Goodreads Recommendation Engine

Palakh Gupta, Akankshi Mody, Catherine Yu Miao, Alisha Fernandes, Thiru Vinayagam

Advantages of a Recommendation Engine

Customer
Retention

Enhance shopping

experience

Shorter

conversion time

Drive traffic and deliver

relevant content

goodreads

Meet your next favorite book.

Dataset Description

Dimension: **847,000+** rows x **39** cols

Rows: Each row represents a rating by a goodreads user for a certain book

Cols: **Info. about books**

- isbn, title, edition, description, format, is_ebook, authors, languages, publishers, publication dates

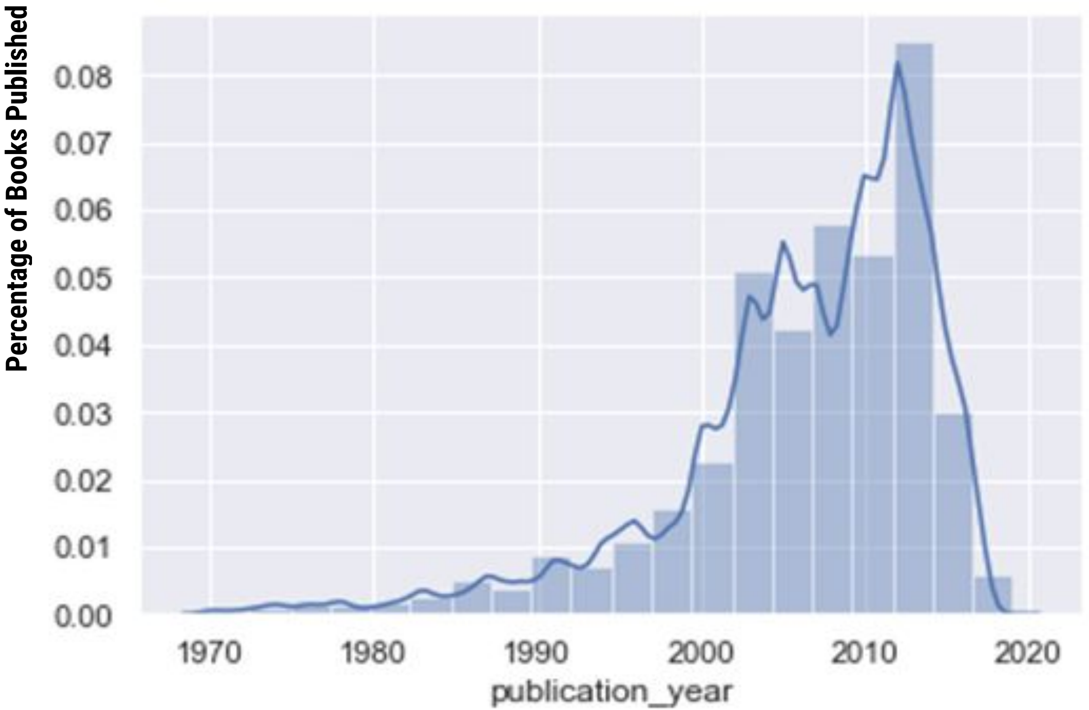
Info. about ratings

- user_id, is_read, rating score

1

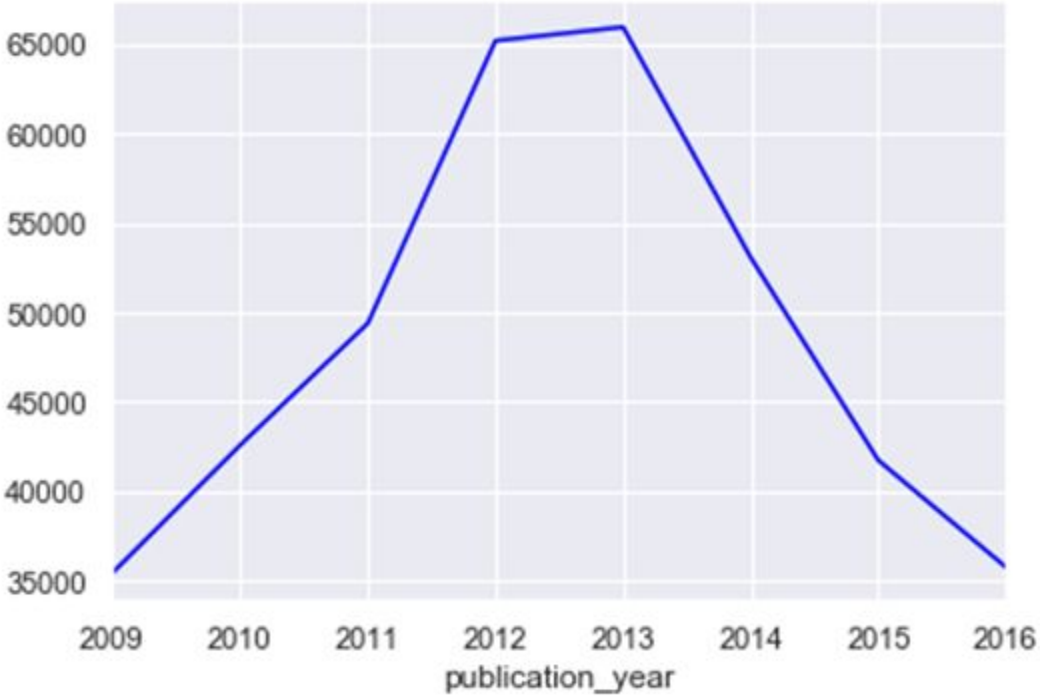
Exploratory Analysis & Visualizations

Distribution of Publication Year

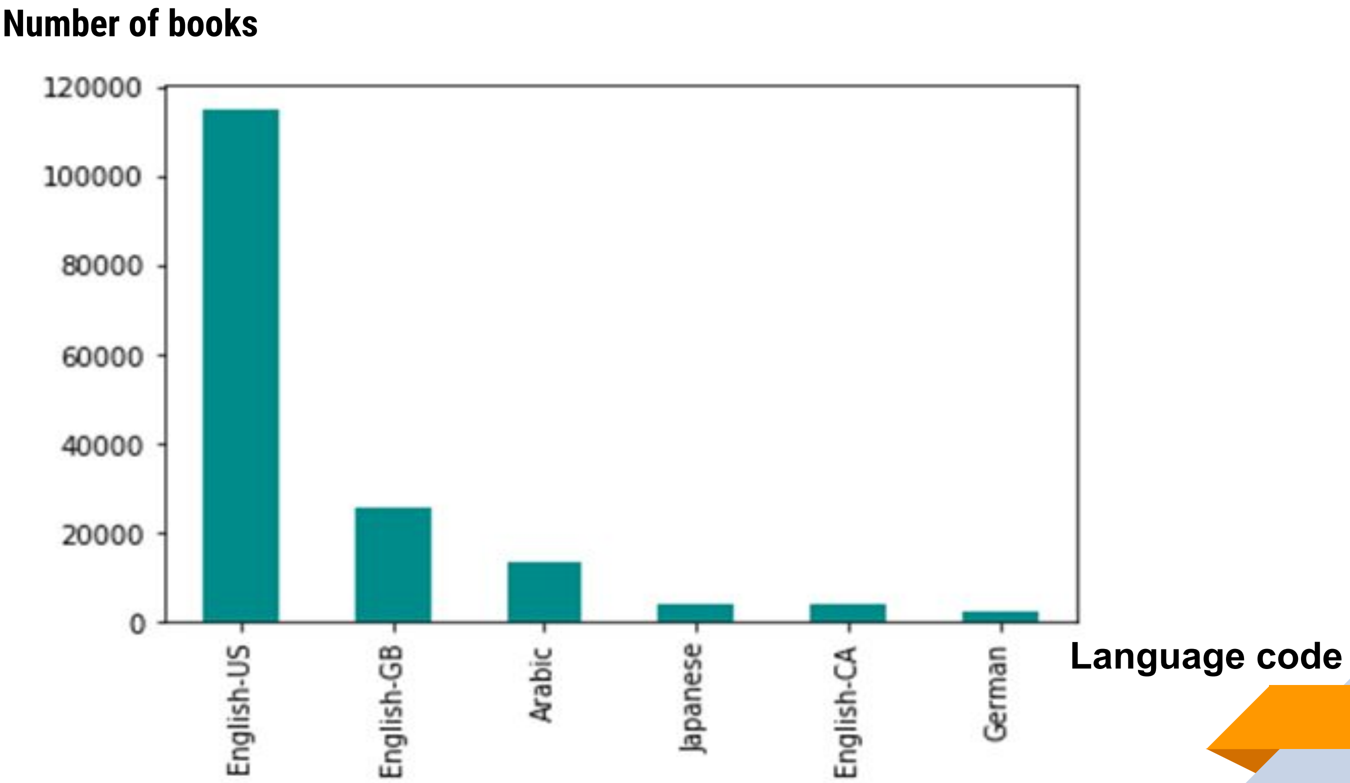


Number of Books Rated v.s. Publication Year

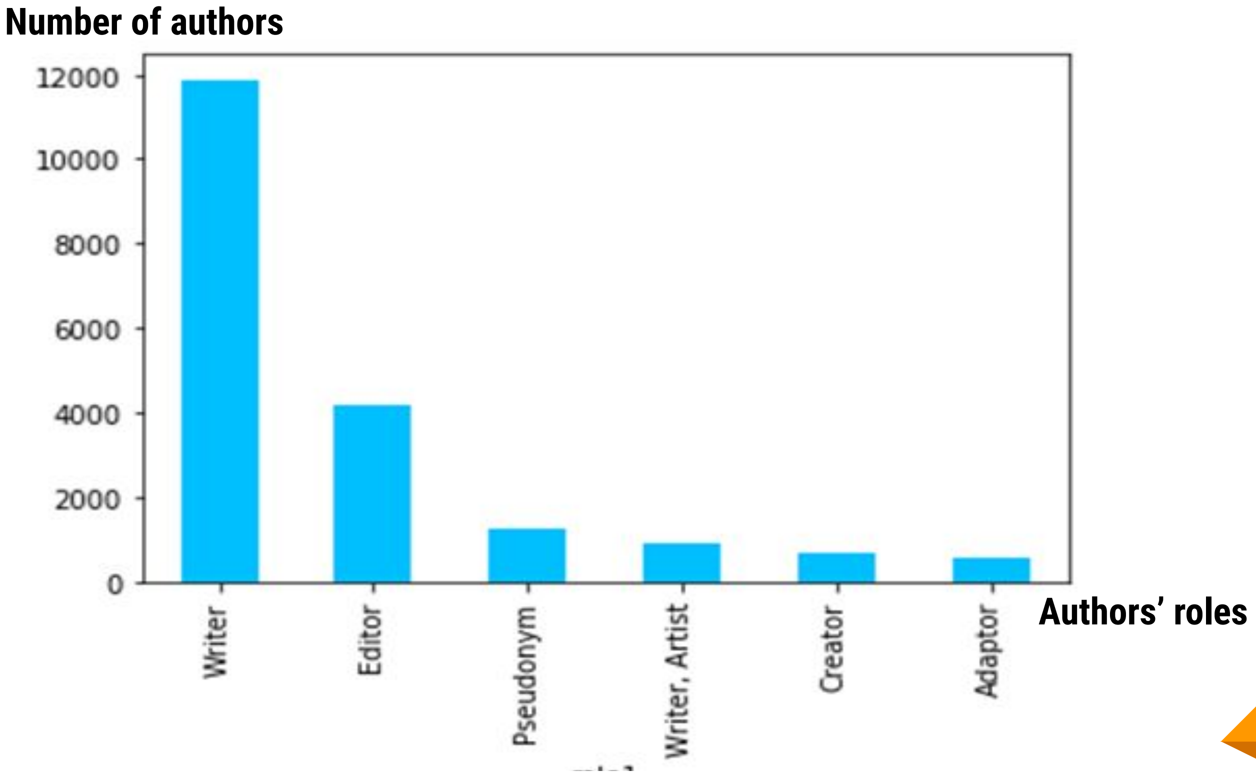
