BACS3074

ARTIFICIAL INTELLIGENCE

202501 Session, Year 2024/25

Assignment Documentation

Project Title: Book Recommender System
Programme: RDS Y2S3
Tutorial Group: 4
Tutor: Dr. Ho Chuk Fong

Team members' data

No	Student Name	Student ID	Module In Charge
1	Saw Xue Yi	24WMR08887	Hybrid-Based
2	Cha Ming En	24WMR08850	Collaborative Filtering
3	Hea Hui Min	24WMR08865	Content-Based Filtering

1. Introduction

1.1. Problem Background

1.1.1. Background

In the digital age, information has become more accessible than ever, with technology fundamentally reshaping how people discover and consume content. Despite the growing dominance of digital media, reading remains a highly valued activity, offering significant cognitive, emotional, and social benefits. Research shows that regular readers enjoy improved mental well-being, higher community engagement, and greater life satisfaction *(Kavitha & Koteeswaran, 2023).* However, the overwhelming volume of books available today has introduced a new challenge — choosing the right book. With approximately 2.2 million new titles published each year globally *(Giordano, 2025)*, readers often experience decision fatigue, struggling to navigate an ever-expanding literary landscape.

The emergence of online bookstores and digital libraries has improved access to literature, but it has not solved the problem of effective discovery. Surveys show that around 33% of users find ebook search tools difficult to navigate, especially when exploring unfamiliar subjects *(Zhang et al., 2017)*. Consequently, many readers fall back on familiar authors, popular adaptations, or word-of-mouth recommendations *(Rutherford et al., 2024)*. While these strategies offer some guidance, they rarely lead to truly personalized or satisfying reading experiences, often resulting in wasted time and frustration.

Recognizing this growing problem, the field of artificial intelligence has introduced **recommender systems** — tools designed to filter massive amounts of data and suggest content tailored to users' unique preferences. Recommender systems are now fundamental across industries: Amazon recommends products based on past purchases and browsing history; Spotify curated music playlists based on listening behavior; Netflix personalizes movie and series suggestions; and Google News adjusts article feeds according to users' reading interests *(Bobadilla et al., 2013).*

Importantly, these systems have demonstrated substantial real-world impact. For example, Amazon attributes around 35% of its total sales to its recommendation engine *according to McKinsey, (LinkedIn, 2025),* showcasing the tremendous business value and user engagement potential of personalized recommendations. Similarly, Netflix has reported that about 80% of the content watched on their platform comes from algorithmic recommendations, highlighting how effective personalized systems can be in guiding user behavior. In the context of books, platforms like Goodreads use user reviews, ratings, and reading lists to suggest titles and authors tailored to individual tastes, helping users make more informed choices in an overwhelming literary marketplace.

However, despite their effectiveness, current recommender systems still face several key challenges. Research by *Priyanka Shahane (2021)*, highlights persistent issues such as the cold start problem (difficulty recommending for new users with little to no data), data sparsity (limited user interaction history), lack of diversity and novelty in suggestions, and limited explainability. Furthermore, most existing systems focus too narrowly on maximizing prediction accuracy without considering broader aspects like recommendation diversity, context awareness, and personalized experience depth — all of which are crucial for maintaining user satisfaction and long-term engagement.

Given these ongoing challenges, this project focuses on building a hybrid book recommendation system that combines collaborative filtering, content-based filtering, and user interaction tracking. Unlike traditional methods that rely heavily on bestsellers, marketing campaigns, or general popularity, our approach aims to adapt dynamically to individual users' preferences and behaviors. By narrowing the focus to user-centered personalization, this system can better support readers in finding books that genuinely match their interests, reduce decision fatigue, and make reading a more enjoyable and engaging experience.

Moreover, the benefits of a personalized recommender system extend beyond individual readers. Online platforms such as Amazon can increase sales by promoting more relevant products; libraries can boost borrowing rates by suggesting titles users are more likely to enjoy; and the broader literary ecosystem can thrive as lesser-known, high-quality works find their audiences. With the continued development of artificial intelligence and big data technologies, intelligent recommendation systems are evolving to offer not only greater accuracy but also richer diversity, novelty, and user satisfaction — making projects like ours a vital step forward in improving how readers navigate the ever-growing world of books.

1.1.2. Problem Statement

Despite the rapid advancement of recommender systems across industries, there remains a significant gap in their ability to support personalized book discovery, particularly for young readers. Current book recommendation platforms are predominantly designed with marketing goals in mind, often emphasizing bestsellers, popular authors, or broad genres rather than offering deeply personalized suggestions based on individual content preferences and user behaviors. This approach frequently overlooks the nuanced interests of readers, leading to recommendations that may not align with a reader's true tastes or reading intentions.

The situation is even more critical in the domain of children's literature. Most existing book recommendation systems cater primarily to adult readers and fail to consider the unique developmental needs of young audiences. Factors such as age appropriateness, educational value, reading level, and thematic relevance are often ignored, resulting in suggestions that are either unsuitable or fail to engage young readers effectively. As *Mathew et al. (2016)* highlight, recommender systems must be adapted to meet the specific needs of their intended audience, yet children's literature remains an underserved area in current research and application.

This gap has tangible impacts. Children, parents, and educators struggle to identify books that not only match reading abilities but also support cognitive and emotional development. Inadequate recommendations can lead to frustration, reduced reading motivation, and missed opportunities for educational enrichment. Furthermore, relying solely on genre or author similarity fails to capture the full complexity of a child’s reading preferences and developmental needs.

While some platforms like Goodreads offer basic recommendation services, they typically do not integrate multi-dimensional attributes, such as book content analysis, user interaction data, and developmental appropriateness, into a cohesive recommendation strategy. As a result, current systems lack the depth and personalization required to meaningfully assist young readers in discovering books that resonate with their interests and support their growth.

Addressing this problem is crucial for fostering a more meaningful reading experience for young audiences. A hybrid recommendation system that combines content-based filtering, collaborative filtering, and real-time user interaction tracking can bridge this gap by delivering personalized, age-appropriate, and educational book recommendations. By focusing not only on popularity but also on individual user needs and item characteristics, such a system would improve recommendation accuracy, enhance reader satisfaction, and promote a lifelong love of reading among children. This project aims to develop such a system, aligning with broader educational and developmental goals by empowering young readers through better, more thoughtful book discovery.

1.1.3. Research Question

How can a hybrid recommender system combining content-based filtering, collaborative filtering, and real-time user interactions be designed to provide personalized, age-appropriate, and educational book recommendations for children, improving recommendation accuracy and user satisfaction?

1.1.4. Hypothesis

If a hybrid recommender system integrates book content features, user historical preferences, and dynamic interaction data, then it will significantly improve the accuracy, relevance, and satisfaction levels of book recommendations compared to systems relying only on popularity or basic genre matching.

1.2. Objectives/Aims

- To develop a hybrid book recommendation system that integrates content-based filtering and collaborative filtering techniques, ensuring personalized book suggestions tailored to the user's preferences, past reading behavior, and similarities with other users.

