_reviewed	
2	0	948	1	5	0	
3	0	947	1	5	1	
4	0	946	1	5	0	
5	0	945	1	5	0	
6	0	944	1	5	0	
7	0	943	1	5	0	
8	0	942	1	5	0	
9	0	941	1	5	0	
10	0	940	1	5	0	
11	0	939	1	5	1	
12	0	938	1	5	1	
13	0	937	1	4	0	
14	0	936	1	4	0	
15	0	935	1	4	0	

Figure 2: Raw Data

```
array([[5.        , 5.        , 5.        , ..., 4.16927564, 2.94860368,
        3.34628419],
       [4.83312839, 5.        , 5.04571274, ..., 3.63526305, 3.33470367,
        3.60114908],
       [3.82413524, 3.        , 5.        , ..., 3.05186377, 2.65140927,
        2.86005606],
       ...,
       [3.93823214, 4.        , 4.        , ..., 5.        , 2.3054987 ,
        3.06081019],
       [2.00233861, 2.31540618, 1.81415673, ..., 2.07685405, 1.05195707,
        1.33555611],
       [3.        , 5.        , 3.23861284, ..., 2.54013786, 3.74926268,
        4.        ]])
```

Figure 3: Formatted Data

3. Since only a subset of the data was used, there were several rows and columns which ended up being empty. These were deleted.

6.3 Matrix Completion

To build and execute the recommender system, initially the Soft Impute algorithm was used for matrix completion. This was implemented with the help of the *fancyimpute* Python library[14]. Once the system was functional, NMF was run on the sparse ratings matrix to compare the performance of the two algorithm. This is explained in further detail in the Results section of this report.

6.4 Selecting Recommendations

The final step in building a recommender system is to find the best items to recommend. There are two scenarios to consider.

6.4.1 Existing User

An existing user is a user who is already present in the system, meaning they have rated several of the books and these ratings are present in the user-book ratings matrix. In this case, recommendation were chosen after performing matrix completion by sorting the user's row of ratings in the filled ratings matrix, and selecting the highest rated books. It is important to select books which

the user has not read (rated) already, so the highest rated books not present in the sparse ratings matrix were selected.

6.4.2 New User

A new user is a user who has rated several of the books, but this row of ratings is not present in the user-book ratings matrix. This project attempted to select recommendations for new users in two ways:

1. **Norm Calculation:** The distance between the ratings of the new user and the original ratings of the existing users (before matrix completion) was calculated. That is, the distance between the ratings vector of the new user and each row of the sparse user-book ratings matrix. The row in the sparse ratings matrix (and hence, the existing user) with ratings closest to the new user was found and their top recommendations were given to the new user.

2. **K-Means Clustering:** An arbitrary k was selected to cluster the existing users, and the cluster for the new user was found. After this, all the books recommended to the existing users in that cluster were calculated, and the most common recommendations for that cluster were found and given to the new user.

7 Results

7.1 Building the System

As mentioned, the basic system was built using the Soft-Impute algorithm for matrix completion, and norm calculation for the task of selecting books to recommend. It was found that the recommended books were indeed similar to the books rated the highest by the user.

