Book Recommender Systems

Nurbek Bektursyn
Asset Kabdula

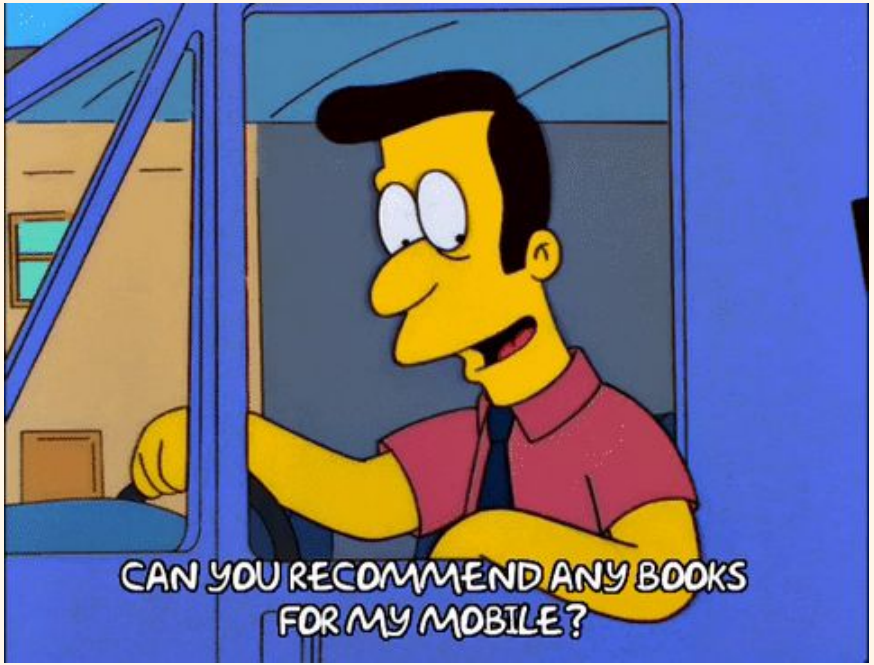
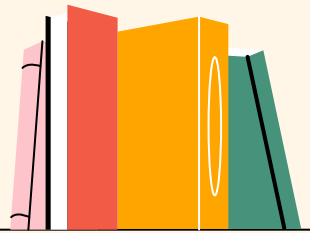
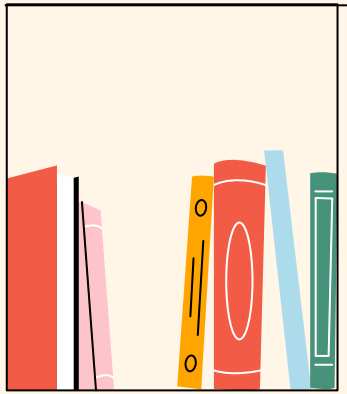
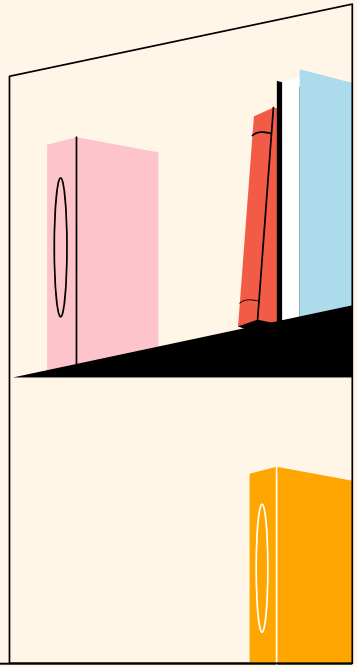