- To evaluate the performance of different recommendation techniques, using metrics like precision, RMSE, and other relevant evaluation criteria to determine the effectiveness of the content-based and collaborative filtering components in providing accurate and relevant book recommendations.

- To assess user engagement and satisfaction by analyzing user interactions, such as click-through rates, book ratings, and retention, to measure the system's ability to provide relevant and diverse book recommendations that match user preferences.

- To improve recommendation accuracy over time by collecting and analyzing user feedback, refining the model iteratively, and comparing recommendation performance before and after model adjustments using metrics such as precision, recall, and RMSE.

1.3. Motivation

This project is dedicated to developing an intelligent book recommendation system that will improve user experience through big data analysis and personalized recommendation technology, bring new business opportunities to the publishing industry, and promote the development of the overall culture in society

Potential Commercialization :

- Increase the sales of book
    - Through a personalized book recommendation system, users can find books that they are interested in, thereby increasing book sales. For publishers and bookstores, this book recommendation system can significantly increase the sales, and can also promote the books through in-app, because nowadays more people pay more attention to mobile applications rather than traditional advertising.
- User data analysis based on the user interest field
    - Enterprise can collect user’s reading data, analyze user behaviour and identify which kind of books are more popular with readers, thereby optimizing recommendation systems and marketing strategies.
- Driving growth in e-commerce and digital publishing
    - Book recommendation system can be seamlessly integrated into e-commerce platforms such Amazon or e-library services, bringing growth opportunities for digital content sales and membership subscription models

Social Impacts :

- Promoting a culture reading for all people
    - Recommendation system reduces the difficulty for users to find high-quality reading content, encourages more people form a continuous reading habit, promotes the overall reading level of society and helps to build a knowledge-based society
- Narrowing the information gap and supporting educational equity
    - Through intelligent recommendations, users from different backgrounds and regions can access a variety of book resources to avoid the information island effect. At the same time, the recommendations system can also provide

personalized learning resources for students, teachers and learners to support the development of educational lifelong learning

- Promoting cultural diversity
    - The recommendations system helps niche themes and emerging authors works get more exposure opportunities to enrich the content ecology and protect cultural diversity and innovative spirit

2. Research Background

2.1. Background of the applications

Recommender systems (RSs) have become fundamental tools for personalized information retrieval in the era of information overload. As online content continues to grow exponentially, users increasingly rely on intelligent systems to filter and suggest items that align with their preferences *(Aggarwal, 2016).* Originally developed as information filtering technologies, RSs aim to enhance decision-making processes by offering tailored suggestions, thereby improving user engagement and satisfaction across various domains, including e-commerce, entertainment, and education.

Recommender systems are typically classified into several main types, each employing distinct methodologies to generate personalized suggestions for users. Content-based filtering (CBF) is a key technique in recommender systems that recommends items based on their features or attributes, which match a user's past preferences *(Lops et al., 2010)*. This method avoids the cold start problem for new items and provides transparent recommendations. However, CBF can struggle with feature engineering, limited content diversity, and overspecialization, as it tends to suggest items similar to those already liked by the user.

Content analysis techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) have been widely used for item similarity evaluation. TF-IDF evaluates the importance of terms in a document relative to their frequency across a corpus, distinguishing keywords between items. On the other hand, BERT (Bidirectional Encoder Representations from Transformers), a more advanced deep learning model, captures contextual relationships between words, allowing for a deeper understanding of textual content *(Devlin et al., 2019).* Unlike TF-IDF, which primarily relies on term frequency, BERT leverages bidirectional context modeling to better comprehend semantic nuances in the text.

Ilona (2021) demonstrated BERT's superior performance over TF-IDF in predicting article relevance on a pregnancy-related platform, attributing this to BERT's context-aware capabilities. However, BERT's higher computational requirements and need for extensive training data make it more resource-intensive compared to TF-IDF. While TF-IDF is simpler and faster, it may overlook content subtleties. The choice between these methods depends on the system's goals and available resources, with BERT being preferable for more complex recommendation tasks.

Secondly, Collaborative filtering (CF) is another prominent technique in recommender systems that suggests items based on the preferences of similar users *(Gong et al., 2009).* The process involves identifying users with tastes similar to the target user, aggregating their preferences, and recommending items that these users liked but the target user has not yet encountered. CF excels in capturing subjective tastes and diversifying recommendations, but it faces challenges such as the cold start problem, sparsity, and scalability. Memory-based CF utilizes user-item matrices to calculate similarities, while model-based CF employs machine learning models to mitigate these issues. Gong et al. (2009) describe a hybrid CF model that combines memory-based CF to fill in missing ratings and model-based CF to improve prediction accuracy.

Thirdly, Hybrid recommender systems combine two or more recommendation techniques to overcome the weaknesses of individual methods and enhance performance. *Mathew et al. (2016)* propose a hybrid system that integrates content-based filtering (CBF), collaborative filtering (CF), and association rule mining to improve recommendation accuracy. By combining CBF and CF, the system addresses the cold start problem (CBF recommends items for new users based on content) and sparsity (CF leverages ratings from multiple users). The incorporation of association rule mining further refines recommendations by detecting co-occurrence patterns between items, resulting in a more comprehensive and diverse recommendation experience.

Book recommender systems face unique challenges such as genre ambiguity, subjective content qualities, and the significance of metadata (e.g., author, publication date) in user decisions. These factors complicate the recommendation process, necessitating more tailored approaches compared to other domains. Despite the proliferation of book recommender systems, there is a noticeable gap in systems designed specifically for children's literature. Children’s books require special consideration of factors such as age-appropriate content, educational value, and reading complexity, elements that are often overlooked by systems built for adult audiences.

There is a noticeable lack of recommender systems tailored specifically for children's literature. Existing systems, which focus on adult literature, fail to address the unique needs of young readers, such as age appropriateness, educational themes, and reading level. *Mathew et al. (2016)* emphasize the importance of adapting recommender systems to meet the needs of their target audience. In the case of children's books, a system needs to incorporate features that ensure the recommendations are both educational and developmentally appropriate.

This project aims to develop a hybrid book recommender system for children’s literature, combining content-based filtering, collaborative filtering, and user interaction data to recommend age-appropriate and educational books. By addressing the gaps in current systems, this approach will ensure that recommendations are tailored to the specific needs of young readers, improving recommendation accuracy and user satisfaction.