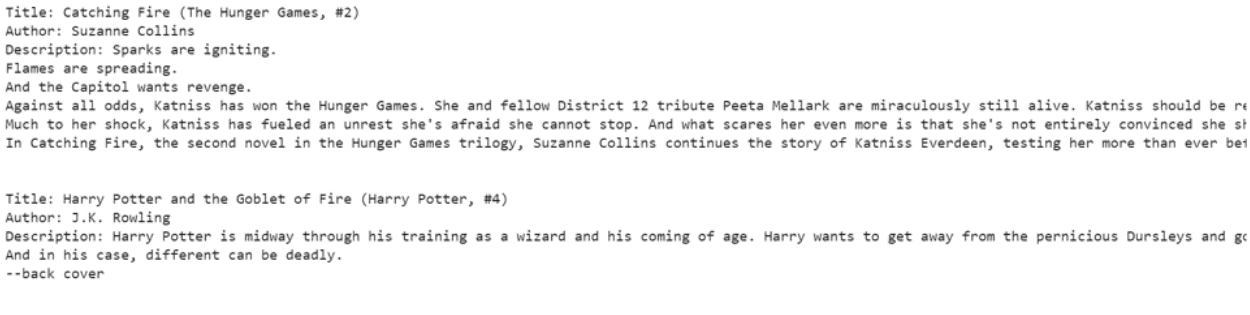

Figure 4: Rated Books

Fig.4 shows a few books that a particular user has given a rating of 5, which is the highest possible rating. According to Goodreads, they both share several common genre tags, including Young Adult, Fantasy, Fiction and Adventure. The recommended books shown in Fig.5 all share very similar themes with the two rated books, including several of the same genre tags. While the last recommended book, Night by Elie Wiesel, strays from the common genre tags of Young Adult and Fantasy, it does share similar dark themes to some of the rated books.

```
Title: Divergent (Divergent, #1)
Author: Veronica Roth
Description: Paperback features over fifty pages of bonus materials, including a sneak peek of Insurgent, an author Q&A, a discussion guide, a
In Beatrice Prior's dystopian Chicago world, society is divided into five factions, each dedicated to the cultivation of a particular virtue--C
During the highly competitive initiation that follows, Beatrice renames herself Tris and struggles alongside her fellow initiates to live out t
Veronica Roth is the New York Timesbestselling author of Divergent, the first in a trilogy of dystopian thrillers filled with electrifying deci


Title: Insurgent (Divergent, #2)
Author: Veronica Roth
Description: One choice can transform you--or it can destroy you. But every choice has consequences, and as unrest surges in the factions all a
Tris's initiation day should have been marked by celebration and victory with her chosen faction; instead, the day ended with unspeakable horro
New York Timesbestselling author Veronica Roth's much-anticipated second book of the dystopian DIVERGENT series is another intoxicating thrill


Title: Cinder (The Lunar Chronicles, #1)
Author: Marissa Meyer
Description: Sixteen-year-old Cinder is considered a technological mistake by most of society and a burden by her stepmother. Being cyborg does
Although eager to impress the prince, Cinder's intentions are derailed when her younger stepsister, and only human friend, is infected with the
But it doesn't take long for the scientists to discover something unusual about their new guinea pig. Something others would kill for.


Title: Night (The Night Trilogy #1)
Author: Elie Wiesel
Description: An autobiographical narrative in which the author describes his experiences in Nazi concentration camps, watching family and frien
Night is Elie Wiesel's masterpiece, a candid, horrific, and deeply poignant autobiographical account of his survival as a teenager in the Nazi
```

Figure 5: Recommended Books

7.2 Algorithm Comparison

The efficiency the two matrix completion algorithms used - Soft Impute and NMF - was compared on the basis of two factors - mean squared error and time.

7.2.1 Completed Matrices

Both algorithms were run on the sparse ratings matrix, and the completed matrices were compared.

```
array([[5.       , 5.       , 5.       , ..., 4.16927564, 2.94860368,
        3.34628419],
       [4.83312839, 5.       , 5.04571274, ..., 3.63526305, 3.33470367,
        3.60114908],
       [3.82413524, 3.       , 5.       , ..., 3.05186377, 2.65140927,
        2.86005606],
       ...,
       [3.93823214, 4.       , 4.       , ..., 5.       , 2.3054987 ,
        3.06081019],
       [2.00233861, 2.31540618, 1.81415673, ..., 2.07685405, 1.05195707,
        1.33555611],
       [3.       , 5.       , 3.23861284, ..., 2.54013786, 3.74926268,
        4.       ]])
```

