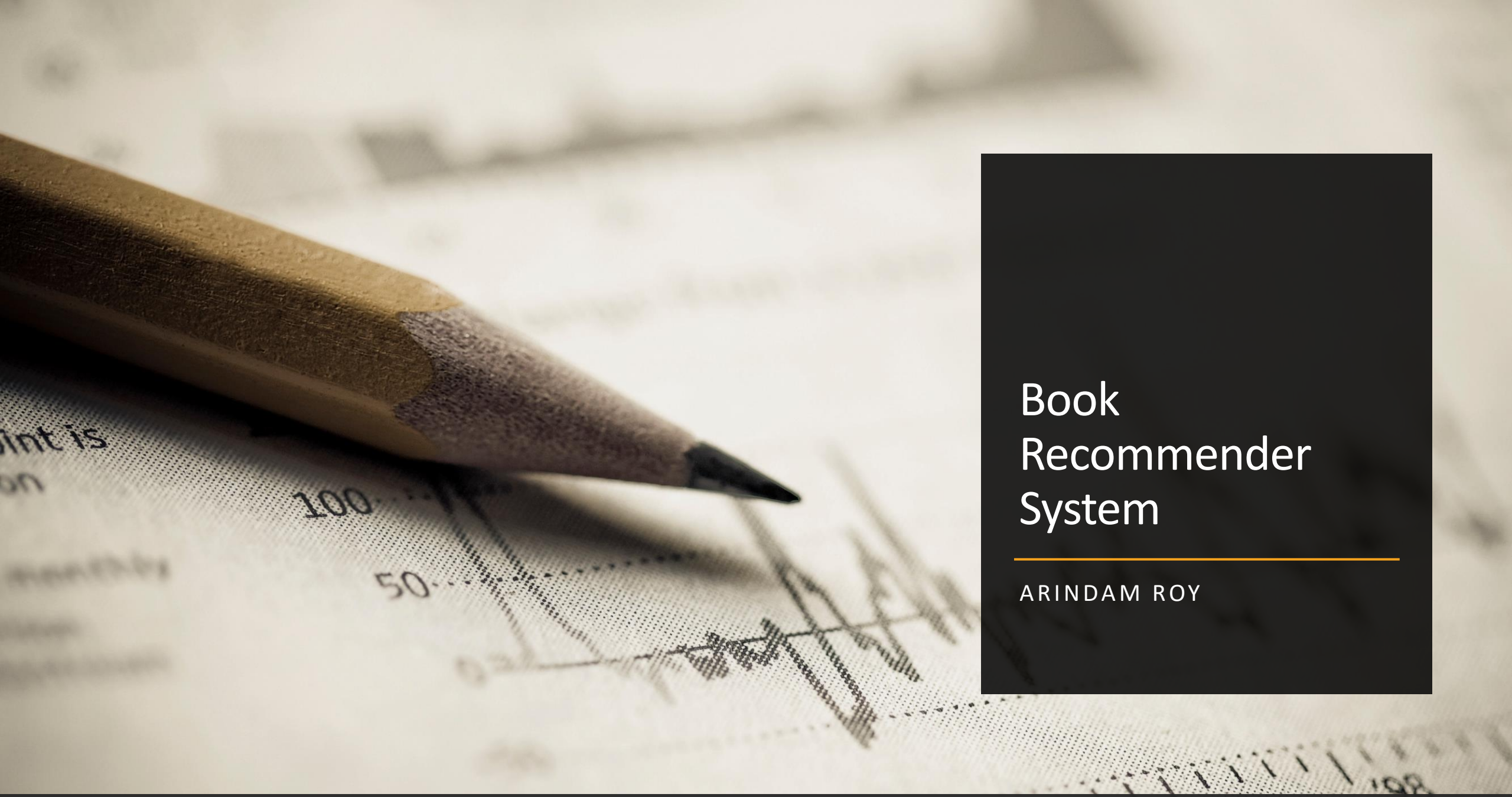


Book Recommender System

ARINDAM ROY

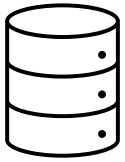
Introduction

Problem Statement

The book recommender system aims to address the challenge of effectively recommending relevant books to users based on their individual interests and the overall popularity of books. The primary objective is to develop a recommendation engine that can analyze user preferences and behavior to provide personalized book recommendations, while also considering the popularity and trends within the user community. By doing so, the system can enhance user satisfaction, engagement, and book discovery.