2.2. Analysis of selected tool with any other relevant tools

Tools comparison	Remark	Jupyter Notebook	Google Colab	Python
Type of license and open source license	State all types of license	3-Clause BSD License	Don’t have specific licensing requirements, Free service, subject to Google’s proprietary terms of service, not an open source project	Python Software Foundation (PSF) license is used for Python itself and the Python documentation
Year founded	When is this tool being introduced?	2014	2017	1991
Founding company	Owner	Fernando Pérez and Brian Granger separated Project Jupyter from the IPython project and launched Project Jupyter	Google (developed by Google Research)	Python was developed by Guido van Rossum and is maintained by the Python Software Foundation
License Pricing	Compare the prices if the license is used for development and business/commercialization	Free to use and didn’t require payment. While Enterprise or hosted version may require payment of platform fees	The free versions provides basic resources while the paid version (Colab Pro / Pro +) provides higher-specification GPUs and longer running time, and the price is approximately USD 9.99 to 49.99 per month	Free and open source
Supported features	What features that it offers?	Supports Live code execution, data visualization, markdown integration, code sharing and collaboration, extension ecosystem and kernel support	Supports Python programming, integrate deep learning libraries such as TensorFlow and Keras, support GPU/TPU acceleration, seamless integration with Google Drive, and multi-person collaborative editing	Support multiple programming paradigms, large standard library and wide range of third-party libraries
Common applications	In what areas this tool is usually used?	Data analysis, data cleaning, machine learning model development and visualization	Machine learning model training, small scale data analysis, deep learning experiment	Data Science, Machine Learning and Artificial Intelligence
Customer support	How the customer support is given, e.g. proprietary, online community, etc.	Github Issues, Stack Overflow, rich official documentation, and tutorial resources	The basic version is supported by the community(Stack Overflow) while paid versions users have access to priority support channels	Python is supported through online communities (such as Stack Overflow)
Limitations	The drawbacks of the software	Insufficient performance when running large projects or high-performance applications. Not suitable for developing large systems or web	The free version has limited running time (usually a maximum of 12 hours), limited memory and GPU resources.	The running efficiency is relatively slow, especially when a lot of calculations are required.

		applications	Large-scale or long-term training projects are easily interrupt due to insufficient resources, and it is not ideal for handling very large datasets	

2.3. Justify why the selected tool is suitable

The Anaconda distribution with Jupyter Notebook was selected as the development tool for this project due to

- Processing large-scale datasets
  The dataset files involved in this project are large, so an efficient data processing and analysis tool is needed. As a flexible development environment that supports multiple data analysis libraries such as Pandas, NumPy, etc, Jupyter Notebook can effectively process large-scale datasets, especially when analyzing and visualizing data, which greatly facilitates our debugging and optimization.

- Memory management
  When the Jupyter Notebook is used with Anaconda, it can use virtual environments to manage memory and computing resources, which is particularly important for processing large datasets. By properly configuring and allocating resources, the performance of the local machine can be maximized, and problems such as memory overflow or insufficient computing resources can be avoided.

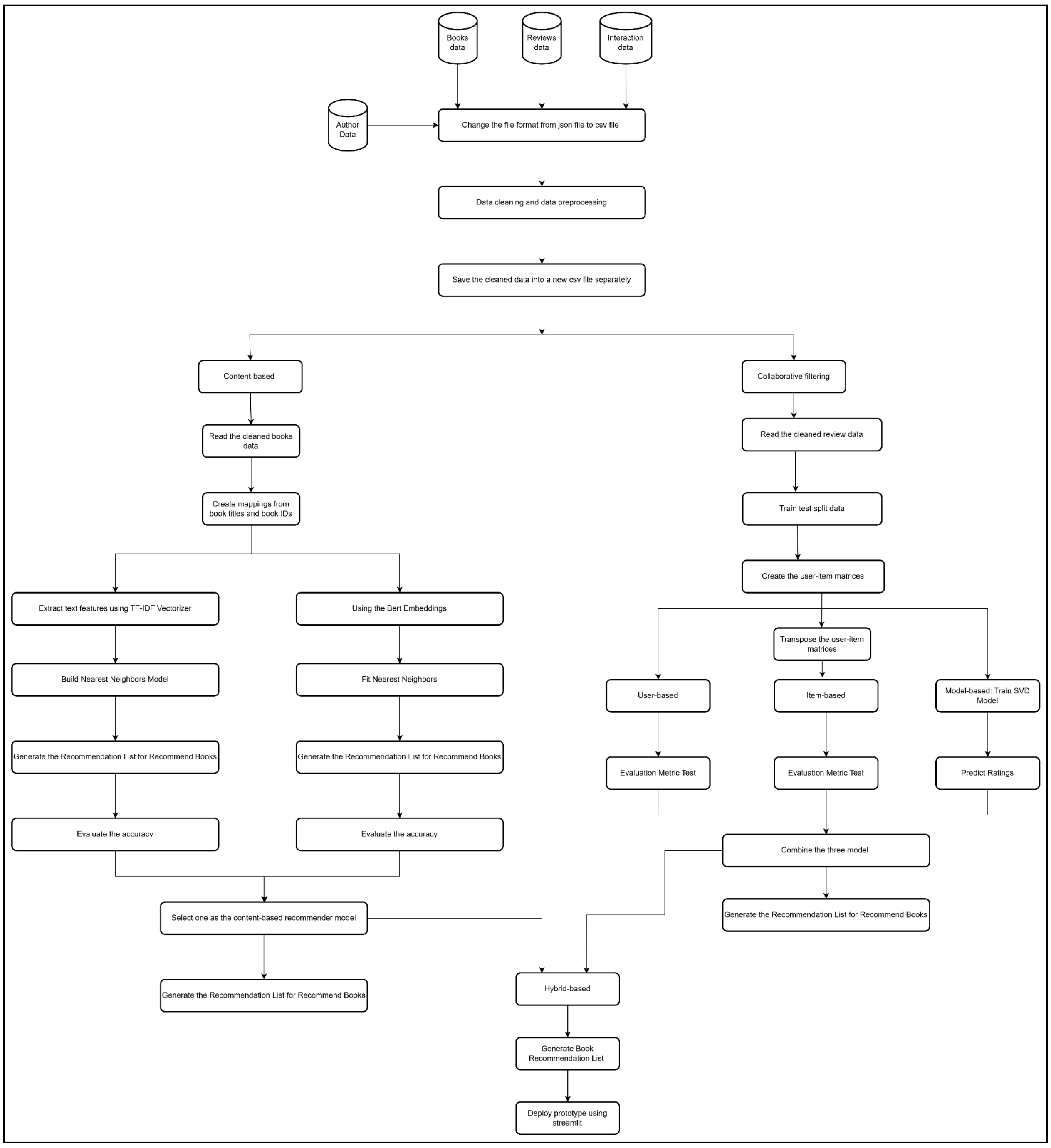
- Interactive Programming
  Due to the interactive nature of Jupyter Notebook, developers can run the code step-by-step and instantly view the data processing and results of each step. This is particularly useful for debugging and data preprocessing, especially when processing big data, to ensure the accuracy of each step by step execution.

The reason that we didn't choose Google Colab is memory and computing resource limitations. Although the free version of Google Colab provides GPU support due to its memory and computing resource limitations, it is often prone to insufficient resources when processing large-scale datasets. In addition, Google colab running time is also capped, and long-term stable operation cannot be guaranteed, so it is not for our long-term, large training tasks.