Table of contents





Data



book_id	original_title	authors	average_rating
1	The Hunger Games	Suzanne Collins	4.34
2	Harry Potter and the Philosopher's Stone	J.K. Rowling, Mary GrandPré	4.44
3	Twilight	Stephenie Meyer	3.57
4	To Kill a Mockingbird	Harper Lee	4.25
5	The Great Gatsby	F. Scott Fitzgerald	3.89
6	The Fault in Our Stars	John Green	4.26
7	The Hobbit or There and Back Again	J.R.R. Tolkien	4.25
8	The Catcher in the Rye	J.D. Salinger	3.79
9	Angels & Demons	Dan Brown	3.85
10	Pride and Prejudice	Jane Austen	4.24

user_id	book_id	rating
1	258	5
2	4081	4
2	260	5
2	9296	5
2	2318	3
2	26	4
2	315	3
2	33	4
2	301	5
2	2686	5

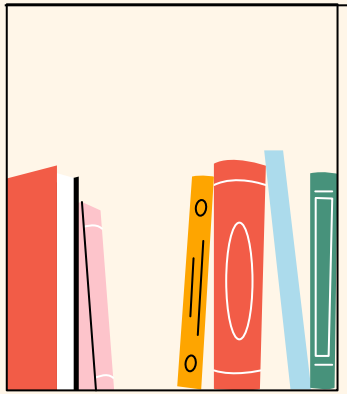
Source: goodbooks-10k (Ten thousand books, one million ratings)

Size: *books.csv* - (9415, 4);
ratings.csv - (5976479, 3)

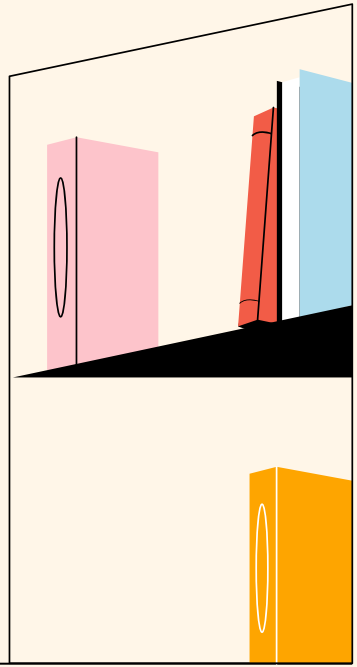
Preprocessing:

- Remove missing values
- Exclude less active users (< 95th percentile)
- Merge two datasets on “book_id”

Final dataset:
merged_data - (449423, 6)

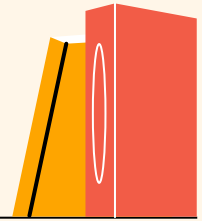


Evaluation



Splitting Dataset into train and test

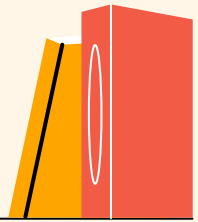
```
train, test = train_test_split(merged_data,
                               stratify=merged_data['user_id'],
                               test_size=0.50,
                               random_state=42)
```



Top-N accuracy metric

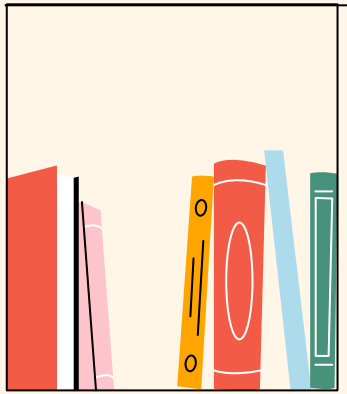
- For each user
  - For each item the user has interacted in test set
    - Sample 100 other items the user has never interacted.
    - Recommender model produces a ranked list of recommended items, from a set composed one interacted item and the 100 non-interacted ("non-relevant!) items
    - Compute the Top-N accuracy metrics for this user and interacted item from the recommendations ranked list
- Aggregate the global Top-N accuracy metrics

True label	Top-3 Predicted labels	Correct
Cat	**Cat**, Lion, Dog	✓
Dog	Giraffe, Lion, Cat	×
Lion	Cat, **Lion**, Dog	✓
Giraffe	**Giraffe**, Dog, Cat	✓
Dolphin	**Dolphin**, Cat, Giraffe	✓

