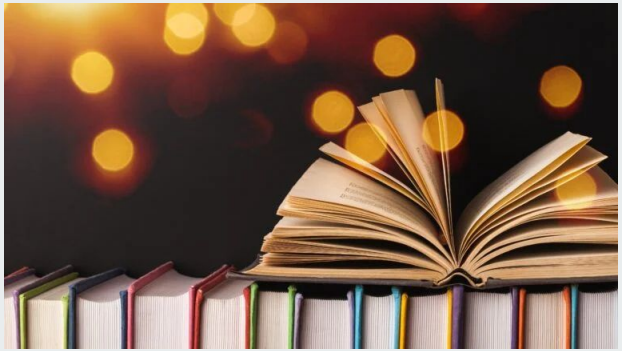
Book Recommendation with CF and Content-Based Filtering

Kevin Shi, Christopher Fluta, Dren Gara



Presentation Outline

1. Our Dataset
2. Data Cleaning and Preprocessing
3. User-Based Collaborative Filtering with KNN
4. Item-Based Collaborative Filtering
5. Content-Based Filtering with TF-IDF
6. Limitation and future improvement of our system

Our Dataset

- Dataset: Book-Crossing: User review ratings (Preprocessed_data.csv) from Kaggle
- 1031175 rows (users)
  - Number of unique users: 92107
- 19 columns
  - 'user_id', 'location', 'age', 'isbn', 'rating', 'book_title', 'book_author', 'year_of_publication', 'publisher', 'Summary', 'Language', 'Category', 'city', 'state', 'country'
- Motivation
  - Dataset contains user ratings -> CF with KNN
  - Features such as age, author, category, etc., allow for content-based filtering using TF-IDF

Number of Ratings:

Number of ratings: 1031175

Number of unique ratings: 11

Rating Value Counts:

Value	Count
0	647323
8	91806
10	71227
7	66404
9	60780
5	45355
6	31689
4	7617
3	5118
2	2375
1	1481

Data Cleaning and Preprocessing

- Processing the full dataset is too computationally expensive, so we cut down the dataset size
  - Only kept users from USA
  - Only kept users who rated books in 'en' language
  - Excluded users with less than 100 ratings
  - Excluded books with less than 20 ratings
- Processed dataframe shape: (70435, 14)
- Rating matrix shape: (813, 1768)
- Filled in missing values with zero

User-Based Collaborative Filtering with KNN

Purpose: recommend books to a user based on their nearest neighbors' ratings

1. Identify Similar Users
   a. Locate the nearest neighbors of the target user using the KNN model
   b. Exclude the user from their own neighbor list
2. Calculate Weighted Ratings
   a. Retrieve the ratings of the nearest neighbors
   b. Calculate the weighted average ratings for items the user has not yet rated
3. Generate Recommendations
   a. Identify items that the user has not rated
   b. Recommend items based on the highest average ratings among neighbors

```python
def get_recommendations(user_id, rating_matrix, knn_model, rating_matrix_scaled, n_neighbors, n_recommendations=10):
    # Get index of the user
    user_index = rating_matrix.index.get_loc(user_id)

    # Get nearest neighbors
    distances, indices = knn_model.kneighbors([rating_matrix_scaled[user_index]], n_neighbors=n_neighbors+1)  # +1 to include the target user

    # Get neighbors' indices (not including the user itself)
    neighbors_indices = indices.flatten()[1:]
    distances = distances.flatten()[1:]

    if len(neighbors_indices) == 0:
        print(f"No neighbors found for user {user_id}.")
        return pd.Series()

    # Get similar users
    similar_users = rating_matrix.index[neighbors_indices]
    print(f"Similar users to {user_id}:")
    for i, user in enumerate(similar_users):
        print(f"User: {user}, Distance: {distances[i]}")

    # Calculate the weighted average of the ratings from the nearest neighbors
    neighbor_ratings = rating_matrix.iloc[neighbors_indices]

    # Recommend items that the user has not rated yet
    user_ratings = rating_matrix.loc[user_id]
    unrated_items = user_ratings[user_ratings.isna()].index

    if len(unrated_items) == 0:
        print(f"User {user_id} has rated all items.")
        return pd.Series()

    # Compute average rating for unrated items
    recommendations = neighbor_ratings[unrated_items].mean().sort_values(ascending=False)

    return recommendations.head(n_recommendations)
```