3. Methodology

3.1. System flowchart/activity diagram

Diagram Link:
https://app.diagrams.net/#G1X6tL3qWkdFOCilvkP29UWoKQHLuyCe9E#%7B%22pageId%22%3A%22EuJ37S_J_I7oVJ7snFLI%22%7D



Books data
Reviews data
Interaction data
Author Data
Change the file format from json file to csv file
Data cleaning and data preprocessing
Save the cleaned data into a new csv file separately
Content-based
Collaborative filtering
Read the cleaned books data
Read the cleaned review data
Create mappings from book titles and book IDs
Train test split data
Create the user-item matrices
Extract text features using TF-IDF Vectorizer
Using the Bert Embeddings
Transpose the user-item matrices
Build Nearest Neighbors Model
Fit Nearest Neighbors
User-based
Item-based
Model-based: Train SVD Model
Generate the Recommendation List for Recommend Books
Generate the Recommendation List for Recommend Books
Evaluation Metric Test
Evaluation Metric Test
Predict Ratings
Evaluate the accuracy
Evaluate the accuracy
Combine the three model
Select one as the content-based recommender model
Generate the Recommendation List for Recommend Books
Generate the Recommendation List for Recommend Books
Hybrid-based
Generate Book Recommendation List
Deploy prototype using streamlit

This diagram describes in detail the entire recommendation system project from data collection, preprocessing to model development and recommendation generation. To improve the recommendation effect, the system uses a combination of content-based recommendations, collaborative filtering recommendations, which are hybrid recommendation methods.

First of all, the recommendation system's data sources are divided into four major categories: Books (book information), Ratings (user ratings), Users (user information), and Authors. The data from various sources is converted into a unified format so that the data tables can be correctly associated. The data is then cleaned and preprocessed to handle missing values, remove abnormal data, and divide it into different subsets based on the recommendation method used later, which facilitates subsequent modeling.

The content-based filtering recommendation function suggests other books similar to the ones users are interested in based on the books' properties. It is mainly based on two technologies: TF-IDF vectorization and BERT vector embedding. The TF-IDF (term frequency-inverse document frequency) method converts the book's content into a numerical vector and extracts key words from the text to represent each book's characteristics. When creating the TF-IDF model, we select the 500 most important words, remove common English stop words and words with extremely low frequency, and the resulting TF-IDF matrix accurately represents the importance of each word in the book.The nearest neighbor model is fitted in the TF-IDF vector space using cosine similarity to recommend other books that are similar to the target book. BERT vector embedding encodes the book text into a dense numeric vector using the Sentence Transformers library's pre-trained BERT model, which can capture the text's deep semantic information rather than relying solely on word frequency. The nearest neighbor model in the BERT embedding space uses cosine similarity calculation to recommend the most relevant books based on semantic similarity. At the end select one as the content-based recommender model based on their results.

Collaborative filtering mainly relies on user historical behavior data, such as rating records, to infer preferences and generate recommendations. After preprocessing, the system constructs a user-item interaction matrix and develops three models: user-based, item-based, and model-based collaborative filtering. User-based filtering identifies similar users and recommends books they like, while item-based filtering recommends similar books based on user rating patterns. Model-based filtering applies Singular Value Decomposition (SVD) to predict unknown ratings. After evaluating the models using RMSE, the system combines their results to enhance recommendation robustness.

The final hybrid model merges content-based and collaborative to generate the final recommendations. By integrating these approaches, the system benefits from both user behavior patterns (collaborative filtering) and book characteristics (content-based filtering), thereby addressing the limitations of each method individually.

To bring the system to life, it is deployed via Streamlit, providing an interactive interface where users can search for books and receive recommendations.

3.2. Description of dataset

Source of the dataset :
https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html#datasets

The dataset that we use for this project comes from the Goodreads book dataset publicly released by the University of California, San Diego (UCSD). We selected the sub-dataset of the dataset classified by the genre, specially the data of the Children category. The category includes 124,082 book information, 10,059,349 user interaction data (such as ratings, likes, favorites, etc.) and 734,640 detailed user reviews. The dataset file that we used is goodreads_books_children.json.gz, goodreads_interactions_children.json.gz and goodreads_reviews_children.json.gz.

1. goodreads_books_children.json.gz

Field Name	Description
isbn	International Standard Book (10 digits), a unique identifier for each book
text_reviews_count	The number of written reviews the book
series	The series to which the book belongs (if any)
country_code	The code for the country of publication
lanaguge_code	The code for the book's language
popular_shelves	The bookshelf this book is most frequently added to on Goodreads("Read","Want to Read" or "Currently Reading")
asin	Amazon standard identification code, used to identify books on Amazon
is_ebook	Indicates whether this is an e-book(TRUE / FALSE)
average_rating	The average rating of the book on Goodreads
kindle_asin	If it is a Kindle version, its ASIN code is displayed
similar_books	A list of other book IDs that are similar to this book
description	Detailed description or introduction to the book
format	The format of the book (e-book)
link	Link the book on Goodreads
authors	Author names, or a list if there are multiple authors
publisher	Name of publisher
num_pages	The number of pages in the book
publication_day	"Day" in the publication date
publication_month	"Month" in the publication date

publication_year	Year of publication
isbn_13	International Standard Book Number (13 digits), new ISBN code
edition_information	Version information (first edition, revised edition)
url	Link to the book’s detail page on Goodreads
image_url	Link to image of book cover
book_id	Book unique identification code used internally by Goodreads
ratings_count	The number of ratings this book has received
work_id	The ID of a “work” on Goodreads (a work can have multiple versions / edition)
title	The full title of the book (including series information)
title_without_series	The title of the book (without series information)

2. goodreads_interactions_children.json.gz

Field Name	Description
user_id	User unique identification code
book_id	Book unique identification code
is_read	Has it been read (TRUE / FALSE)
rating	User rating of the book (1 - 5 mark)
review_id	A unique identifier for each review record.
date_added	The time when the user added the book to his / her “interest to read” or “reading list”
date_updated	The last time a user updated their reading status or commented on a content
read_at	The specific time when the user marked the book as “finished reading”. If the user has not finished reading it, this field may be empty
start_at	The time when the user marked “started reading” the book

3. goodreads_reviews_children.json.gz

Field Name	Description
user_id	User unique identification code

book_id	Book unique identification code
review_text	Review content that user write
date_added	Date the review was added
date_updated	Date the review was updated
rating	User rating of the book

Besides the three files, we also retrieve the dataset goodreads_book_authors,json.gz file in order to extract the author information as the authors stored at the book dataset is only the author id, which is less meaningful in the recommendation system.

4. goodreads_book_authors,json.gz

Field Name	Description
average_rating	Average rating score of the author
author_id	Author unique identification code
text_reviews_count	Number of text review the author received
name	Author name
ratings_count	Number of ratings the author received

As all the datasets obtained are in the json format file, the first step we carried out is to convert all the datasets into the csv file for the ease of analysis and manipulation.

3.3. Description of algorithm(s)

In this project, a hybrid-based recommendation system is developed by integrating multiple recommendation techniques to provide users with more accurate and personalized suggestions. Below are the algorithms and techniques implemented.

