Book Recommendation Systems

E. H. GOMBRICH A LITTLE HISTORY of THE WORLD
SAM ANDERSON BOOM TOWN
NORMAL PEOPLE A NOVEL SALLY ROONEY
DRIVE YOUR PLOW OVER THE BONES OF THE DEAD OLGA TOKARCZUK
My Private Property Mary Ruefle
Growing Without Schooling
HOW TO BE IDLE TOM HODGKINSON
LIFE AMONG THE SAVAGES SHIRLEY JACKSON
The Annotated Alice LEWIS CARROLL
HOW BUILDINGS LEARN What happens after they're built STEWART BRAND
IVAN ILLICH Tools for Conviviality
A Love Letter to Texas Women Sarah Bird
KENKŌ AND CHŌMEI Essays in Idleness and Hōjōki
STEFAN ZWEIG The World of Yesterday
W. H. AUDEN The Dyer's Hand
THE GARDENER AND THE CARPENTER ALISON GOPNIK
EVERYTHING SHE TOUCHED THE LIFE OF RUTH ASAWA
Caste The Origins of Our Discontents Isabel Wilkerson
RAY BRADBURY THE ILLUSTRATED MAN
THE COMEDY OF SURVIVAL Literary Ecology and a Play Ethic JOSEPH W. MEEKER

Marie-Noël Lepage
November 8, 2023

Problem Statement

Over the past few decades, thanks to the emergence of platforms such as YouTube, Amazon, Netflix and many other web services, recommendation systems have become an increasingly integral part of our daily online experiences. From guiding customers through e-commerce by suggesting items that might interest them, to tailoring online advertising to user preferences, recommender systems have become an indispensable part of our digital journeys.

These algorithms designed to provide users with suggestions that match that with needs and preferences, in a variety of business sectors. These suggestions can range from recommending movies to watch, articles to read, products to buy, or other items, depending on the specific domain. Recommendation systems are of great importance in certain sectors, as their effectiveness can help generate substantial revenues and gain a definite competitive edge.

In this project, we delve into the realm of recommendation systems, exploring various techniques aimed at assisting customers by providing book suggestions tailored to their interests.

Data Sources

The dataset[1] used for this project was collected by Cai-Nicolas Ziegler in 4-week crawl (August / September 2004) from the Book-Crossing community. Contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books. The rating explicit is an exact number given by a user to a book. The rating implicit doesn't directly reflect the interest of the user but acts as a proxy for a user's interest. Example: click on the link, search for the book, etc.

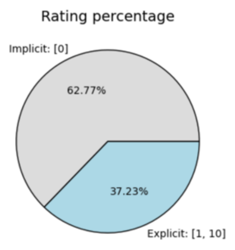
Data	Description
👫	Contains the users. Note that user IDs have been anonymized and map to integers. Demographic data is provided (Location, Age) if available.
📖	Books are identified by their ISBN. Some content information is provided (title, author, year of publication, publisher, image), obtained from Amazon Web Services.
★★☆	Contains the book rating information. Ratings are either explicit (expressed on a scale from 1-10), or implicit (expressed by 0).

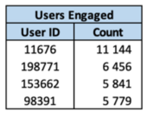
[1] https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset

Data Cleaning and Exploration

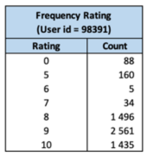
Ratings:

- The ratings dataset should only contain books that exist in our book's dataset. 118,633 (10,32%) ratings were deleted for this reason.
- Implicit & Explicit ratings are 1 per user / book. I don't have a definition for implicit ratings, but maybe it's buying / reading a book without given a rating.