Figure 6: Filled Matrix Using Soft Impute

```
array([[4.99990788e+00, 5.00000471e+00, 4.99999415e+00, ...,
        2.87126112e-08, 3.77270251e-02, 0.00000000e+00],
       [8.20196354e-04, 5.00000396e+00, 3.78750043e-07, ...,
        0.00000000e+00, 4.92826012e-05, 0.00000000e+00],
       [2.70712073e-04, 2.99999736e+00, 4.99999371e+00, ...,
        6.42956660e-10, 7.74611146e-03, 2.70493013e-04],
       ...,
       [6.22739877e-04, 4.00000087e+00, 3.99999615e+00, ...,
        5.00000027e+00, 1.49560742e-01, 0.00000000e+00],
       [6.56157084e-04, 4.05704598e-18, 2.42960710e-13, ...,
        2.60537335e-18, 0.00000000e+00, 2.52391789e-06],
       [2.99986837e+00, 4.99999542e+00, 3.75330046e-06, ...,
        7.26904842e-09, 1.93354750e-10, 3.99969980e+00]])
```

Figure 7: Filled Matrix Using NMF

In the matrix filled using Soft Impute, the known ratings are unchanged, whereas in the matrix filled using NMF, they have been approximated, showing that Soft Impute is relatively accurate. It can also be observed that Soft Impute approximates the unknown ratings in the same range as the known ratings, that is between 1 and 5. On the other hand, NMF approximates ratings below 1 and above 5 as well, which is not desirable.

7.2.2 Mean Squared Error (MSE) Comparison

In order to perform MSE calculations, first each algorithm was run on the sparse ratings matrix and the filled ratings matrices were saved. After this, a certain percentage of the ratings in the

sparse ratings matrix were held out. The algorithms were then run on this new sparse matrix. For each algorithm, the new filled matrix was compared with the original filled matrix to find how accurately the algorithm predicted the held-out ratings. This was calculated by finding the mean squared error between each original and new filled matrix. This procedure was repeated for multiple iterations and the average MSE was calculated, with a different set of randomly selected ratings held out each time.

The above procedure was repeated for different sizes of the hold-out set, that is, with different percentages of the ratings held out.