Techniques	Algorithms
Content-Based Filtering	Content-based filtering recommends books similar to those the user has shown interest in, based on the attributes of the books themselves. Two main techniques are used: ● **TF-IDF Vectorizer** The book content is transformed into numerical vectors using Term Frequency-Inverse Document Frequency (TF-IDF). A TF-IDF model is built by extracting the top 500 most important words, ignoring common English stop words and rare words appearing in fewer than 5 books. The resulting TF-IDF matrix represents the importance

	of words in each book. To measure similarity between books, a Nearest Neighbors model is fitted using cosine distance on the TF-IDF vectors. Based on this, books with similar content are recommended. ● **BERT Embeddings** Book content is encoded into dense numerical vectors using a pre-trained BERT model from the Sentence Transformers library. These embeddings capture the semantic meaning of the text beyond simple word frequency. A Nearest Neighbors model is then fitted using cosine distance on the BERT embeddings. Based on the semantic similarity of the content, books are recommended by finding the most similar embeddings to a given query.
Collaborative Filtering	Collaborative filtering recommends books based on the behavior of similar users or items. Three collaborative filtering techniques are combined: ● **User-Based Collaborative Filtering** It generates book recommendations by first identifying similar users through K-nearest neighbors (KNN) on a user-item rating matrix using cosine similarity. It then selects candidate books rated by these similar users (excluding those already read by the target user), computes weighted scores by multiplying ratings with user similarity values, and normalizes these scores by the sum of similarities. Finally, it ranks books by these normalized scores to provide personalized recommendations ● **Item-Based Collaborative Filtering** It recommends books by analyzing similarities between books rather than users. It first constructs an item-item similarity matrix using cosine similarity on the transposed user-item rating matrix, then identifies books similar to those the target user has already rated. For each rated book, the algorithm finds its nearest neighbors (excluding itself), aggregates similarity scores for candidate books, and recommends those with the highest cumulative similarity scores ● **Model-Based Collaborative Filtering** Uses Singular Value Decomposition (SVD) to predict book ratings and generate recommendations. The algorithm decomposes the user-item interaction matrix into latent factors, capturing hidden patterns in user preferences and book characteristics. After training on 80% of the rating data, it predicts ratings for unread books by

	reconstructing the matrix factorization, then recommends the highest predicted-rated books the target user hasn't yet rated. Recommendations merge predicted ratings with book metadata for interpretable results.
Hybrid-Based Recommendation	The system combines both content-based filtering and collaborative filtering results to generate the final recommendations. By integrating these approaches, the system benefits from both user behavior patterns (collaborative filtering) and book characteristics (content-based filtering), thereby addressing the limitations of each method individually.
Popularity-Based Recommendation	To highlight trending books, a popularity-based ranking is implemented. A weighted rating (WR) formula is used to balance the number of votes and average ratings: Formula: WR = [(v * R)/(v + m)] + [(m * c)/(v + m)] Where, ● v = number of votes for the books; ● m = minimum votes required to be listed in the chart; ● R = average rating of the book; and ● C = mean vote across the whole report. This method ensures that books with a high number of ratings and high average scores are ranked higher, making the trending list more reliable.

Nearest neighbour:

- In this project, we are using the nearest neighbors algorithm to implement a book recommender system. The nearest neighbor algorithm is a similarity-based recommendation technology. It is to find other users or books that are most similar to the current user or current book. Based on the preferences or characteristics of these similar objects, recommend books that may be of interest to users.

User-based Collaborative Filtering:

- Nearest Neighbors compares the similarity of interest or ratings between different users. It is to find other users with similar interests to the target user. It recommends books liked by these similar users to the target user.

Item-based Collaborative Filtering:

- Nearest Neighbors compares the similarity of user ratings between different books. It is to find other books that the user has liked high ratings between different books and recommend these similar books to the user.

We are using the **Cosine Similarity** to measure the similarity between users or books. Based on the similarity, select a group of neighbors that are closest to the current user or current book. After combining the behavior or attributes of the nearest neighbors, it will generate a recommended book list.

3.4. Proposed test plan/hypothesis

Our hypothesis focuses on evaluating the user's satisfaction with the recommendations generated by the system. However, since there are various techniques available for developing recommender systems, we have chosen to evaluate the performance of these techniques as part of our test plan to assess the overall performance of our system..

Evaluation Metrics:

1. Content-Based Filtering
    - Metric: Precision@K
    - Test: Since the original dataset contains the "similar_books" column, we compare the recommendations generated by the system with the ground truth by counting how many of the recommended books match those in the "similar_books" column.
    - Interpretation: A higher precision score indicates that the technique is more effective in recommending books based on attributes such as descriptions, authors, and language.

2. Collaborative Filtering
    - Metric: Root Mean Squared Error (RMSE) between predicted and actual user ratings.
    - Test: Use a hold-out validation set (20% of user ratings) to measure prediction error.
    - Interpretation: A lower RMSE indicates that the technique produces more accurate predictions, as the difference between the predicted and actual ratings is smaller.

4. Result

4.1. Results

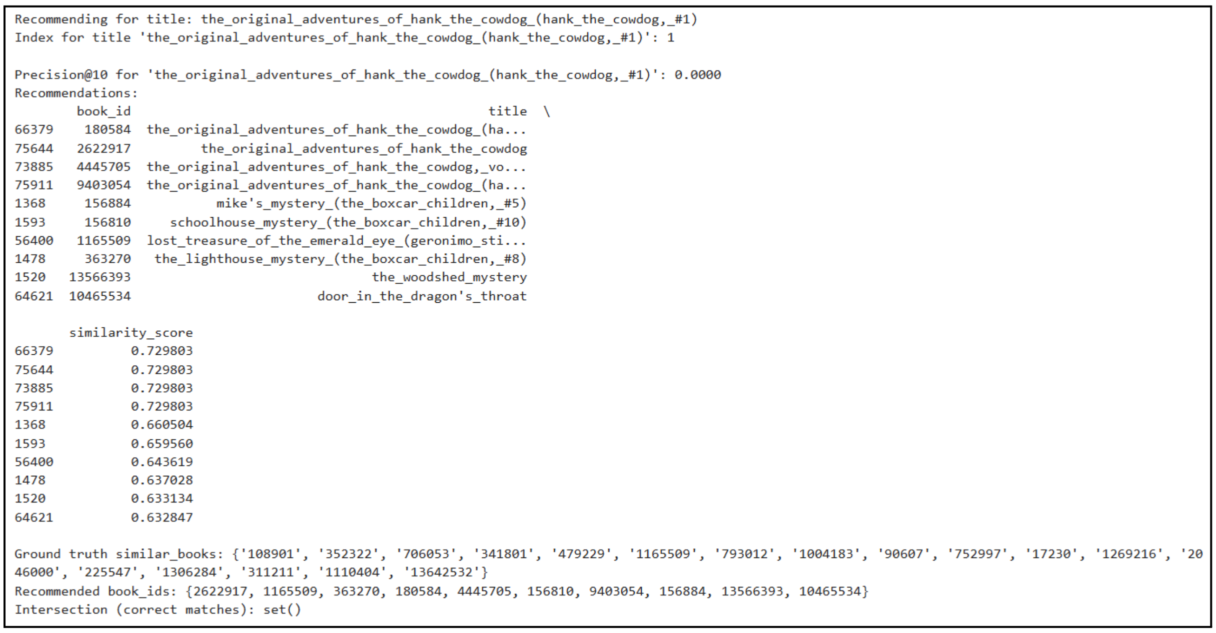
4.1.1. Evaluation on Content-Based Filtering Techniques (Precision@K)

- Firstly, we generated the list of book recommendations by using the corresponding model, one is using TF-IDF Vectorizer for calculating the difference between the words, another model is using the Bert Embeddings.
- Then, we compare the list of book recommendations towards the column 'similar_book' to evaluate how many matches on the recommendation and the dataset predefined similar book.