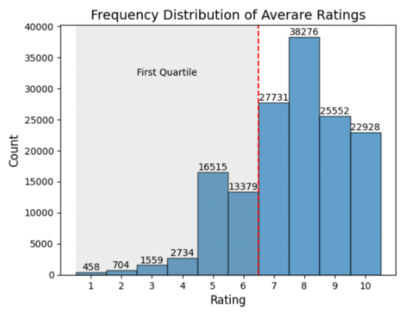




Users Engaged	
User ID	Count
11676	11 144
198771	6 456
153662	5 841
98391	5 779



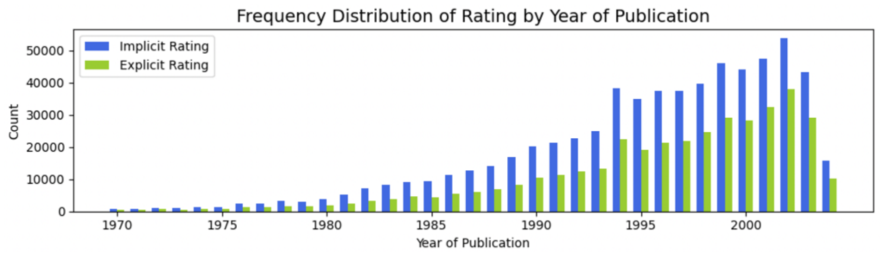
Frequency Rating (User id = 98391)	
Rating	Count
0	88
5	160
6	5
7	34
8	1 496
9	2 561
10	1 435

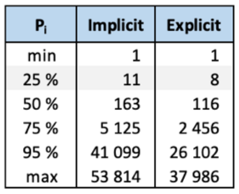


37% of users have rated a book, and the first quartile of the average rating is 6.5. Some users are highly engaged (passionate readers, account with several users, bots?).

Books:

- 2 books have no author and have been filled by unknown.
- 2 books have no publisher and have been filled by internet search.
- 3 books have shifted columns caused by a comma in the title and have been corrected.
- Some publication years are filled by 0 or with a date after 2004 (the year for which the database was created). These data have been replaced by the average.





P_i	Implicit	Explicit
min	1	1
25 %	11	8
50 %	163	116
75 %	5 125	2 456
95 %	41 099	26 102
max	53 814	37 986

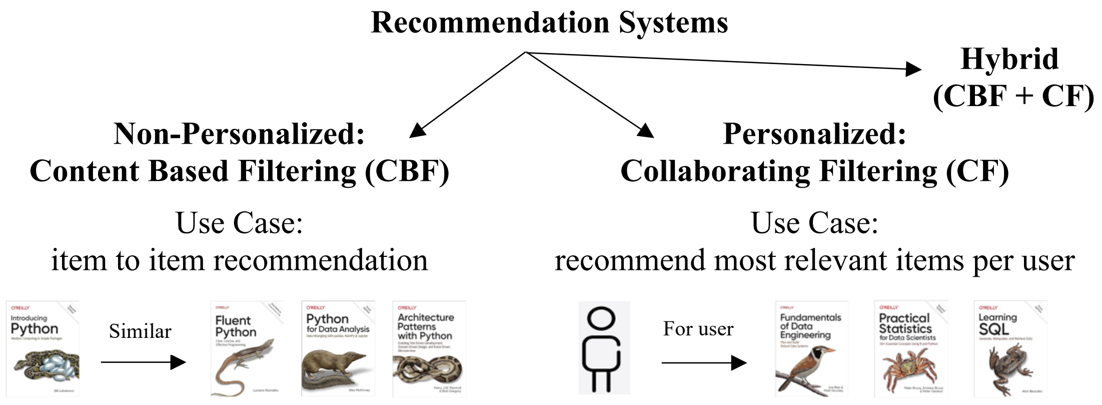
The older a book is according to its year of publication, the less the user will interact with it. 25% of publication years contain only 11 (implicit) / 8 (explicit) books or less.

Users:

- The age of some users is missing or inconsistent (< 8 or > 100 years). I have replaced this data with the average.
- I selected the country from the location. To clean up the name mistake and standardize the format, I used the dataprep[2] library. Missing values have been filled by unknown.

[2] https://docs.dataprep.ai/index.html

Technique Used


Recommendation Systems
Hybrid (CBF + CF)
Non-Personalized: Content Based Filtering (CBF)
Personalized: Collaborating Filtering (CF)
Use Case: item to item recommendation
Use Case: recommend most relevant items per user
Similar
For user

- Non-Personalized: Suggests general products for users.
- Personalized: Suggests products to a user based on their previous interaction history.