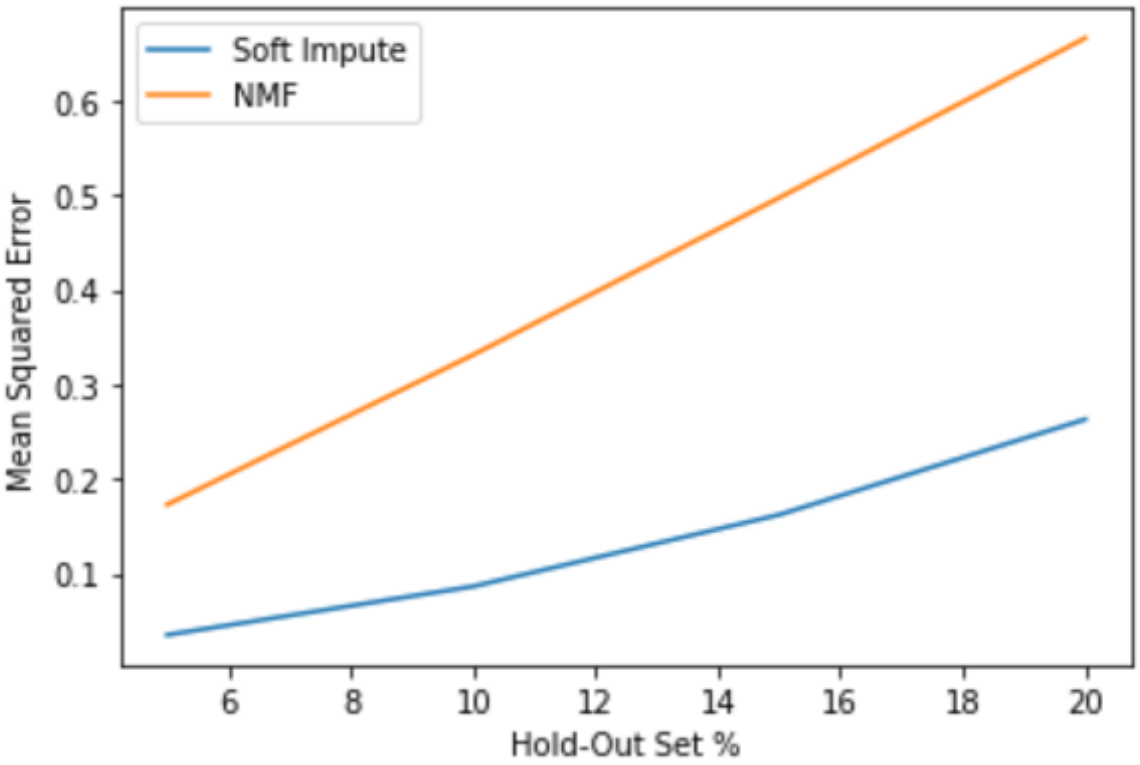

Figure 8: MSE vs Hold-Out Percentage for Soft-Impute and NMF

As seen in Fig.8, Soft Impute has a lower MSE in general, of around 0.1 on average, and it doesn't fluctuate by more than 0.1 to 0.15 as the percentage of ratings held out increases from 5% to 20%. However, NMF has a higher average MSE of around 0.5, and fluctuates around twice as much as Soft Impute. It can clearly be seen that Soft Impute has a more accurate performance.

7.2.3 Timing Comparison

The time taken to run one iteration of each algorithm was found, and was averaged over multiple iterations.

As shown in Table 1, NMF takes almost 2 seconds longer than Soft Impute to run. Clearly, Soft Impute performs better in regard to time.

Soft Impute	NMF
2.06s	3.93s

Table 1: Execution Time for Soft-Impute and NMF

7.3 Varying Dataset Size

This entire project was implemented using a ratings matrix of size 1000x100, meaning using 1000 users and 100 books. Varying the number of users and books would certainly have an impact on timing and MSE. The Soft Impute algorithm was run on datasets of different sizes to observe this difference. First, the number of users was kept constant at 1000 and the number of books was varied from 250 to 1750. Then, the number of books was kept constant at 1000 and the number of users was varied from 250 to 1750.

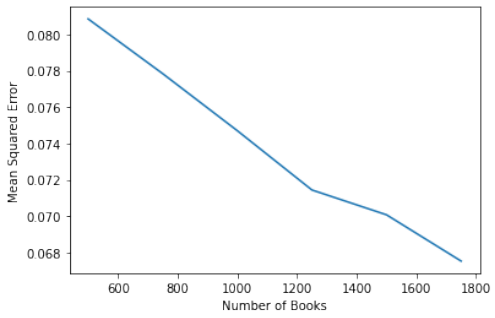

Figure 9: Number of Books vs MSE

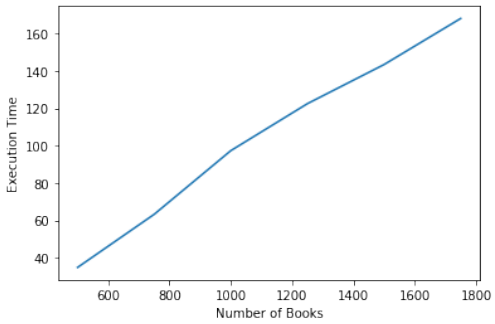

Figure 10: Number of Books vs Execution Time

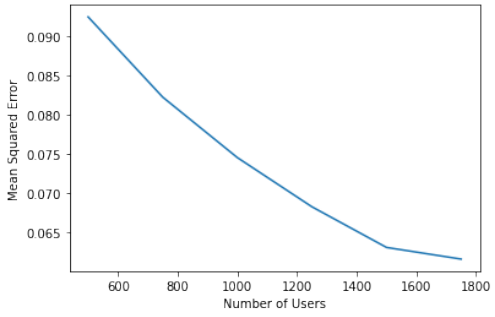

Figure 11: Number of Users vs MSE

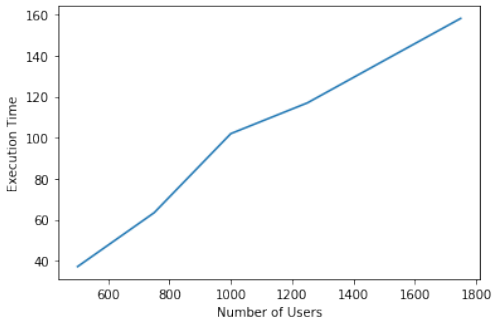

Figure 12: Number of Users vs Execution Time

It can be observed that the change in execution time is almost the same when the number of either books or users is changed. As the number of books or users is increased, the MSE reduces. It is interesting to note that the MSE is better when there are more users than books, than when there are more books than users. This makes sense, as each user would only have rated a certain number of books. So increasing only the number of users would increase the number of ratings a book has, whereas increasing only the number of books would potentially increase sparsity of the

ratings matrix.