By passing in the same query towards the two models for evaluating performance:

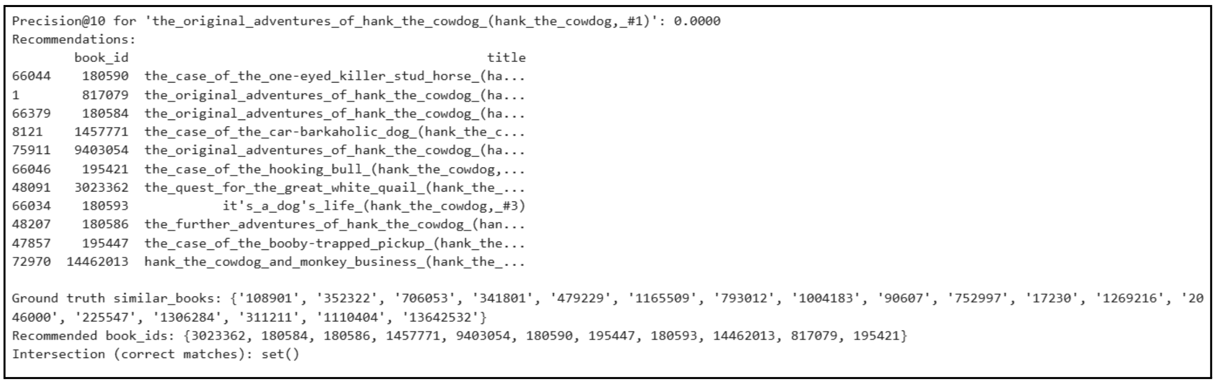
- title = "the_original_adventures_of_hank_the_cowdog_(hank_the_cowdog,_#1)"

1. TF-IDF Vectorizer

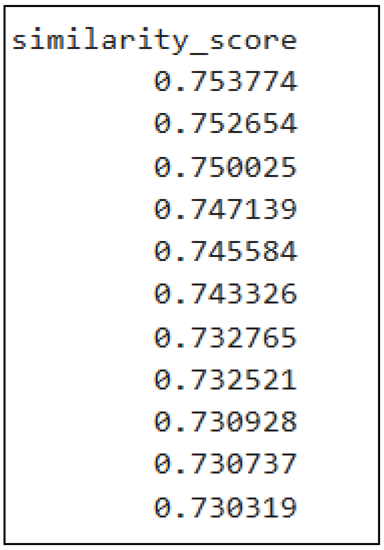


```
Recommending for title: the_original_adventures_of_hank_the_cowdog_(hank_the_cowdog,_#1)
Index for title 'the_original_adventures_of_hank_the_cowdog_(hank_the_cowdog,_#1)': 1

Precision@10 for 'the_original_adventures_of_hank_the_cowdog_(hank_the_cowdog,_#1)': 0.0000
Recommendations:
        book_id                                              title  \
66379    180584  the_original_adventures_of_hank_the_cowdog_(ha...
75644   2622917         the_original_adventures_of_hank_the_cowdog
73885   4445705  the_original_adventures_of_hank_the_cowdog,_vo...
75911   9403054  the_original_adventures_of_hank_the_cowdog_(ha...
1368     156884           mike's_mystery_(the_boxcar_children,_#5)
1593     156810     schoolhouse_mystery_(the_boxcar_children,_#10)
56400   1165509  lost_treasure_of_the_emerald_eye_(geronimo_sti...
1478     363270   the_lighthouse_mystery_(the_boxcar_children,_#8)
1520   13566393                              the_woodshed_mystery
64621  10465534                      door_in_the_dragon's_throat

       similarity_score
66379          0.729803
75644          0.729803
73885          0.729803
75911          0.729803
1368           0.660504
1593           0.659560
56400          0.643619
1478           0.637028
1520           0.633134
64621          0.632847

Ground truth similar_books: {'108901', '352322', '706053', '341801', '479229', '1165509', '793012', '1004183', '90607', '752997', '17230', '1269216', '20
46000', '225547', '1306284', '311211', '1110404', '13642532'}
Recommended book_ids: {2622917, 1165509, 363270, 180584, 4445705, 156810, 9403054, 156884, 13566393, 10465534}
Intersection (correct matches): set()
```

2. Bert Embeddings



```
Precision@10 for 'the_original_adventures_of_hank_the_cowdog_(hank_the_cowdog,_#1)': 0.0000
Recommendations:
        book_id                                              title
66044    180590  the_case_of_the_one-eyed_killer_stud_horse_(ha...
1        817079  the_original_adventures_of_hank_the_cowdog_(ha...
66379    180584  the_original_adventures_of_hank_the_cowdog_(ha...
8121    1457771  the_case_of_the_car-barkaholic_dog_(hank_the_c...
75911   9403054  the_original_adventures_of_hank_the_cowdog_(ha...
66046    195421  the_case_of_the_hooking_bull_(hank_the_cowdog,...
48091   3023362  the_quest_for_the_great_white_quail_(hank_the_...
66034    180593            it's_a_dog's_life_(hank_the_cowdog,_#3)
48207    180586  the_further_adventures_of_hank_the_cowdog_(han...
47857    195447  the_case_of_the_booby-trapped_pickup_(hank_the...
72970  14462013  hank_the_cowdog_and_monkey_business_(hank_the_...

Ground truth similar_books: {'108901', '352322', '706053', '341801', '479229', '1165509', '793012', '1004183', '90607', '752997', '17230', '1269216', '20
46000', '225547', '1306284', '311211', '1110404', '13642532'}
Recommended book_ids: {3023362, 180584, 180586, 1457771, 9403054, 180590, 195447, 180593, 14462013, 817079, 195421}
Intersection (correct matches): set()
```

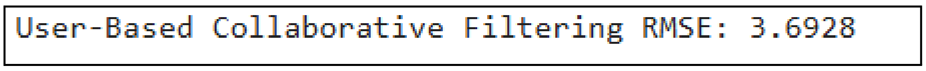


```
similarity_score
        0.753774
        0.752654
        0.750025
        0.747139
        0.745584
        0.743326
        0.732765
        0.732521
        0.730928
        0.730737
        0.730319
```

4.1.2. Evaluation on Collaborative Filtering Techniques (RMSE)

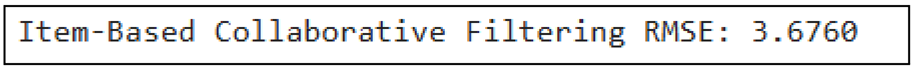
- We split the data into train (80%) and test data (20%); in which the model trained on the train data and evaluated through the test data.
- The user-based and item-based (Memory-based CF) are using the KNN for the model fitting. Then, we used the formula formed to make predictions on test data and calculate the RMSE error.
- The model-based CF, which is the SVD, is from the Python model library, hence no manual formula required to calculate the RMSE.