Interpretation of Results

```
user_id = 2033
if user_id not in rating_matrix.index:
    print(f"User ID {user_id} not found in the dataset.")
else:
    recommendations = get_recommendations(user_id, rating_matrix, knn, rating_matrix_scaled, n_neighbors=10, n_recommendations=10)
    print(recommendations)
```

```
Similar users to 2033:
User: 51386, Distance: 0.6993345281500734
User: 77809, Distance: 0.7042121104782539
User: 79186, Distance: 0.7701577083139439
User: 179978, Distance: 0.795280923782677
User: 208568, Distance: 0.7953880272221509
User: 208141, Distance: 0.8010742442401894
User: 201783, Distance: 0.8031639446178255
User: 175003, Distance: 0.8173567676878375
User: 219683, Distance: 0.8265117988582696
User: 170634, Distance: 0.8319682436038494
book_title
Skeleton Crew                                                                10.0
The Te of Piglet                                                             10.0
The 9 Steps to Financial Freedom                                             10.0
The Cat in the Hat                                                           10.0
Suzanne's Diary for Nicholas                                                 10.0
Matilda                                                                      10.0
What to Expect the First Year                                                10.0
Midnight in the Garden of Good and Evil: A Savannah Story                    10.0
The Tao of Pooh                                                              10.0
Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))             10.0
dtype: float64
```

```
Similar users to 201783:
User: 179978, Distance: 0.6639419305818689
User: 198711, Distance: 0.7286309706385543
User: 208141, Distance: 0.7426408755469847
User: 175003, Distance: 0.763641490895293
User: 170634, Distance: 0.7842833922709755
User: 2033, Distance: 0.8031639446178255
User: 196985, Distance: 0.8147167213522095
User: 259625, Distance: 0.8147167213522095
User: 210792, Distance: 0.8147167213522095
User: 133868, Distance: 0.8147167213522095
book_title
The 9 Steps to Financial Freedom                                                    10.000000
Christmas Box (Christmas Box Trilogy)                                               10.000000
Ender's Shadow                                                                      10.000000
Charlie and the Chocolate Factory                                                   10.000000
Matilda                                                                             10.000000
What to Expect the First Year                                                       10.000000
Christy                                                                              9.000000
Chicken Soup for the Soul (Chicken Soup for the Soul)                                9.000000
Harry Potter and the Sorcerer's Stone (Book 1)                                       8.166667
Chicken Soup for the Christian Soul (Chicken Soup for the Soul Series (Paper))       8.000000
dtype: float64
```

Accuracy Metrics

- Accuracy decreases as n_neighbors increases
- RMSE increases as n_neighbors increases
- Precision decreases as n_neighbors increases
- Recall generally stays low

n	Accuracy	Precision	Recall	F1-score	RMSE
1	0.8271455	0.6554054	0.07677	0.1374424	3.14201
2	0.8261872	0.6228482	0.07874	0.1398208	3.15512
3	0.8255365	0.6048387	0.07914	0.1399743	3.16343
4	0.8247675	0.5851528	0.07954	0.1400452	3.17439
5	0.8241499	0.5713467	0.07890	0.1386648	3.18404
6	0.8238446	0.5644171	0.07888	0.1384179	3.18938
7	0.8236773	0.5604967	0.07908	0.1386177	3.19271
8	0.8236050	0.5590192	0.07894	0.1383555	3.19427
9	0.8234463	0.5556934	0.07874	0.1379496	3.19650
10	0.8232483	0.5517433	0.07827	0.1370992	3.19904