Number of books rated



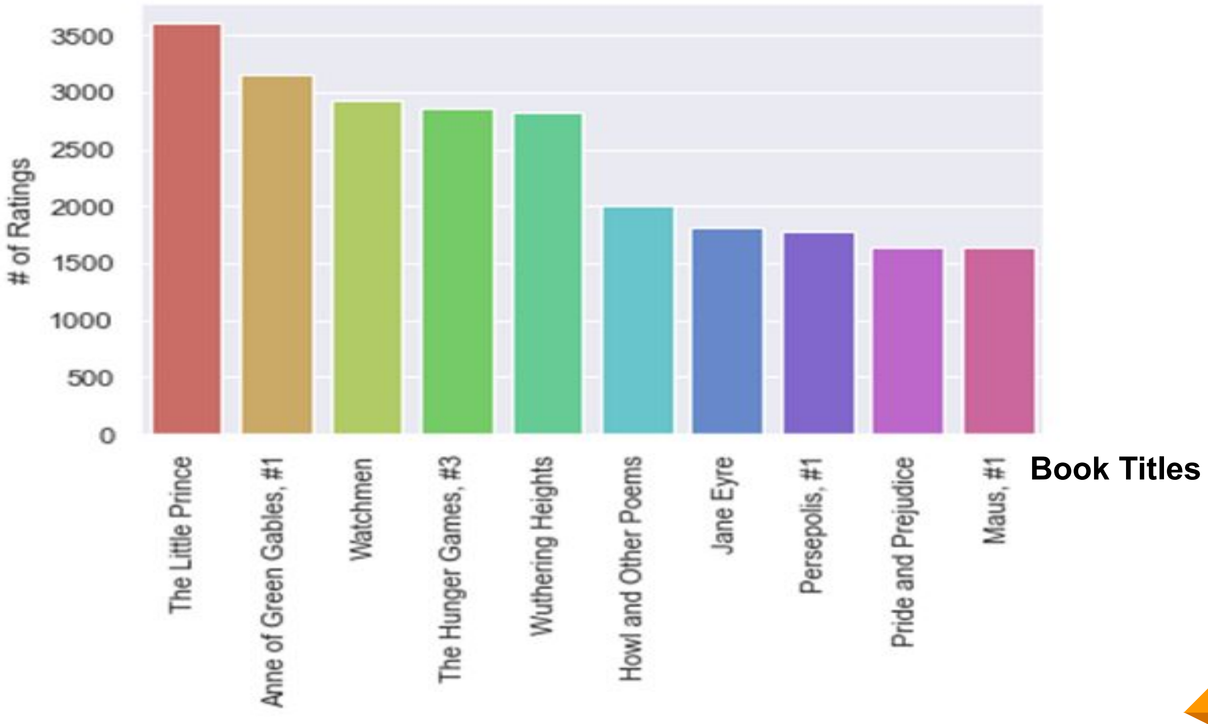
Top 6 Language Code



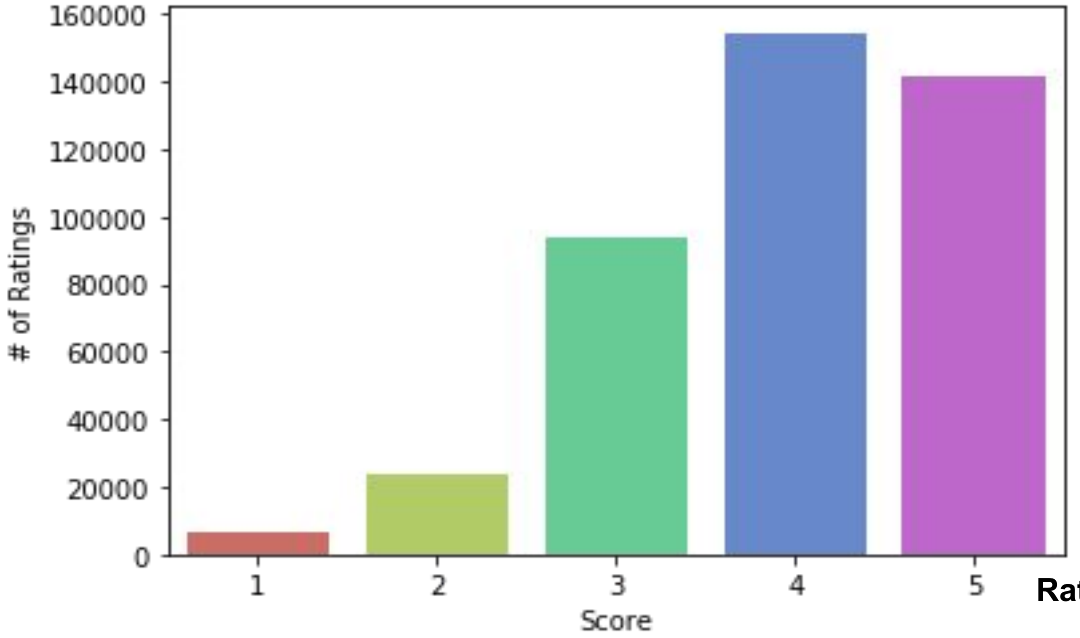
Different Roles of Authors



Top 10 Most Rated Books



Frequency of Rating Scores

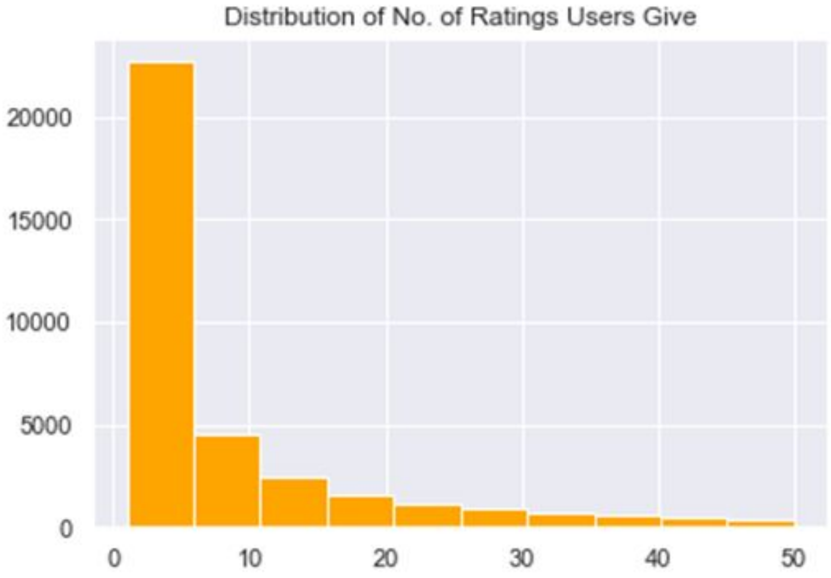


Rating Scores

The rating scores range from 1 to 5

Number of Books Users Rate

No. of Users

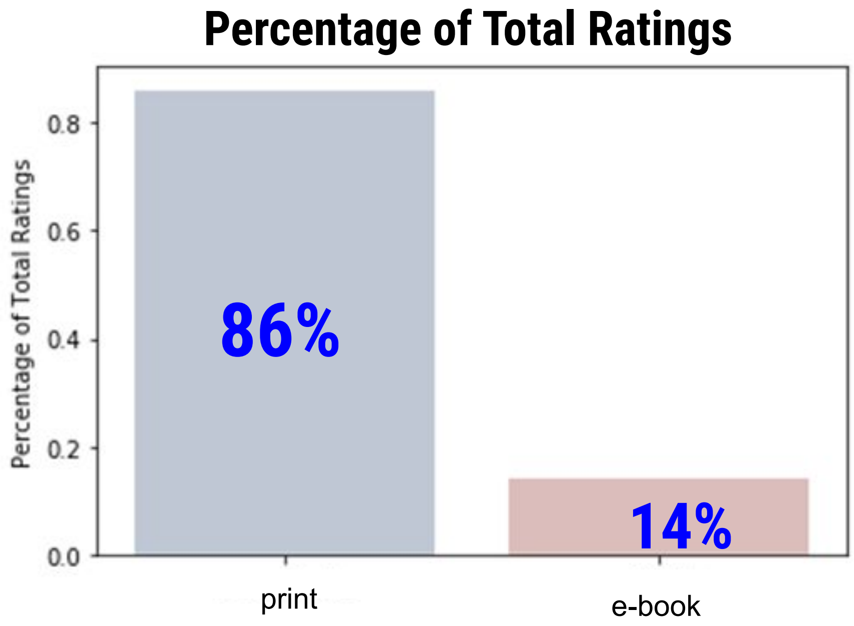


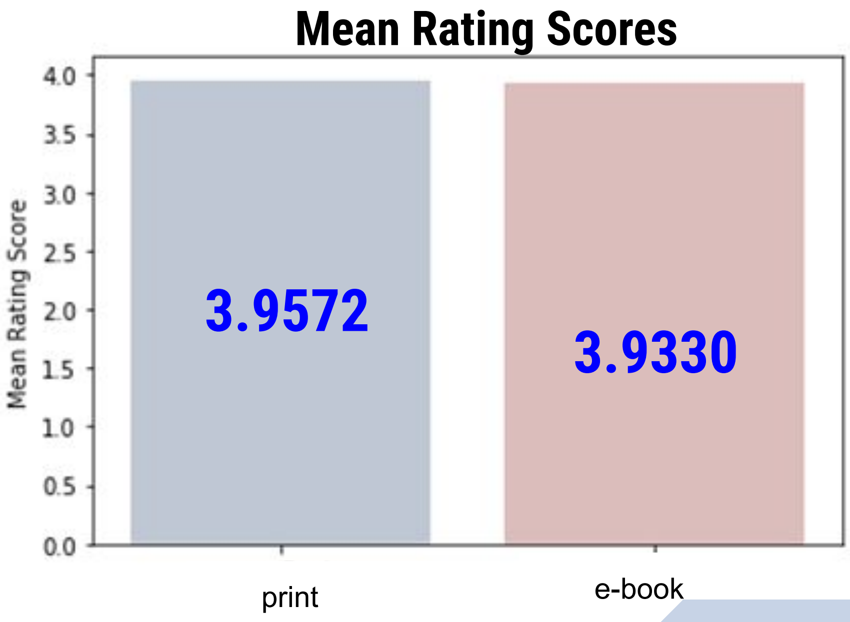
No. of Ratings

of Ratings by Each User

min	1
median	4
max	5559
mode	1

Ratings: E-Book vs Non E-book





2 Modeling

1. Naive Model
2. Content-based Filtering
3. Collaborative Filtering

Naive Model Using K Nearest Neighbors

Approach: Used to find the 5 closest books to a given book

1. Train model on the whole dataset with pivot table where:
   - each row is a book
   - each column is a user
   - ratings are in the table
2. Find the distance between the books using cosine similarity and Euclidean distance