1. **User-Based**



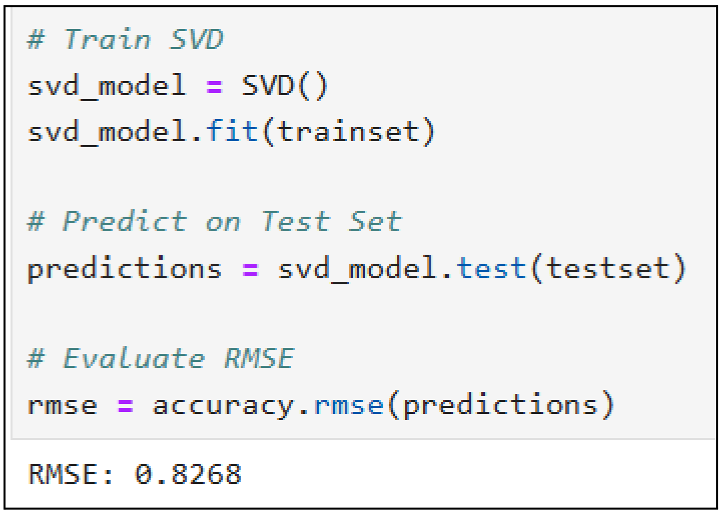
```
User-Based Collaborative Filtering RMSE: 3.6928
```

2. **Item-Based**



```
Item-Based Collaborative Filtering RMSE: 3.6760
```

3. **Model-Based (SVD)**



```
# Train SVD
svd_model = SVD()
svd_model.fit(trainset)

# Predict on Test Set
predictions = svd_model.test(testset)

# Evaluate RMSE
rmse = accuracy.rmse(predictions)
```

```
RMSE: 0.8268
```

4.2. Discussion/Interpretation

4.2.1. Content-Based Filtering

- The results show that the recommendation list generated by the two techniques does not match any of the books listed at the similar book columns.
- However, this result cannot directly conclude that both the techniques are not performed well as the model is trained on a cleaned dataset, which means some of the book id listed at the similar book columns may not exist in the list of books that can be recommended by the system.
- As when we look through the result generated, we can still find some similarities among the books' attributes, which has proven that both the techniques have successfully group those books with similar attributes together and generate recommendations to the users.
- The reason for choosing Bert Embeddings as the final content-based filtering model is because the similarity score generated along with the recommendation list from the Bert Embeddings are having higher scores compared to the TF-IDF Vectorizer. As the Bert Embeddings consider the context or the meaning of the words, while the TF-IDF only considers the word frequency, hence Bert Embeddings was chosen as it is more suitable for the case when the user searches for keywords in order to get book recommendations.

4.2.2. Collaborative Filtering

- In the context of collaborative filtering techniques, the Root Mean Squared Error (RMSE) is a metric used to evaluate how well the model's predictions match the actual user ratings. Lower RMSE values indicate better prediction accuracy, as the model is closer to the true ratings.
- The user-based collaborative filtering model works by finding users with similar preferences and recommending items based on those similarities. The RMSE of 3.6928 suggests that the predictions are somewhat off from the actual ratings, but it's not particularly bad. It shows moderate performance in predicting user preferences.
- This model recommends items based on similarities between items themselves. The RMSE of 3.676 is slightly better than the user-based model, indicating that item similarities might be a bit more reliable for your dataset than user similarities. This suggests the importance of understanding item relationships in your dataset.
- The SVD model significantly outperforms the user-based and item-based models, with a much lower RMSE of 0.8268. This suggests that SVD is capturing latent factors in the data that both user-based and item-based methods are missing. SVD is a matrix factorization technique that captures complex patterns in the data, leading to more accurate predictions.
- Although the results show that SVD performs exceptionally well compared to the other two models, we have decided to incorporate a hybrid collaborative filtering model as it is more efficient and likely to offer more personalized recommendations. This decision stems from the fact that SVD mainly focuses on predicting a user's rating for previously unrated books. However, rating scores are not always the sole consideration when people select new books.

- By incorporating the other two models — one based on user similarity and the other based on similar preferences — we can generate a broader recommendation list. Using SVD to predict ratings and sort the recommendations is the most optimal approach, as it will provide a more diverse set of suggestions and ultimately enhance the performance of the recommender system.

5. Discussion and Conclusion

5.1. Achievements

This project successfully created and implemented a personalized book recommendation system with basic functions, meeting the expected goal of improving user reading experience and recommendation relevance. The system can provide users with more accurate and personalized book recommendation results using Hybrid-based Filtering (which combines two recommendation strategies, Content-Based Filtering and Collaborative Filtering), based on user interests, historical behaviors, and other users' reading habits.

In terms of content recommendation, the system uses TF-IDF feature extraction technology to convert the book's title and description into a vector representation, and the Cosine Similarity method to measure the similarity between different books, resulting in personalized recommendations based on content. This module successfully established a complete text processing process, including data cleaning, feature engineering, model training, and similarity calculation, as well as demonstrated the effectiveness of content features in recommending in the absence of user rating data.

In terms of collaborative filtering, the system implements a neighbor user recommendation model based on user rating data, which can mine user groups with similar interests by analyzing the similarity of ratings between users, and push books that the current user may be interested in. Through offline testing on static data sets, the collaborative filtering model successfully demonstrated the basic capabilities of user interest migration trends and similarity mining between users to a certain extent, verifying the feasibility of collaborative filtering for effective recommendations in the absence of clear content labels.

In terms of user management, this project created modules for user registration, login authentication, and database storage to support local account creation and management. Users can view the recommendation results, establishing an interactive mechanism between users and the recommendation system. This lays the groundwork for future developments in dynamic user behavior analysis and real-time recommendation.

Despite limited computing resources and interactive functions, the project successfully evaluated the recommendation system's effectiveness on a small data set using offline testing methods. The content recommendation section displays reasonable recommendation matching results via similarity scores, while the collaborative filtering section employs RMSE (root mean square error) as an evaluation indicator to confirm the closeness between the model's predicted score and the actual score. These experimental results demonstrate that the recommendation system as a whole can achieve certain levels of recommendation accuracy and user preference coverage.

In summary, this project not only met the expected system prototype development objectives, but it also gained valuable technical experience and problem-solving techniques during the actual implementation and testing process. As the system functions and performance are continuously improved, this project is expected to grow into a dynamic and intelligent personalized book recommendation platform that can run stably in the real world and continue to provide users with an efficient, accurate, and pleasant reading recommendation experience.