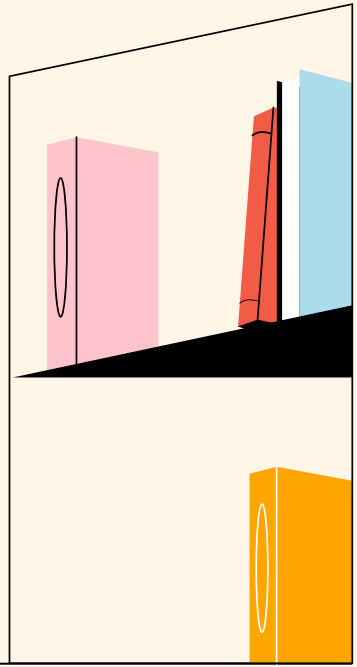


Models

- Popularity
- Content-based filtering
- Collaborative filtering



Popularity filtering



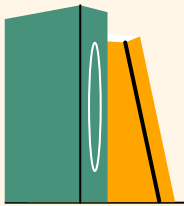
book_id	original_title	authors	average_rating
3628	The Complete Calvin and Hobbes	Bill Watterson	4.82
862	Words of Radiance	Brandon Sanderson	4.77
8854	Mark of the Lion Trilogy	Francine Rivers	4.76
4483	It's a Magical World: A Calvin and Hobbes Collection	Bill Watterson	4.75
6361	There's Treasure Everywhere: A Calvin and Hobbes Collection	Bill Watterson	4.74
422	Complete Harry Potter Boxed Set	J.K. Rowling	4.74
6920	The Indispensable Calvin and Hobbes: A Calvin and Hobbes Treasury	Bill Watterson	4.73
3753	Harry Potter Collection (Harry Potter, #1-6)	J.K. Rowling	4.73
6590	The Authoritative Calvin and Hobbes	Bill Watterson	4.73
1308	A Court of Mist and Fury	Sarah J. Maas	4.72

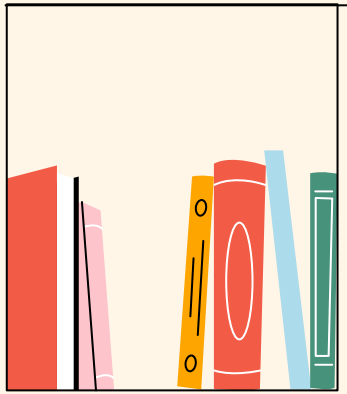
Global metrics:

'modelName': 'Popularity', simply recommends to a user **the most popular books** that the user *has not previously read.*

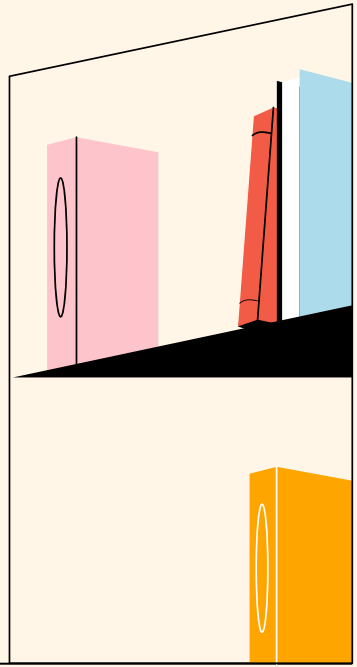
recall@5: 0.058728505820783934, or ***5.8%*** success rate **with top 5 picks**

recall@10: 0.1001726654562284, or 10% success rate **with top 10 picks**





Content-based filtering



Preprocessing & Building user profiles

- Initialize with English resources: Use English stopwords and lemmatization.
- Preprocesses text input: Tokenizes, lowers case, and remove stopwords and non-alpha characters.
- Lemmatize tokens: Convert words to their base or dictionary form.
- TF-IDF Vectorization: Apply on preprocessed book titles and authors.
- Configure Vectorizer: Consider 1-2 word combinations, filter by document frequency, limit to 5000 features.
- Generate TF-IDF Matrix: Represent books as numerical vectors based on title and author text.

