




JULY 19, 2025

ROY PHELPS

Book Recommender Systems



## Table of Contents

Introduction.....	2
Problem 1 – Running the Recommender Engines .....	2
Problem 2 – Parameter Tuning .....	3
Impact of the Number of Latent Factors.....	3
Impact of Similarity Metric .....	4
Interaction Between Factors and Similarity Metric .....	4
Problem 3 – User – User Recommendations .....	4
Conclusion .....	5

## Introduction

Recommender systems play a key role in how users and people discover content, information, and products in the digital world. Everything from retail, streaming platforms, to social media including book sites like Goodreads. These systems help filter large volumes of data to deliver personalized suggestions tailored to individual preferences.

The core of recommender systems, they analyze patterns in the behavior of the users for things like ratings, purchases, or interactions to predict what a user might enjoy next. This assignment I implemented a collaborative filtering approach using matrix factorization techniques to identify the book-to-book similarities based on user ratings.

## Problem 1 – Running the Recommender Engines

This section was to use a basic book based collaborative filtering recommender system using user ratings from Goodreads. The provided datasets, *goodreads\_5k.csv*, *goodreads\_8k.csv*, and *goodreads\_9k.csv*, included ratings from 7 users across 7,076 books, 11 users across 7915 books, and 15 users across 8463 books respectively. The engine was designed to recommend books that are most like given title, based on how users have rated them. The system first builds a user item ranking matrix, then applies dimensionality reduction using Truncated Singular Value Decomposition (SVD) and lastly computes a similarity. Matrix using a specified distance metric. This matrix then was saved as a .pkl file or reuse during the testing stage and the matrix computed this in 0.14 seconds for the 5k file, 0.29 seconds, 0.35 seconds for the 9k file.

A test script was used to retrieve the five books most like a chosen input, in this case *The Ultimate Hitchhiker's Guide to the Galaxy*. The system returned a mis of related and unrelated recommendations. For example, *Stardust*, *Gathering Blue*, and *The Game of Thrones* were logical suggestions based on the genre or the reader interest, even though *Gone Girl* and *Room* seemed less aligned in theme or tone. There are some challenges to these systems. Recommendations are driven by user behavior and not necessarily content similarity. Below are the results for the 5k file:

- Book ID tested: 948
- Book Title: *The Ultimate Hitchhikers Guide: Five Complete Novels and One Story, (Hitchhikers Guide to the Galaxy, #1-5)*

Top 5 most similar books returned:

- *Gone Girl*
- *Gathering Blue (The Giver, #2)*
- *Stardust*
- *A Game of Thrones (A Song of Ice and Fire, #1)*
- *Room*

## Problem 2 – Parameter Tuning

This section explores how different hyperparameters affect the behavior and output of the recommender system. The focus is on two main parameters, the number of latent factors and the similarity metric. Below is a table of some test runs.

Sampl Size (FN)	Factors	Metric	Top Match (Book ID 948) The Ultimate Hitchhikers Guide	Summary
5k	7	Cityblock	Gone Girl, Stardust, A Game of Thrones, Room	Some genre overlap, thrillers, dark fiction
5k	5	Euclidean	The Help, Pride and Prejudice, The Handmaid's Tale, Memoirs of Geisha	Stronger tonal, fantasy, satire
8k	5	Euclidean	Memoirs of Geisha, Catching Fire (The Hunger Games, #2, The Scorch Trails, Pride and Prejudice, Insurgent	More serious, literary fiction, less alignment with original books humorous tone

### Impact of the Number of Latent Factors

Varying the *Factors* hyperparameters across different values, 5 and 7, while holding other settings constant to observe its impact. The latent factors determine the dimensionality of the reduced item feature space, directly influencing how similarity is measured.

- With fewer factors, 5, the recommendations tended to be broader and more general. While some matches aligned with the input book's theme, others were less relevant.
- Increasing the factors to 7 added more nuance, capturing subtler patterns in the user item matrix. The recommendations appeared more tailored, suggesting stronger associations between books.
- However, very low or excessively high values for this parameter can cause degenerate behavior.

Across both the 5 and 8k datasets, the optimal number of factors seemed to be between 5 and 7, balancing computational cost with relevance.

### Impact of Similarity Metric

Testing *cityblock*, *Euclidean*, and *cosine* each metric has different assumptions and implications:

- Cityblock consistently returned recommendations that felt thematically and stylistically similar the input book
- Euclidean showed slightly broader diversity in results
- Cosine emphasized directional similarity in the feature space

Based on qualitative inspection, *cityblock* performed most consistently, especially when paired with at moderate number of latent factors.

### Interaction Between Factors and Similarity Metric

When varying both the number of factors and the similarity metric together, I observed the following:

- Certain combinations such as low factors with Euclidean led to unstable recommendations
- Higher factors combined with cityblock, or cosine proved the most coherent results
- When sample size increased form 5k, 8k, and 9k the effects became more pronounced. The larger dataset allowed for richer SVD representations.

### Problem 3 – User – User Recommendations

The final task was to use the original item to item recommendation system to build a user-to-user recommender. This version computes the similarity between users based on their rating patterns and then recommends books liked by similar users. The new system was implemented through wo scrips: *build\_user\_similarity.py* and *test\_user\_similarity.py*. In the table below, the different hyperparameters were adjusted to capture any similarities.

FN	Factors	Metric	Test ID	Top Similar Users	Similarity Scores	Noes
5k	7	Cosine	2	1,0,4,3,6	.862, .913, .986, .992, 1	Top books from users displayed
8k	5	L1	2	1,6,3,10,7	34.6, 55.5, 70.5, 71.1, 71,22	Distances vary wildly
5k	2	Cityblock	3	6,4,1,2,5	0.68, 2.94, 5.05, 5.51, 68.8	Distances are closer

Across all trials, the system behaved as expected. Users with similar book preferences tended to have overlapping high related books, and the quality of the results generally improved as the

dataset size increased. However, there were diminishing returns past a certain complexity, for example, for smaller sample sizes.

The item-to-item method excels in surfacing books like one already liked, the user-to-user shines when trying to personalize recommendations based on taste alignment. But it can suffer from a slow to start issues or the results are skewed when user data is limited.

## Conclusion

This assignment I explored two distinct approaches to building recommender system: item to item and user to user similarity. Both methods leveraged filtering and dimensionality reduction to identify meaningful patterns in the user behavior rating. The item-based approach proved with effective for surfacing books with shared characteristics, while the user-based system provided a more personalized recommendation experience by identifying the users with similar preferences. This is not terribly surprising if you think about it realistically. Users often rate on a lot of different methods for themselves, and it becomes more granular.