7.4 Norm Calculation vs K-Means Clustering for Selecting Recommendations

To compare the performance of these two methods, an existing user's recommendation were first found, Then, this user's ratings row was deleted from the ratings matrix, and treating them as a new user, recommendations were chosen for this user using both methods. These were then compared to the original recommendations. It was found that there were more recommendations in common using norm calculation han using clustering.

8 Conclusion

A working recommender system was built and implemented using multiple methods and algorithms. On comparing the matrix completion algorithms, it was found that Soft Impute (iterative singular value thresholding) is more accurate and more efficient than Non-Negative Matrix Factorization. It was also found that norm calculation worked better than K-Means clustering for selecting recommendations.

9 Future Work

One of the primary challenges over the course of this project was the size of the dataset. It was too large to run with just a CPU. The immediate next step would be to scale this work up, and observe the results on the entire dataset.

In addition, there are several steps that can be taken to potentially improve the results on the smaller dataset. Hyperparameter optimization for K-Means clustering could be done to find parameters that yield better results. Also, the current mapping of the dataset does not take into account series' of books: it regards each book in a series as an independent book. Grouping all the books in a series together could improve recommendations, as it would ensure that a book which is part of a series but is not the first, is not recommended to a user who has not yet read the first book.

10 Acknowledgements

I'd like to express my gratitude to Dr. Matthew Malloy for guiding me through this project and teaching me a lot in the process.

References

[1] URL: `https://web.archive.org/web/20090924184639/http://www.netflixprize.com/community/viewtopic.php?id=1537`.

[2] H. Sebastian Seung Daniel D. Lee. "Algorithms for Non-negative Matrix Factorization". In: *Neural Information Processing Systems 2000* ().

[3] Simon Funk. URL: https://sifter.org/~simon/journal/20061211.html.

[4] *Goodreads*. URL: www.goodreads.com.

[5] *Goodreads Dataset*. URL: https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home.

[6] Zuowei Shen Jian-Feng Cai Emmanuel J. Candes. "A Singular Value Thresholding Algorithm for Matrix Completion". In: *Electronic Journal of Statistics* 9.2 (2015), pp. 2348–2369. DOI: 10.1214/15-EJS1076.

[7] Olga Klopp. "Matrix completion by singular value thresholding: Sharp bounds". In: (2008).

[8] Pavel Kordík. *Machine Learning for Recommender systems — Part 1 (algorithms, evaluation and cold start)*. URL: https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed.

[9] Madasamy M. *Introduction to recommendation systems and How to design Recommendation system,that resembling the Amazon*. URL: https://medium.com/@madasamy/introduction-to-recommendation-systems-and-how-to-design-recommendation-system-that-resembling-the-9ac167e30e95.

[10] Julian McAuley Mengting Wan. "Item Recommendation on Monotonic Behavior Chains". In: *RecSys'18* ().

[11] Parul Pandey. *Recommendation Systems in the Real world*. URL: https://towardsdatascience.com/recommendation-systems-in-the-real-world-51e3948772f3.

[12] Robert Tibshirani Rahul Mazumder Trevor Hastie. "Spectral Regularization Algorithms for Learning Large Incomplete Matrices". In: *Journal of Machine Learning Research 11* (2010), pp. 2287–2322.

[13] Chris Volinsky Robert M. Bell Yehuda Koren. *The BellKor 2008 Solution to the Netflix Prize*. URL: https://www.netflixprize.com/assets/ProgressPrize2008_BellKor.pdf.

[14] Alex Rubinsteyn. *fancyimpute*. URL: https://github.com/iskandr/fancyimpute.