EXPLORATORY DATA ANALYSIS

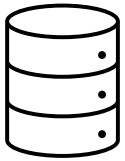
Data Description



Books

Description: This table provides information about the books included in the Book-Crossing Dataset.

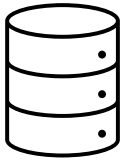
Columns: *isbn, book_title, book_author, year_of_publication, publisher,img_s, img_m, img_l, book_title_identifier*



Users

Description: The Users table contains information about the individuals who participate in the Book-Crossing Dataset. It includes attributes such as the user's ID, their age and location.

Columns: *user_id, city, state, country*



Ratings

Description: The Ratings table captures the ratings given by users to different books in the Book-Crossing Dataset. It includes attributes such as the user ID, the book's ISBN and the rating value.

Columns: *isbn, user_id, rating*

Primary Findings In Data

Key Statistics

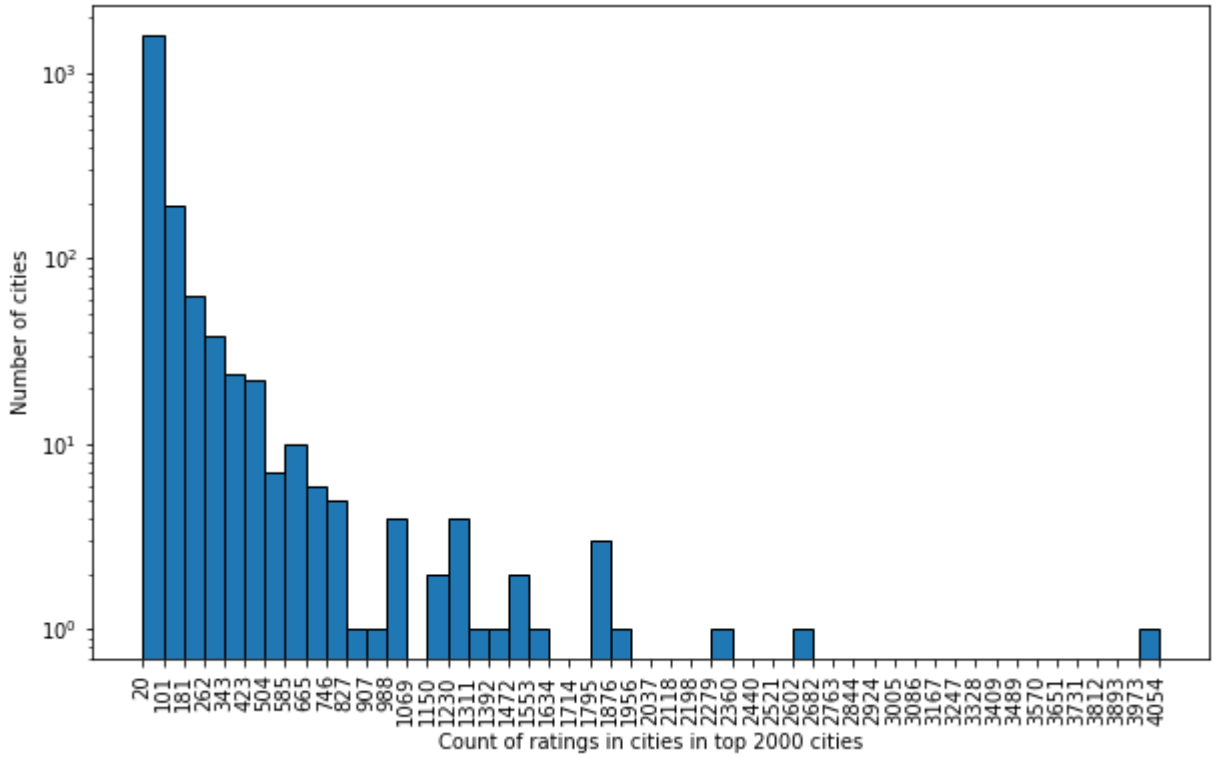
Statistic	Count
Total number of books	271,360
Total number of users	278,858
Total number of ratings	1,149,780
After Data Fixes	
Total number of books	242,505
Total number of users	278,858
Total number of ratings	1,144,516

Issues In Data And Fixes

1. There are various ISBN codes for the same books. All books with the same title and author surname have been assigned one ISBN.
2. The age data in user table has 70% of the data as null. This column has been dropped.
3. City, country and location are in the same column (location) in the users table. These have been split into respective columns
4. ~70,000 reviewed books in the reviews table are not present in the books table. ~36,000 of these are non – zero reviews.