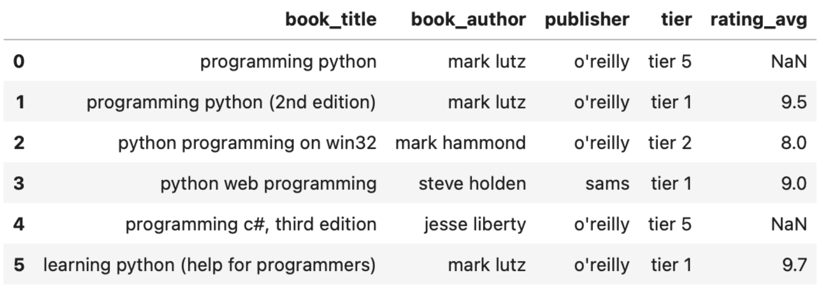
Content Based Filtering

This system analyzes the content of items and makes recommendations based on the similarity between item content and user preferences. It can be both personalized (building user and items profile on user rated content) and non-personalized (content description analysis only) depending on how it's implemented. In this project, we ignore user engagement and rely solely on the similarity books content.

Methodology:

- Feature engineering: Calculate the average explicit ratings for each book. Transform into tier (tier 1:]8, 10], tier 2:]6, 8], tier 3:]4, 6], tier 4:]1, 4], tier 5: NaN).
- Preprocessing: With lower case, deletion of leading & trailing whitespaces, merging of attributes (title, publisher, and tier) and deletion of stop words.
- Feature extraction with TF-IDF[3]: Statistical approach used to identify terms and reduce the impact of common and less informative words in textual data. It calculates how relevant of a word to a document in a corpus.
- Estimate similarity with nearest neighbors[4] (Euclidian distance).

Top 5: Most Similar Book of Programming Python



	book_title	book_author	publisher	tier	rating_avg
0	programming python	mark lutz	o'reilly	tier 5	NaN
1	programming python (2nd edition)	mark lutz	o'reilly	tier 1	9.5
2	python programming on win32	mark hammond	o'reilly	tier 2	8.0
3	python web programming	steve holden	sams	tier 1	9.0
4	programming c#, third edition	jesse liberty	o'reilly	tier 5	NaN
5	learning python (help for programmers)	mark lutz	o'reilly	tier 1	9.7

The first line [0] represents the book selected by the user. The next five lines [1, 5] represent the recommendation.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[4] https://scikit-learn.org/stable/modules/neighbors.html

Pros	Cons
Useful for cold start problem: when the data available for users or items is limited.	Non-Personalized: don't consider a user's past interactions and preference.
More interpretable: can be easy to explain why a particular item was recommended.	Lack of diversity: recommendation of very similar items (no content discovery).
No data sparsity problem: common in collaborative filtering where there may be few user-item interactions.	Scalability: can become computationally expensive for system with many items or high dimensional feature.

Collaborative Filtering

To provide users with personalized recommendations, I used the collaborative filtering technique. It relies on the idea that people who have similar preferences in the past are likely to have similar preferences in the future. This system is based on user-item interactions, such as ratings, purchases, or views. Like CBF. It can be personalized (in this project) and non-personalized depending how it's implemented.

Methodology:

- I use two libraries: Implicit[5] with ALS / BPR algorithms (on implicit or explicit rating) and LightFM[6] (on implicit and explicit rating) to find the best model. ALS is a matrix factorization technique that aims to factorize the user-item interaction matrix into two lower-dimensional matrices representing user and item factors. BPR focuses on optimizing the ranking order of items for each user. WARP is an extension of BPR, whit a more aggressive approach to improving the ranking accuracy.
- I create a user-item interaction matrix in which the rows represent users, the columns represent items, and the values represent user interactions between the users and items.
- I use BM25 weight to reduce the weight given to popular item with the implicit library.
- To evaluate the model, I split the data into 80% for training and 20% for testing. To find the best hyper parameters for the selected model, I proceed by iteration. In addition, to obtained leverage on the score, I try by retrieve the first quartile of the year or the average rating in the implicit library. Based on the evaluation, it appears that the best model is that of the explicit rating using ALS.