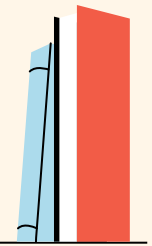
	token	relevance
0	smith	0.291435
1	william	0.269649
2	ann	0.247159
3	jane	0.226299
4	anne	0.210372
5	john	0.203345
6	brook	0.196299
7	white	0.184692
8	david	0.173840
9	james	0.166999
10	sara	0.165289
11	little	0.164414
12	robert	0.143590
13	margaret	0.140877
14	secret	0.139598
15	george	0.138449
16	agatha christie	0.120226
17	christie	0.120226
18	agatha	0.119810
19	anderson	0.110193

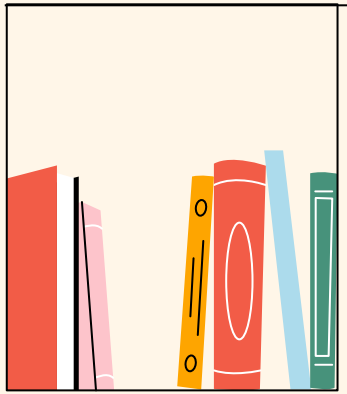
Evaluation of content-based filtering

	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	_person_id
792	7	16	99	0.070707	0.161616	12381
1851	19	36	99	0.191919	0.363636	30944
2687	23	31	98	0.234694	0.316327	52036
1263	14	21	98	0.142857	0.214286	19729
824	8	14	98	0.081633	0.142857	12874
2456	13	20	98	0.132653	0.204082	45554
2147	7	12	97	0.072165	0.123711	37834
595	5	15	97	0.051546	0.154639	9668
600	23	34	97	0.237113	0.350515	9731
1023	16	22	97	0.164948	0.226804	15604

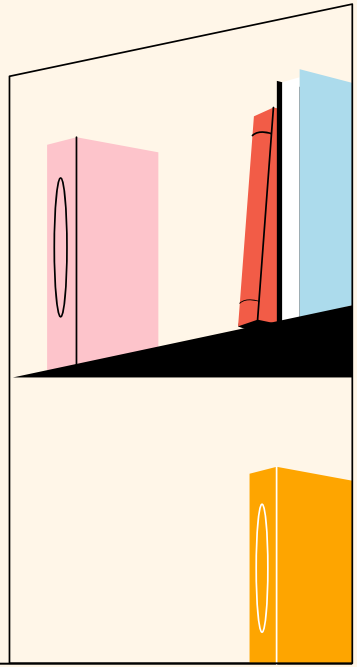
recall@5: 0.208, or ***20.8%*** success rate **with top 5 picks**