5.2. Limitations and Future Works

This project faces several limitations primarily stemming from technical constraints. Due to limited computational resources, we were unable to train the models using optimal parameters or on the full dataset. This restricted the model’s ability to learn comprehensive patterns from the data and affected the overall quality and consistency of recommendations. Additionally, the heavy load during data processing sometimes caused execution failures, preventing the recommender system from being repeatedly trained and fine-tuned under ideal conditions.

Another significant limitation is related to user interaction. Although the system implements collaborative filtering alongside login, signup, and database connection features at the project’s prototype, it currently does not support real-time user interaction through the interface. Even the users can locally save or rate books, but these interactions are not uploaded to the server or incorporated into the recommender model dynamically. As a result, collaborative filtering can only be tested using static user data from the original training dataset, and new users cannot fully benefit from personalized recommendations unless pre-existing in the dataset.

Future improvements can significantly enhance the system's functionality and impact. Implementing real-time data integration would allow user interactions, such as ratings and saved books, to immediately update the user profile and dynamically adjust recommendation outputs. To address the cold start problem for new users, a simple onboarding questionnaire or interest selection process during registration could collect initial preference data and generate personalized recommendations even for users without prior history. Moreover, upgrading computational resources, such as utilizing cloud servers or GPU-based environments, would enable training on the full dataset with optimal parameters, resulting in better model generalization and performance. An incremental learning approach could also be introduced, allowing the model to continuously learn from new interactions without the need for complete retraining. Enhancing the user interface to support seamless interaction and linking these actions directly to the backend would further enrich the user experience and improve recommendation quality. Finally, establishing a feedback loop where users can evaluate the relevance of recommendations would provide valuable data for ongoing model improvement and evaluation.

Reference & Source

Abdelrahman M. (2024, January 22). *Hypier-based Book recommendation system (end-to-end with python).* Medium. https://medium.com/@abdelrahman.m2922/book-recommendation-system-fa510e2d5a24

Aggarwal, C. C. (2016). *Recommender Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-319-29659-3

ashima96. (n.d.). *Book-recommendation-system/brs.ipynb at master · Ashima96/book-recommendation-system*. GitHub. https://github.com/ashima96/Book-Recommendation-System/blob/master/BRS.ipynb

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems,* 46, 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. ArXiv.org. https://doi.org/10.48550/arXiv.1810.04805

Giordano, V. (2025). How many books are in the world? (2025) - ISBNDB blog. *ISBNDB Blog.* https://isbndb.com/blog/how-many-books-are-in-the-world/

Gong, S., Ye, H., & Tan, H. (2009). Combining Memory-Based and Model-Based Collaborative Filtering in Recommender System. *2009 Pacific-Asia Conference on Circuits, Communications and Systems.* https://doi.org/10.1109/paccs.2009.66

Google colab. Medium. (n.d.). https://medium.com/google-colab

Google. (n.d.). *Colab paid services pricing*. Google. https://colab.research.google.com/signup

Google. (n.d.). Google colab. https://research.google.com/colaboratory/faq.html

Ilona, A. (2021). Comparison of TF-IDF model and BERT model for the classification of articles in an information portal for a case-based recommender system. *Opus.fhv.at.* https://doi.org/10.25924/opus-4104

Journal Of L A T E X Class, & Files. (2021). *Recent Developments in Recommender Systems: A Survey.* 14(8). https://arxiv.org/pdf/2306.12680

Jupyter. (n.d.). *Jupyter/notebook: Jupyter interactive notebook*. GitHub. https://github.com/jupyter/notebook

Jupyter-Notebook - features. Jupyter-Notebook - Features | Elest.io. (n.d.). https://elest.io/open-source/jupyter-notebook/resources/software-features

Kavitha, V. K., & Koteeswaran, S. (2023). Study on book recommendation system. *2023 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA),* 1–8. https://doi.org/10.1109/accthpa57160.2023.10083372

Licensing terms for project jupyter code. Licensing terms for Project Jupyter code - Project Jupyter Governance. (n.d.). https://jupyter.org/governance/projectlicense.html

LinkedIn. (2025). Linkedin.com. https://www.linkedin.com/pulse/15-realizing-analytical-triumphs-understanding-impact-sulagna-tah-6ozvc

Lops, P., de Gemmis, M., & Semeraro, G. (2010). Content-based Recommender Systems: State of the Art and Trends. *Recommender Systems Handbook,* 73–105. https://doi.org/10.1007/978-0-387-85820-3_3

Mathew, P., Kuriakose, B., & Hegde, V. (2016). Book Recommendation System through content based and collaborative filtering method. *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE).* https://doi.org/10.1109/sapience.2016.7684166

Mohammed Al-Ghuribi, S., Azman, S., & Noah, M. (n.d.). *A Comprehensive Overview of Recommender System and Sentiment Analysis.* https://arxiv.org/pdf/2109.08794

Newest "Jupyter-notebook" questions. Stack Overflow. (n.d.). https://stackoverflow.com/questions/tagged/jupyter-notebook

piyushsharma7757. (2025). *Book-Recommendation-System/book-recommendation-system.ipynb at main · piyushsharma7757/Book-Recommendation-System.* GitHub. https://github.com/piyushsharma7757/Book-Recommendation-System/blob/main/book-recommendation-system.ipynb

Priyanka Shahane. (2021, May 1). *A Survey on Book Recommendation Systems.* https://www.researchgate.net/publication/364699963_A_Survey_on_Book_Recommendation_Systems

Project jupyter documentation#. Project Jupyter Documentation - Jupyter Documentation 4.1.1 alpha documentation. (n.d.). https://docs.jupyter.org/en/latest/

recommenders/examples/02_model_collaborative_filtering/surprise_svd_deep_dive.ipynb at main · recommenders-team/recommenders. (n.d.). GitHub. https://github.com/recommenders-team/recommenders/blob/main/examples/02_model_collaborative_filtering/surprise_svd_deep_dive.ipynb

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (Eds.). (2011). Recommender Systems Handbook. Springer US. https://doi.org/10.1007/978-0-387-85820-3

Rutherford, L., Singleton, A., Reddan, B., Johanson, K., & Dezuanni, M. (2024). *Discovering a Good Read: Exploring Book Discovery and Reading for Pleasure Among Australian Teens.* Geelong: Deakin University.

https://teenreading.net/wp-content/uploads/sites/250/2024/03/Discovering-a-Good-Read-Survey-Report_FINAL.pdf

singh, S. (2021, June 9). *BOOK RECOMMENDATION SYSTEM*. Medium. https://shwetasingh8597.medium.com/book-recommendation-system-1679ed4f9f6d

Welcome to Python.org. Python.org. (n.d.). https://www.python.org/

Welcome to Python.org. Python.org. (n.d.-a). https://www.python.org/psf-landing/

Zhang, T., Niu, X., & Promann, M. (2017). A*ssessing the user experience of E-Books in academic libraries.* Zhang | College & Research Libraries. https://crl.acrl.org/index.php/crl/article/view/16713/18220