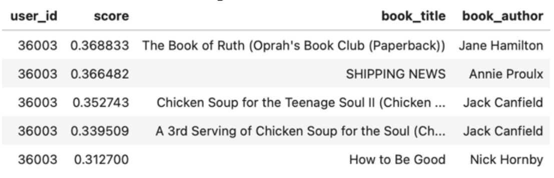
Implicit: BPR	Explicit: ALS	Implicit & Explicit: WARP
BM25: K1:1, B: 0.15 **BPR:** Factors: 1024 Iteration: 200 Learning Rate: 0.3 Regularization: 0.005 **Evaluation:** MAP@5: 1.30	**BM25:** K1: 1, B: 0.05 **ALS:** Factors: 1024, Iterations: 10, Regularization: 0.5 Without Subclass of avg ratings **Evaluation:** MAP@5: 1.64, Precision@5: 3.38, NDCG@5: 2.24, AUC@5: 51.08	**LightFM:** Loss: warp No components: 50 Learning Rate: 0.01 **Fit function:** Epochs: 10 **Evaluation:** Precision@5 Train / Test: 1.18 / 0.69 AUC Train / Test: 86.84 / 76.04

[5] https://benfred.github.io/implicit/index.html
[6] https://making.lyst.com/lightfm/docs/index.html

- Finally, we train the model on the entire dataset to capture all patterns and display the top 5 recommendations. The ALS model shows recommendation which the user has not already interacted with the book and excludes books in the 25% percentile for the average rating.

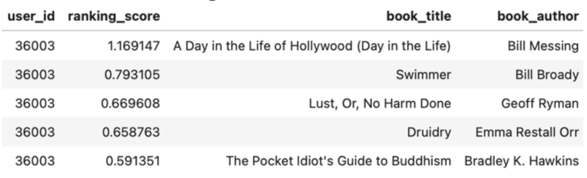
Top 5: Recommendation for user_id = 36003

Implicit ALS model



user_id	score	book_title	book_author
36003	0.368833	The Book of Ruth (Oprah's Book Club (Paperback))	Jane Hamilton
36003	0.366482	SHIPPING NEWS	Annie Proulx
36003	0.352743	Chicken Soup for the Teenage Soul II (Chicken ...	Jack Canfield
36003	0.339509	A 3rd Serving of Chicken Soup for the Soul (Ch...	Jack Canfield
36003	0.312700	How to Be Good	Nick Hornby

LightFM WARP model



user_id	ranking_score	book_title	book_author
36003	1.169147	A Day in the Life of Hollywood (Day in the Life)	Bill Messing
36003	0.793105	Swimmer	Bill Broady
36003	0.669608	Lust, Or, No Harm Done	Geoff Ryman
36003	0.658763	Druidry	Emma Restall Orr
36003	0.591351	The Pocket Idiot's Guide to Buddhism	Bradley K. Hawkins

Pros	Cons
Personalized: based on user interactions and preferences.	Cold start problem: when the data available for the users / items is limited.
Scalability: not require detailed item attributes. Can scale to large datasets and handle a wide range of users and items.	Data sparsity: users may not have interacted with a significant portion of the items.
Diversity: especially when users have diverse preferences and interactions.	Lack of transparency: may be a drawback for some users.