recall@10: 0.282, or ***28.2%*** success rate **with top 10 picks**

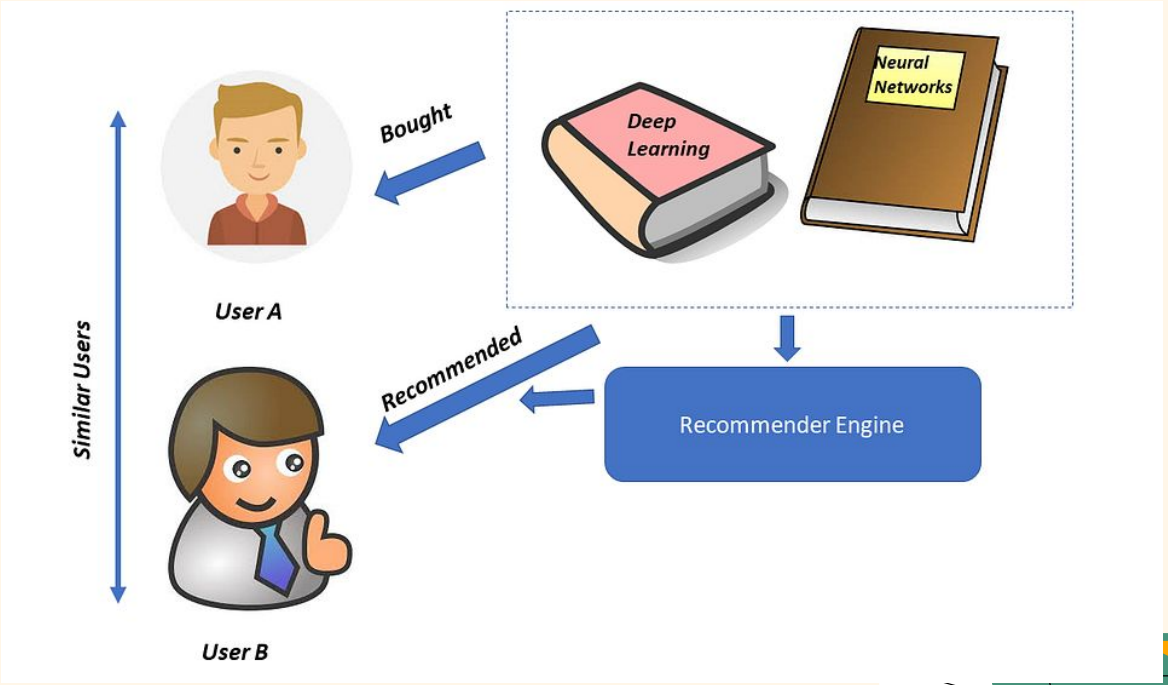




Collaborative filtering



Collaborative filtering

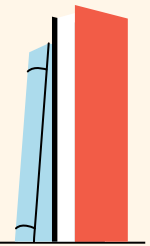


Creating a sparse pivot table

book_id	1	2	3	4	5	6	7	8	9	10	...	9990	9991	9992	9993	9994	9995	9997	9998	9999	10000
user_id																					
35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75	4.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
143	5.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
145	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
173	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
178	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
202	0.0	0.0	4.0	0.0	0.0	0.0	4.0	4.0	3.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
215	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
230	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
247	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

10 rows × 8578 columns

Users in rows and books in columns



Applying SVD

SVD(Singular value decomposition) compresses user-book matrix into a low-dimensional representation in terms of latent factors. Instead of having a high dimensional sparse matrix we will be dealing with a much more smaller matrix in lower-dimensional space.

```
#The number of factors to factor the user-item matrix.
NUMBER_OF_FACTORS_MF = 15
#Performs matrix factorization of the original user item matrix
#U, sigma, Vt = svds(users_items_pivot_matrix, k = NUMBER_OF_FACTORS_MF)
U, sigma, Vt = svds(users_items_pivot_sparse_matrix, k = NUMBER_OF_FACTORS_MF)   #Partial singular value decomposition of a sparse matrix.
                                                #Compute the largest or smallest k singular values and corresponding singular vectors of a sparse matrix
```

[1] Python

```
U.shape    #
```

[2] Python

```
(2729, 15)
```

```
Vt.shape
```

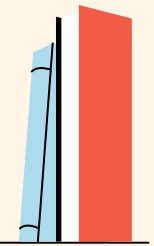
[3] Python

```
(15, 8578)
```

```
sigma = np.diag(sigma)
sigma.shape
```