Offline Evaluation: Spearman Rank Correlation

- `from scipy.stats import spearmanr`
- Spearman Coefficient in range (-1, +1)
  - Larger positive values being more favorable since they indicate a stronger correlation between the predicted and actual rankings
- Scores are all very high for the top-10 and top-20 recommendations
- Scores decrease slightly as the number of neighbors increases
- Scores decrease slightly as we increase the top-n recommendations

```
Spearman Rank Correlation Scores for top-10 recommendations:
n_neighbors=1: 0.9891807190688452
n_neighbors=2: 0.9878108173362568
n_neighbors=3: 0.9871114465432644
n_neighbors=4: 0.987076697058867
n_neighbors=5: 0.986910745941212
n_neighbors=6: 0.9868161997628925
n_neighbors=7: 0.9867377161906097
n_neighbors=8: 0.9866924494333745
n_neighbors=9: 0.9867237031324948
n_neighbors=10: 0.986590498115909
Spearman Rank Correlation Scores for top-20 recommendations:
n_neighbors=1: 0.9863709562578192
n_neighbors=2: 0.9832757442608347
n_neighbors=3: 0.9808934516618629
n_neighbors=4: 0.979810068215124
n_neighbors=5: 0.9793395704252771
n_neighbors=6: 0.9789360253558824
n_neighbors=7: 0.9787074245772838
n_neighbors=8: 0.9788260337535619
n_neighbors=9: 0.9787219597915408
n_neighbors=10: 0.9786109552433198
```

Offline Evaluation: R-score

- R-score: measure of utility
  - Initial Increase in R-Score (n=1 to n=2)
  - Fluctuations in Mid-Range (n=3 to n=9)
  - Significant Drop at n=10
- Novelty?
- Serendipity?

$$U(u,i) = \frac{\text{rating-based utility}}{\text{ranking-based utility}} = \frac{max\{\text{rating}_{ui} - \text{neutral}, 0\}}{2^{(v_i - 1)/\alpha}}$$

```
R-scores (top-10 items) by n_neighbors:
n_neighbors=1: 0.7052956823633222
n_neighbors=2: 0.7351961556377895
n_neighbors=3: 0.5508598319349033
n_neighbors=4: 0.43867926366198395
n_neighbors=5: 0.5511304999328153
n_neighbors=6: 0.5039570280995685
n_neighbors=7: 0.6252825209137143
n_neighbors=8: 0.6008484670504077
n_neighbors=9: 0.60747675738075
n_neighbors=10: 0.3507794955789136
```

Item-Based Collaborative Filtering

Purpose of Item-Based Collaborative Filtering

- Recommend books to users by analyzing the similarities between items (books) based on user ratings.
- Focus on finding items similar to those a user has already interacted with and using these similarities to suggest new items.

Identify Similar Items

- Construct an item-user matrix where rows represent books and columns represent users.
- Use the K-Nearest Neighbors (KNN) algorithm with cosine similarity to calculate the similarity between items.
- Identify the top N most similar books for each item in the dataset.

Generate Recommendations

```
# Instantiate the KNN model
knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=5)
knn.fit(item_user_matrix)
```

```
NearestNeighbors(algorithm='brute', metric='cosine')
```

- For a given book, the KNN model finds the most similar books based on user ratings.
- Displays the top N similar books, providing personalized recommendations for each item.

```
def recommend_books_knn(book_title, knn_model, item_user_matrix, top_n=5):
  # Get the index of the book
  book_index = item_user_matrix.index.get_loc(book_title)

  # Find the k nearest neighbors
  distances, indices = knn_model.kneighbors(item_user_matrix.iloc[book_index, :].values.reshape(1, -1), n_neighbors=top_n + 1)

  # Get the indices of the nearest neighbors (excluding the book itself)
  recommended_indices = indices[0][1:]

  # Get the book titles for the recommended indices
  recommended_books = [item_user_matrix.index[i] for i in recommended_indices]

  return recommended_books
```

Example Recommendations

For each of the top 5 most popular books, recommendations are generated using the item-based KNN model.