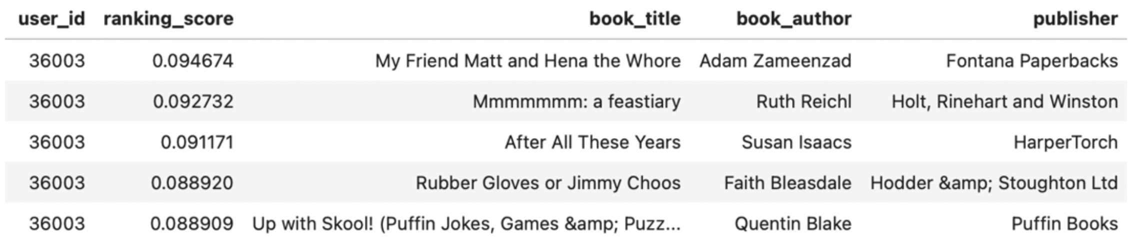
Hybrid

The hybrid model combines several recommendation techniques. This method can be implemented in several ways (weighted, switching, stacking). It some cases, it could take advantage of the strengths of different approaches, such as cold start and the sparsity problem.

Methodology:

- I used the LightFM library to build the hybrid model, which combines the CBF and CF models using weighting technique. The methodology is very similar to that of collaborative filtering seen previously, but the difference is that we need to implement a user or / and item features included in the model fitting process.
- For item features, I've tried with the publisher. It's a large feature that contains a list of 16,726 unique categories. Unfortunately, I don't have the genre of the books in my dataset, but it might be possible to increase the result comparatively to the publisher.
- For the user feature, I've tried with the country and an age class I've created with my dataset: children: [8, 14], teenager:]14, 24], adult:]24, 64], senior:]64, 100].
- With the same hyper parameters in the collaborative filtering model, my best result is when I use only the user features for age class. The precision is very close to that of my collaborative filtering but the auc score decrease slightly (train / test precision@5: 1.19 / 0.73, train / test auc: 83.34 / 75.10. This shows that this hybrid system is not necessarily outperform than the pure collaborative system.

Top 5: Recommendation for user_id = 36003



user_id	ranking_score	book_title	book_author	publisher
36003	0.094674	My Friend Matt and Hena the Whore	Adam Zameenzad	Fontana Paperbacks
36003	0.092732	Mmmmmmm: a feastiary	Ruth Reichl	Holt, Rinehart and Winston
36003	0.091171	After All These Years	Susan Isaacs	HarperTorch
36003	0.088920	Rubber Gloves or Jimmy Choos	Faith Bleasdale	Hodder &amp; Stoughton Ltd
36003	0.088909	Up with Skool! (Puffin Jokes, Games &amp; Puzz...	Quentin Blake	Puffin Books

Conclusion

Including recommendations in systems is a worthwhile investment. Recommendation systems not only improve the user experience and engagement but also generate more revenue for businesses. In this project, I developed different taxonomies of recommender systems. The choice of which depends on the business cases and the data available.

By removing outliers and dealing with items or users with few interactions, you can improve the quality and robustness of your recommender system. The specific techniques and thresholds depend on your data and the characteristics of your business rules. In my project, I don't do this because if I remove books and users with less than one interaction, I remove more than 60% of my data.

When a recommendation system is put into production, a good practice is to use A/B testing to measure the effectiveness of recommendations generated. During an A/B test, only a sample of users are exposed to the service of each personalization solution. The goal is to determine if the measured difference in their reactions would be valid for all users. You can select a variety of key metrics for this, for example: gross merchandising value (GMV), click-through rate (CTR), watch time through recommendation.[7]

Note that, for the Content-Based Filtering, we can use embedding (such as sequence transformer[8] and universal sentence encoder[9]) instead to TD-IDF. Also, to find similarity, we can use other technique such as the cosine similarity[10] and approximate nearest neighbors (Annoy[11], NMSLIB[12] and Faiss[13]). For the Collaborative Filtering, we can use other algorithms such as the Nearest Neighbors and the Neural Collaborative Filtering[14].

[7] https://www.yusp.com/blog-posts/behind-the-scenes-of-ab-testing-recommendation-systems/#:~:text=During%20an%20A%2FB%20test,be%20valid%20for%20all%20users.
[8] https://www.sbert.net/docs/pretrained_models.html
[9] https://tfhub.dev/google/universal-sentence-encoder-large/5
[10] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html
[11] https://github.com/spotify/annoy
[12] https://github.com/nmslib/nmslib
[13] https://github.com/facebookresearch/faiss
[14] https://arxiv.org/abs/1708.05031