[4] Python

```
(15, 15)
```



Reconstructed matrix

The values in the matrix are the likelihood of interaction between a user and a book

book_id	35	75	143	145	173	178	202	215	230	247	...	53145	53165	53173	53245	53279
1	0.431804	0.766716	0.724470	0.545489	0.398700	0.676712	0.401317	0.350642	0.296738	0.355488	...	0.831927	0.536969	0.430559	0.325806	0.421697
2	0.386021	0.546534	0.501913	0.324217	0.631422	0.419030	0.648636	0.510993	0.402641	0.537713	...	0.692195	0.520721	0.393065	0.392839	0.361336
3	0.422541	0.460007	0.482912	0.397726	0.465719	0.396806	0.447598	0.430792	0.389554	0.378388	...	0.562048	0.594663	0.425341	0.407137	0.395120
4	0.405104	0.418965	0.314730	0.496417	0.466162	0.380388	0.646625	0.508242	0.591308	0.625269	...	0.454597	0.565527	0.510040	0.461197	0.369145
5	0.420733	0.492274	0.428842	0.511857	0.503402	0.435743	0.515260	0.488716	0.502501	0.503830	...	0.472997	0.477509	0.408667	0.404624	0.443290
6	0.406801	0.463337	0.494717	0.383509	0.581138	0.433386	0.403394	0.381576	0.483074	0.371745	...	0.414066	0.586031	0.398351	0.449820	0.455584
7	0.433952	0.462368	0.459178	0.362965	0.493613	0.408540	0.558327	0.580769	0.433267	0.433545	...	0.534030	0.435896	0.386888	0.391333	0.398763
8	0.398902	0.461098	0.412662	0.489492	0.493222	0.408051	0.541654	0.490885	0.505964	0.527479	...	0.444424	0.459403	0.431779	0.429263	0.411006
9	0.420079	0.460836	0.485164	0.419589	0.463154	0.386911	0.497130	0.466260	0.422687	0.405378	...	0.485287	0.481175	0.470750	0.426492	0.389085
10	0.470973	0.451027	0.437818	0.399109	0.527748	0.397690	0.447502	0.501909	0.480912	0.457773	...	0.576058	0.502093	0.357791	0.410045	0.448808

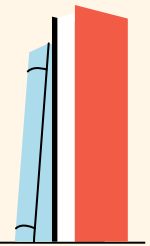
10 rows × 2729 columns

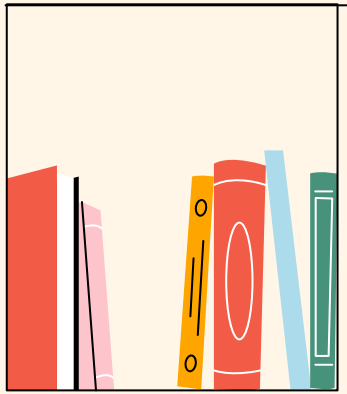
Evaluation of collaborative filtering

	hits@5_count	hits@10_count	interacted_count	recall@5	recall@10	_person_id
792	79	90	99	0.797980	0.909091	12381
1851	85	92	99	0.858586	0.929293	30944
2687	80	86	98	0.816327	0.877551	52036
1263	71	85	98	0.724490	0.867347	19729
824	85	90	98	0.867347	0.918367	12874
2456	94	97	98	0.959184	0.989796	45554
2147	63	75	97	0.649485	0.773196	37834
595	52	65	97	0.536082	0.670103	9668
600	80	93	97	0.824742	0.958763	9731
1023	75	83	97	0.773196	0.855670	15604

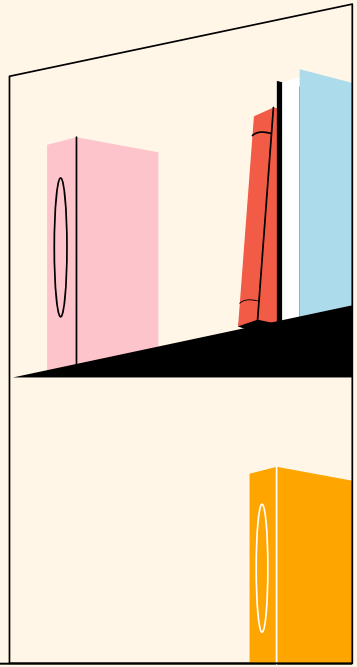
recall@5: 0.6285, or ***62.9%*** success rate **with top 5 picks**