3. The input to the KNN model [k=20 (arbitrary)] is a matrix constructed using the distance obtained in step 2

4. Model will return the 5 most similar books using KNN

Naive Model Using K Nearest Neighbors

A powerful classification algorithm used in machine learning

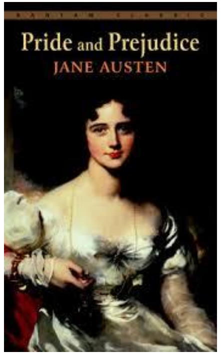
Advantages

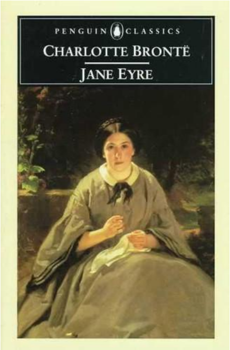
- Training is very fast
- Learn complex target functions
- Effective at preserving information

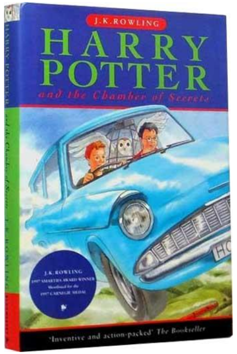
Disadvantages

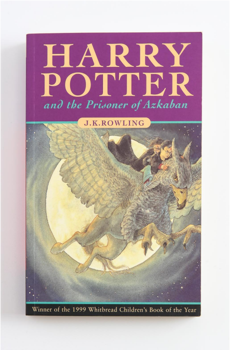
- Slow at query time

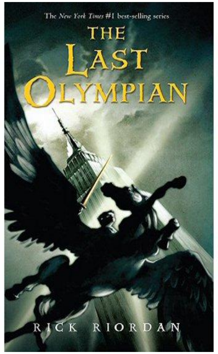
Output of KNN Using Cosine Similarity



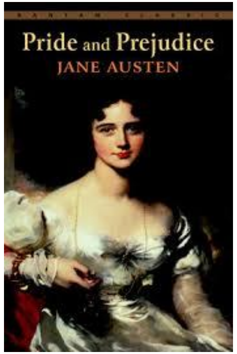


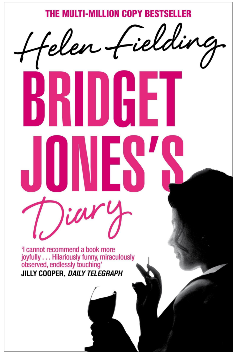


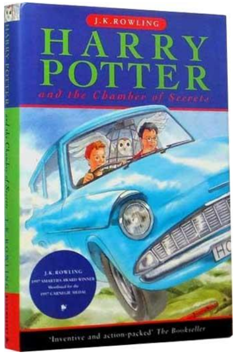


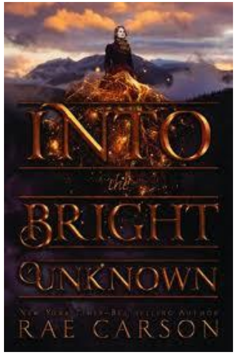


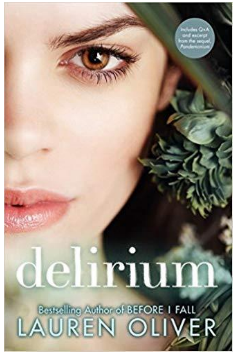
Output of KNN Using Euclidean Distance



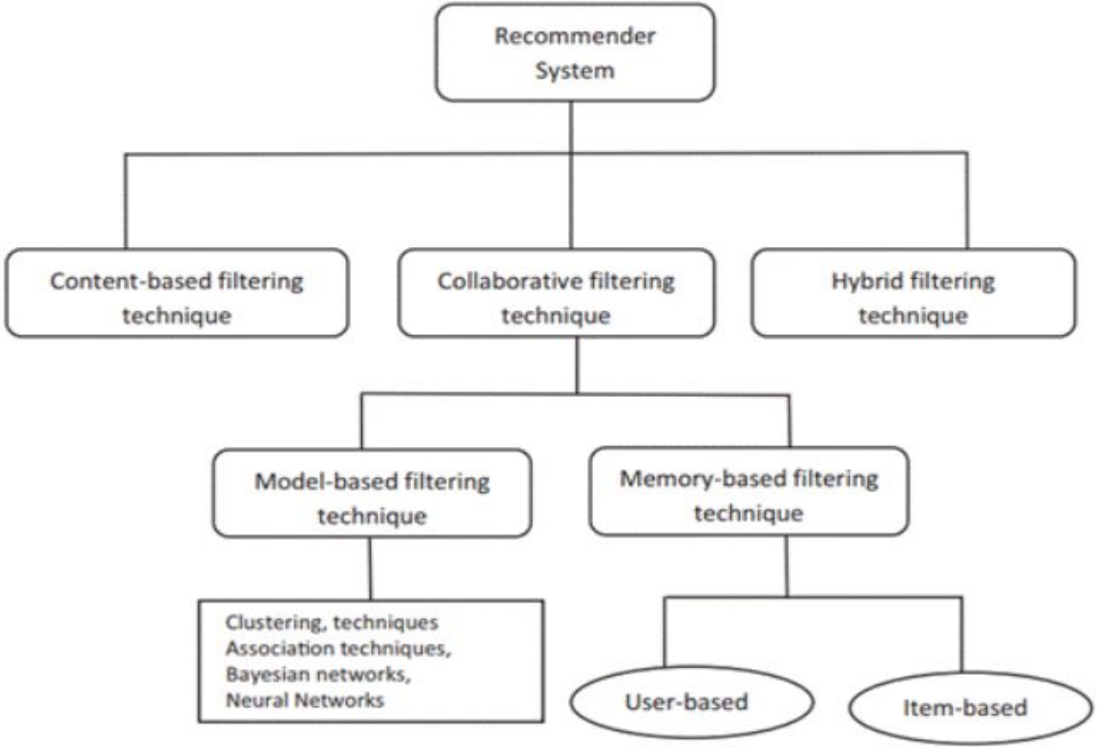








Different Modeling Techniques



Content Based Filtering

***Definition**: Based on users' preferences, the algorithm will simply pick items with similar content to recommend to the users.*

In our case, we use users' reviews rather than their ratings to build the recommendation system.