- Recommendations for 'The Da Vinci Code': ["Widow's Walk", 'TickTock', 'Touching Evil', 'Doing Good', 'The Arraignment']
- Recommendations for 'Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))': ['Harry Potter and the Goblet of Fire (Book 4)', 'Harry Potter and the Prisoner of Azkaban (Book 3)', 'Harry Potter and the Order of the Phoenix (Book 5)', 'Harry Potter and the Chamber of Secrets (Book 2)', 'A Time to Kill']
- Recommendations for 'The Secret Life of Bees': ['Under the Tuscan Sun', 'Good in Bed', "Dude, Where's My Country?", "Patty Jane's House of Curl", 'Snow Falling on Cedars']
- Recommendations for 'Harry Potter and the Goblet of Fire (Book 4)': ['Harry Potter and the Prisoner of Azkaban (Book 3)', 'Harry Potter and the Order of the Phoenix (Book 5)', 'Harry Potter and the Chamber of Secrets (Book 2)', "Harry Potter and the Sorcerer's Stone (Book 1)", "Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))"]
- Recommendations for 'Bridget Jones's Diary': ['The Red Tent (Bestselling Backlist)', 'Under the Tuscan Sun', 'A Walk to Remember', 'Zoya', 'SHIPPING NEWS']

Accuracy Metrics

Evaluated the model using accuracy, precision, recall, and F1-score.

Evaluation Results for "Harry Potter and the Goblet of Fire (Book 4)"

Top_n	Accuracy	Precision	Recall	F1-score
1	0.6190	0.6190	1.0000	0.7647
2	0.5515	0.5515	1.0000	0.7109
3	0.5811	0.5811	1.0000	0.7350
4	0.5672	0.5672	1.0000	0.7238
5	0.4990	0.4990	1.0000	0.6658
6	0.4809	0.4809	1.0000	0.6495
7	0.4682	0.4682	1.0000	0.6378
8	0.4664	0.4664	1.0000	0.6361
9	0.4498	0.4498	1.0000	0.6205
10	0.4375	0.4375	1.0000	0.6087

Content-Based Filtering

Data Cleaning and Preprocessing

- Regular preprocessing was applied
  - Keeping only books from USA and in English ensures better content recommendations
- Data was further filtered for Content-Based system
- Kept content-relevant columns
  - Title, author, summary, category, publisher, isbn, year of publication, and language
- Dropped duplicate entries of book title, isbn, and summary
  - Some books have multiple versions in publication, so there can be two versions of a book with the exact same name but different publishers
- Reset Indices for easier access to entries

Content Based Recommender Method

1. Feature Selection
    a. Created combined_features out of:
        i. Book_author
        ii. Summary
        iii. Category
        iv. Publisher
    b. Did not include book_title
    c. String stripped combined_featuers to ensure consistency
2. TF-IDF Vectorization of stripped combined_features
3. Create Cosine Similarity matrix from TF-IDF vectors
4. Get pairwise similarity for a given book
5. Return top 10 books with highest pairwise similarity, excluding given book

```
Recommendations for 'The Hobbit' recommendations:
                                                Book Title
0  The Fellowship of the Ring (The Lord of the Ri...
1     The Two Towers (The Lord of the Rings, Part 2)
2                                     The Silmarillion
3                            The Fellowship of the Ring
4   The Book of Ruth (Oprah's Book Club (Hardcover))
5                          The Wind Done Gone: A Novel
6               Taltos: Lives of the Mayfair Witches
7                               Interpreter of Maladies
8   The Heart Is a Lonely Hunter (Oprah's Book Club)
9                                     Rubyfruit Jungle
```

System Results

1. Handling Exact and Partial Matches:
   a. The recommender attempts to find exact matches for a book title; if none are found, it falls back to partial matches.
2. Recommendation Outputs:
   a. Displays both recommended book titles and their similarity scores.
   b. Example: For "Dune," the system recommends other science fiction books with high similarity scores, including those in the Dune series.