recall@10: 0.7459, or ***74.6%*** success rate **with top 10 picks**



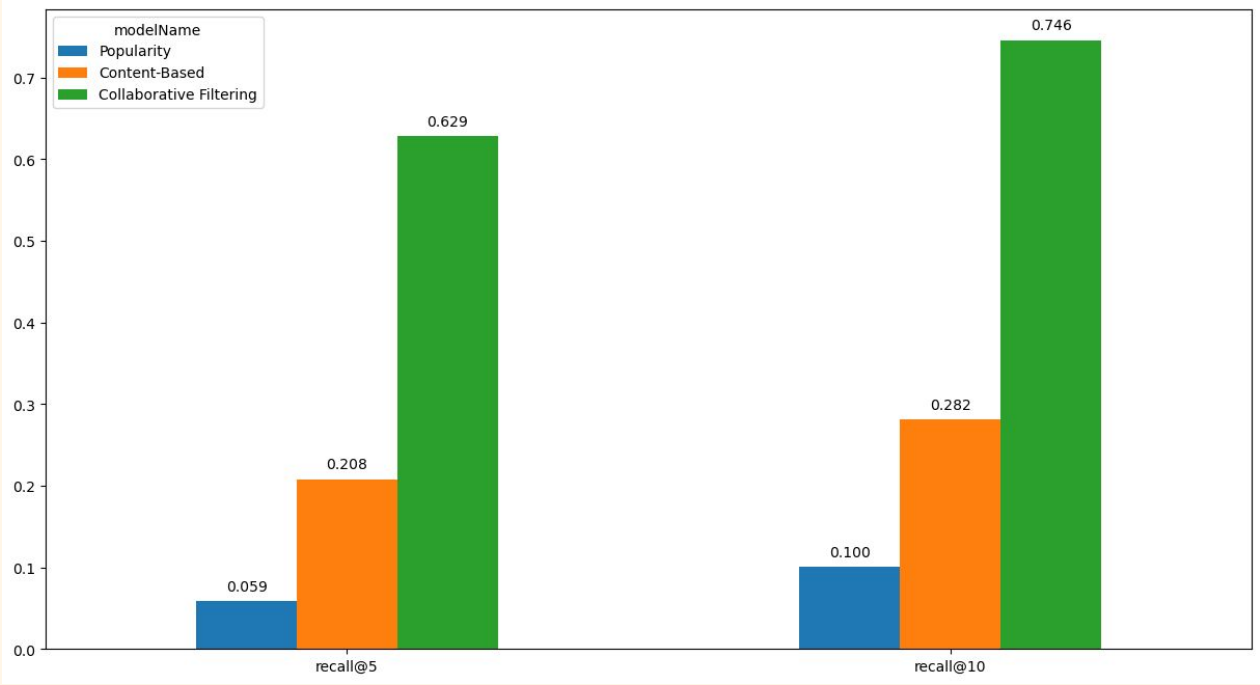


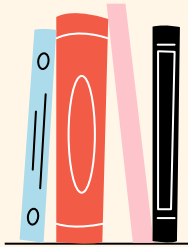
Comparison



Comparative analysis of the 3 models

Comparison of popularity, content-based and collaborative filtering models using Top-N accuracy metric





Thanks!

Review questions:

1. What is a recommender system, and why is it relevant in today's digital world?
2. Can you describe the distinction between collaborative and content-based filtering?
3. What role does popularity-based filtering play in recommender systems?
4. Describe how content-based filtering recommends items to users.
5. Describe how collaborative filtering recommends items to users.
6. What are Top-N accuracy metrics, and why are they important in evaluating recommender systems?
7. What is Spearman rank correlation coefficient? How does it work in context of book recommendation?