TF-IDF (Term Frequency – Inverse Document Frequency) along with **cosine similarity** between space vector model techniques are used in content-based filtering.

Content Based Filtering

Advantages

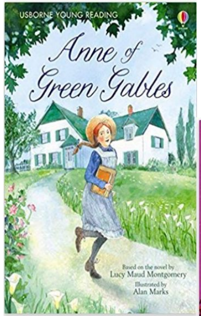
- Only analyze the items and user profile for recommendation and does not require us to find similarity between users
- No cold start: opposite to collaborative filtering, new items can be suggested before being rated by a substantial number of users

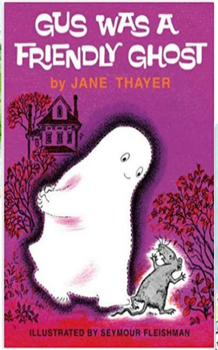
Disadvantages

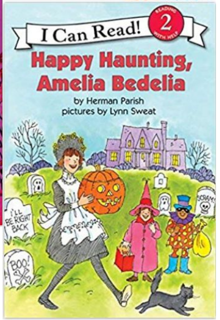
- Limited content analysis: if the content does not contain enough information to discriminate the items precisely, the recommendation will be not precisely at the end.
- Determining what characteristics of the item the user dislikes or likes is not always obvious.

Output of Content Based Filtering

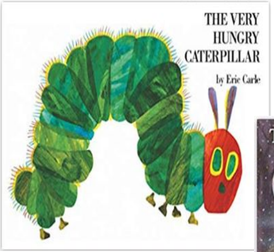
Top Rated Books

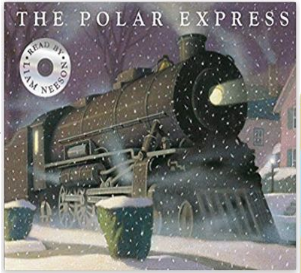


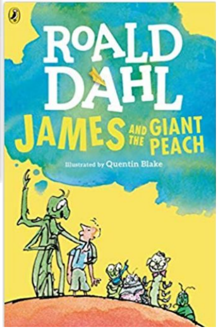




Recommended Books

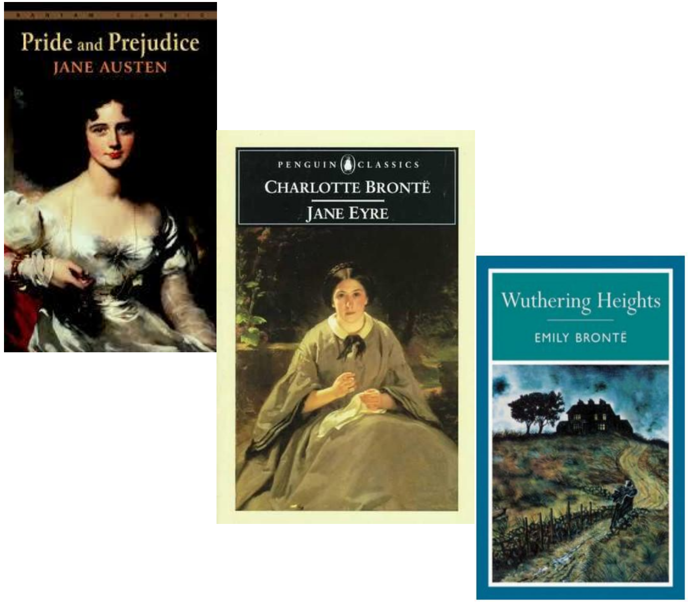






Output of Content Based Filtering

Top Rated Books



Recommended Books

Collaborative Filtering

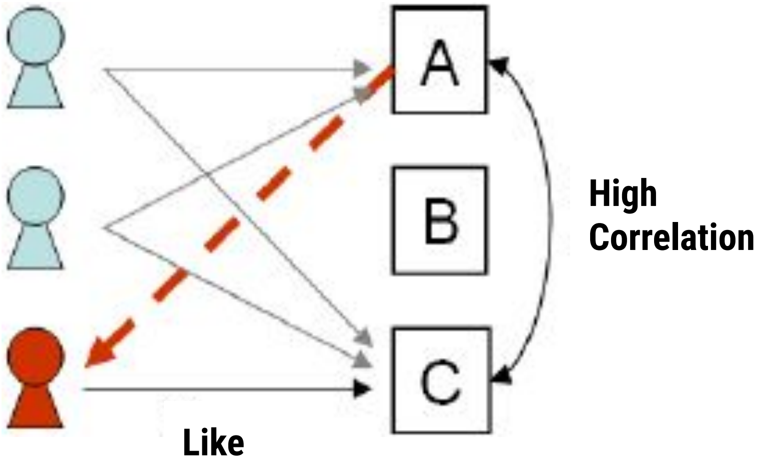
Collaborative filtering (CF) is a technique commonly used to build personalized recommendations.

In collaborative filtering, algorithms are used to make automatic predictions about a user’s interests by compiling preferences from several users with similar interests.

There are of two types of collaborative filtering techniques

- Item Based Collaborative Filtering
- User Based Collaborative Filtering

Item Based Filtering



Drachsler, Hendrik & Hummel, Hans & Koper, Rob. (2008). Using Simulations to Evaluate the Effects of Recommender Systems for Learners in Informal Learning Networks. CEUR Workshop Proceedings. 382.