Recommendations for 'Dune' recommendations:

	Book Title	Similarity Score
0	House Atreides (Dune: House Trilogy, Book 1)	0.303155
1	Dune Messiah (Dune Chronicles, Book 2)	0.289427
2	The Hunt for Red October	0.283712
3	Children of Dune (Dune Chronicles, Book 3)	0.249197
4	MY SWEET AUDRINA	0.228903
5	Song of Solomon (Oprah's Book Club (Paperback))	0.226211
6	Heaven	0.219066
7	Winter's Tale	0.215134
8	Gates of Paradise	0.185545
9	Where Are the Children	0.182447

Diversity score for 'Dune' recommendations: 0.8620482214930824

Recommendations for 'Animal Farm' recommendations:

	Book Title	Similarity Score
0	1984	0.152695
1	Of Mice and Men (Penguin Great Books of the 20th Century)	0.119605
2	The Cricket in Times Square (Newbery Winners for Microcomputers)	0.118604
3	Firestarter	0.110605
4	Murder in Georgetown (Capital Crime Mysteries)	0.103145
5	The Green Mile: Night Journey (Green Mile Series)	0.098591
6	The Green Mile: Coffey on the Mile (Green Mile Series)	0.098591
7	A Thousand Acres (Ballantine Reader's Circle)	0.097151
8	Battlefield Earth: A Saga of the Year 3000	0.095135
9	Pay It Forward: A Novel	0.093390

Diversity score for 'Animal Farm' recommendations: 0.9449747280794807

Accuracy Metrics

- Similarity Scores:
  a. Each recommendation comes with a similarity score, indicating how closely related the book is to the input.
  b. Higher scores indicate stronger content similarity.
- Manual Validation:
  c. There is logical consistency of recommendations for tests performed on many books
    i. Books in a series recommend other books within the series
    ii. Books not in series recommend other books by the same author or similar topics
    iii. Category (genre) an d author carry a lot of weight in recommendations

Offline Evaluation: Diversity Score

- Diversity Scores:
  a. Measures the variety of recommendations based on the average pairwise cosine similarity among the top 10 recommendations.
  b. Diversity score is the inverse of similarity—higher scores indicate more diverse recommendations.
- Book title decreases diversity without and discernible benefit to recommendations

With Title as Feature

```
Diversity score for 'Dune' recommendations: 0.8531272229226876
Diversity score for 'Dune 2' recommendations: 0.8915165964676427
Diversity score for 'The Hobbit' recommendations: 0.9482575827164509
Diversity score for 'The Testament' recommendations: 0.968758918036998
Diversity score for 'Animal Farm' recommendations: 0.940056906672012
```

With Title as Feature

```
Diversity score for 'Dune' recommendations: 0.8620482214930824
Diversity score for 'Dune 2' recommendations: 0.9152126833145062
Diversity score for 'The Hobbit' recommendations: 0.9548689884047775
Diversity score for 'The Testament' recommendations: 0.9712675355149434
Diversity score for 'Animal Farm' recommendations: 0.9449747280794807
```

Limitation and future improvement of our system

Collaborative Filtering

- Limitations
  - Only considers users from USA and books written in English
  - Large scale computation/memory use
  - Challenges with sparsity
  - Cold Start Problem
  - Item-based methods recommendations may lack novelty
- Future Improvement
  - Broaden system to consider users from different countries and books in different languages
  - Try filling in missing values with the mean of the particular user or book ratings (instead of 0)
  - Use user studies for system evaluation

Content-based Filtering

- Limitations
  - Lack of novelty and serendipity
  - Similarity score is not a comprehensive measure
  - Only considers books written in English
  - Handling of multiple versions of the same book is an oversight
- Future Improvement
  - Incorporate user profiles and or combine with a Collaborative Filtering method, particularly User-Based
  - Implement a canonical title system to handle multiple publications of the same book