- Item Based Collaborative Filtering looks for items that are similar to those that user has already rated and recommend most similar ones
- Similarity is not a measure of attributes. It refers to how people treat two items the same in terms of like and dislike.
- Here, we consider ratings rather than the features of the books

Item Based Filtering

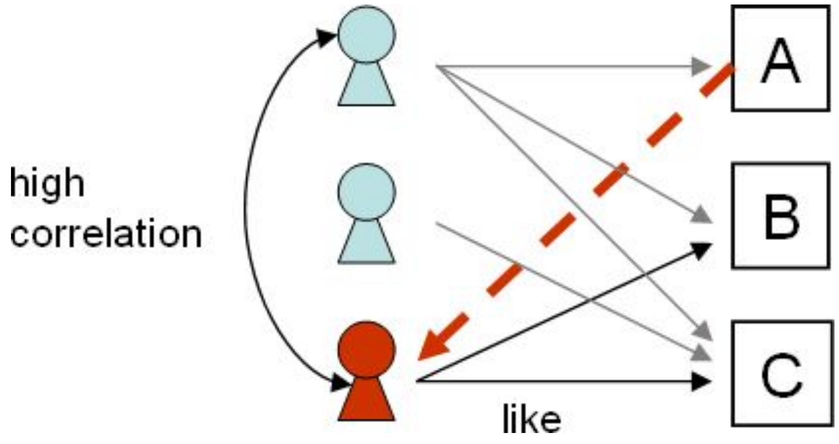
Advantages

- Similarity estimates between items are more likely to converge over time than similarity estimates between users.
- Item based recommendation engines begin with a list of a user's preferred items. Therefore, this technique does not need a nearest item neighborhood like user based recommendation engine

Disadvantages

- Difficult to discover bold recommendations since this technique is highly grounded in the data
- Not much evidence on whether the user will like the recommended item or not

User Based Filtering



- The method identifies users that are similar to the queried user and estimate the desired rating to be the weighted average of the ratings of these similar users.
- We try to find another user who has likes and dislikes books similar to our primary user

Kalz, Marco & Drachsler, Hendrik & Bruggen, Jan & Hummel, Hans & Koper, Rob. (2008). Wayfinding Services for Open Educational Practices. International Journal of Emerging Technologies in Learning (iJET). 3. 24-28.

User Based Filtering

Advantages

- Higher personalization is possible with user based filtering
- Model evolves as different types of users engage with the product

Disadvantages

- It is difficult to find recommendations for a new user since there are no priors available
- This method like item based filtering also faces the sparsity problem

Singular Value Decomposition

$$R = \begin{pmatrix} 1 & ? & 2 & ? & ? \\ ? & ? & ? & ? & 4 \\ 2 & ? & 4 & 5 & ? \\ ? & ? & 3 & ? & ? \\ ? & 1 & ? & 3 & ? \\ 5 & ? & ? & ? & 2 \end{pmatrix}$$

The matrix of user to book ratings is very **sparse**, with close to 99% of the entries missing. Our goal is to **predict the missing entries.**

Using SVD, we can reduce the dimensionality of this sparse matrix to obtain **latent factors** which explains a specific aspect of the data.

Singular Value Decomposition

$$\underset{m \times n}{\overset{\hat{X}}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} \approx \underset{m \times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}} \underset{r \times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}} \underset{r \times n}{\overset{V^{\mathsf{T}}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}}$$

X denotes the utility matrix, and U is a left singular matrix, representing the relationship between users and latent factors. S is a diagonal matrix describing the strength of each latent factor, while V transpose is a right singular matrix, indicating the similarity between items and latent factors.

Optimize $\min\sum(r_{ui} - U \cdot \sum \cdot V^T)^2$ using SGD

Singular Matrix Decomposition (http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/images/svd2.png)

Singular Value Decomposition

Advantages

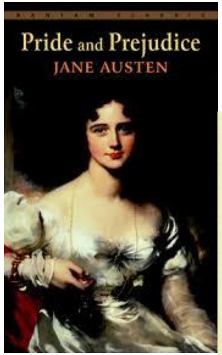
- Handles the scalability and sparsity issue encountered by CF

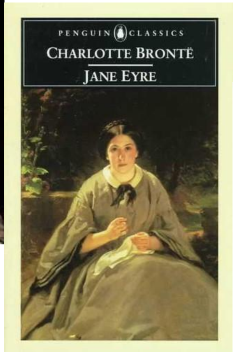
Disadvantages

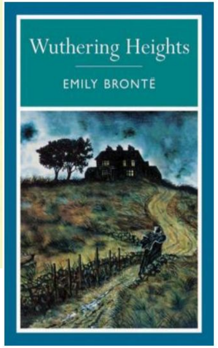
- The main drawback of SVD is that there is no to little explanation to the reason that we recommend an item to a user.

Output of Item Based SVD

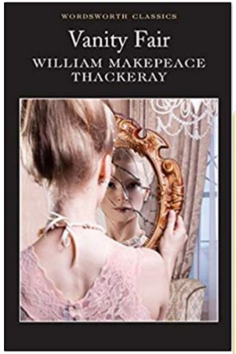
Top Rated Books

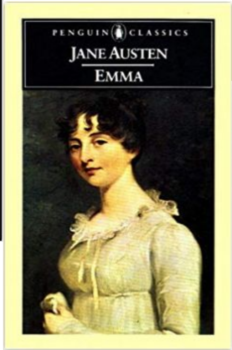


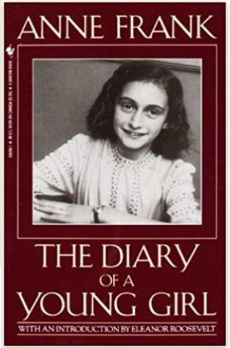




Recommended Books

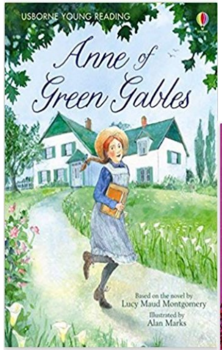


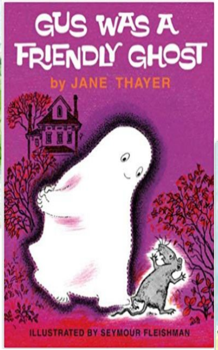


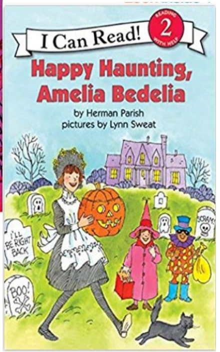


Output of Item Based SVD

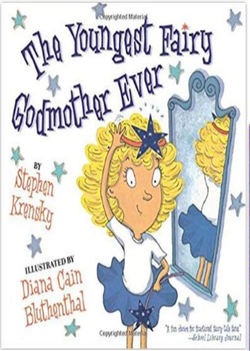
Top Rated Books

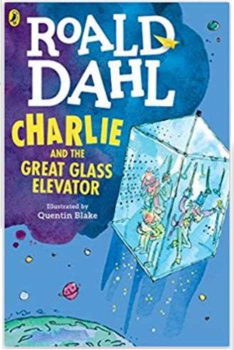


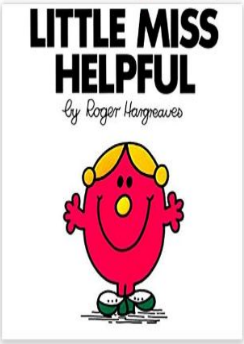




Recommended Books







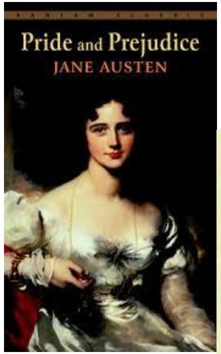
Output of User Based SVD

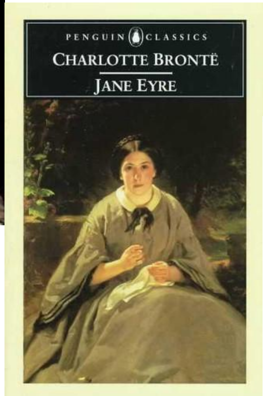
Top Rated Books

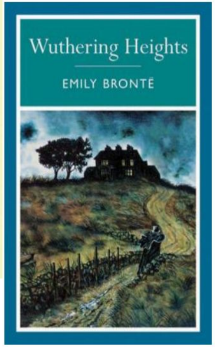
Recommended Books

Output of User Based SVD

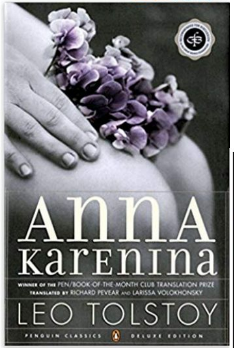
Top Rated Books

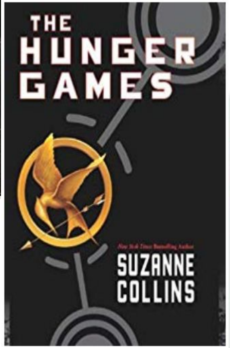


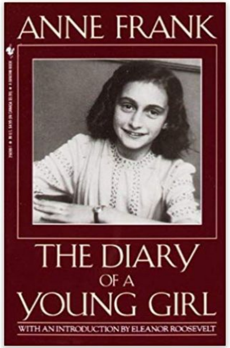




Recommended Books







Concerns with Collaborative Filtering

1. **Users' preference can change over time**, therefore, precomputing the matrix based on their neighboring users may lead to bad performance.
2. Even though item-based CF successfully avoids the problem posed by dynamic user preference as it's more static, the **computation grows** with both the customer and the product. In addition, **sparsity** is another concern.
3. In extreme cases, we can have millions of users and the **similarity between two fairly different books could be very high** simply because they have similar rank for the only user who ranked them both.

Source : https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75

3

Conclusion

Conclusion

Considering the pros and cons in the previous models, we can customize the recommendations such that we change the recommendation system based on **whether the user is a new user**.

New Users:

We can recommend books for new users based on a ***Naive Model*** as we wouldn't have historical information to form a personalized recommendation

Recurring Users: As the user-item data matrix is sparse, we would prefer using an ***SVD based CF approach with item based filtering*** as it provided better outputs when we qualitatively compared the different models

4

Appendix

Appendix: TF-IDF Score

TF stands for **Term Frequency**: How often does the term you are talking about appear in the document ?

IDF stands for **Inverse Document Frequency**: How rare it is for a document to have this term or for a tag to be applied to the movie? We calculate it by taking the inverse of how many documents have this tag divided by total number of documents.

Appendix: Cosine Similarity

$$similarity = cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}}$$

If similarity = 1, the two vectors are identical

If similarity = 0, the two vectors are orthogonal

Appendix: Euclidean Distance

Formula:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Output from Item based Collaborative filtering models using TuriCreate method

Model Based Matrix Factorization RMSE

#Matrix factorization Overall RMSE:
2.438094020279795

We will address Matrix Factorization later in the presentation

training_data, validation_data =
tc.recommender.util.random_split_by_user(books, 'user_id_y', 'title',item_test_proportion=0.3)

model =
tc.recommender.ranking_factorization_recommender.create(training_data,user_id='user_id_y', item_id='title',target='rating_y')

results = model.recommend(k=3)

model.evaluate(validation_data)

Item based Collaborative Filtering RMSE

#Item based Collaborative Filtering Overall RMSE:
2.981362699909857

training_data, validation_data =
tc.recommender.util.random_split_by_user(books, 'user_id_y', 'title',item_test_proportion=0.3)

model =
tc.recommender.item_similarity_recommender.create(training_data,user_id='user_id_y', item_id='title',target='rating_y')

Finished training in 2.81257s

items_similarity = model.get_similar_items()

model.evaluate(validation_data)

Additional References

All the models that we implemented can be found on :

https://github.com/alishafdes/GoodReads_Marketing_Analytics