User Geographic Analysis: City

Count Of Ratings In Cities In Top 2000 Cities

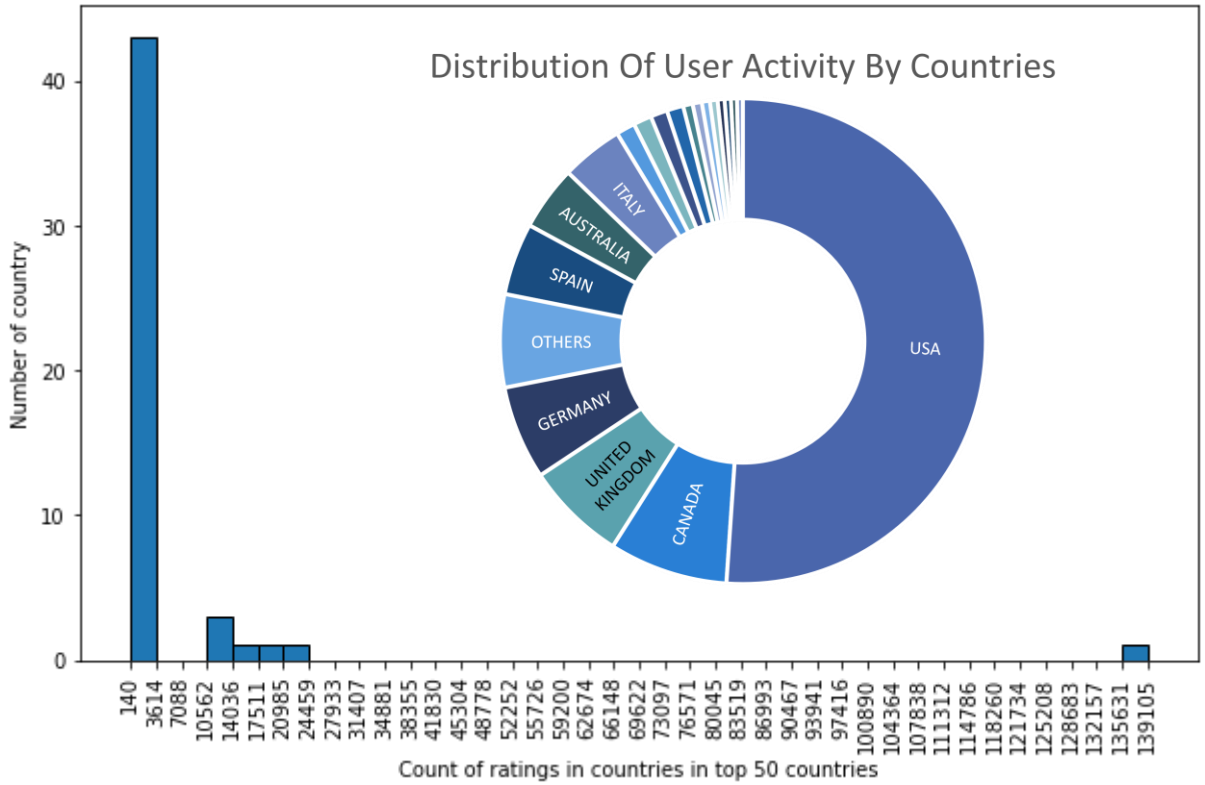


Observations

1. Most of the cities have less than 100 searches.
2. There are only 9 cities with over 1500 searches.
3. London is an outlier with over 4000 searches.
4. Top cities : London, Barcelona, Toronto, Madrid, Sydney, Melbourne, Portland, Vancouver, Chicago, Seattle, New York, Milano, San Diego, Berlin, San Francisco, Ottawa, Houston, Paris, Los Angeles, Austin, Roma

User Geographic Analysis: Country

Count Of Ratings In Countries In Top 50 Countries



Observations

1. There are 7 countries generating a bulk of traffic. Following are the countries: USA, Canada, United Kingdom, Germany, Spain, Australia, Italy
2. Only 11 other countries have generated over 1000 ratings.
3. There are erroneous country names in the user table

Ratings Analysis

Key Statistics

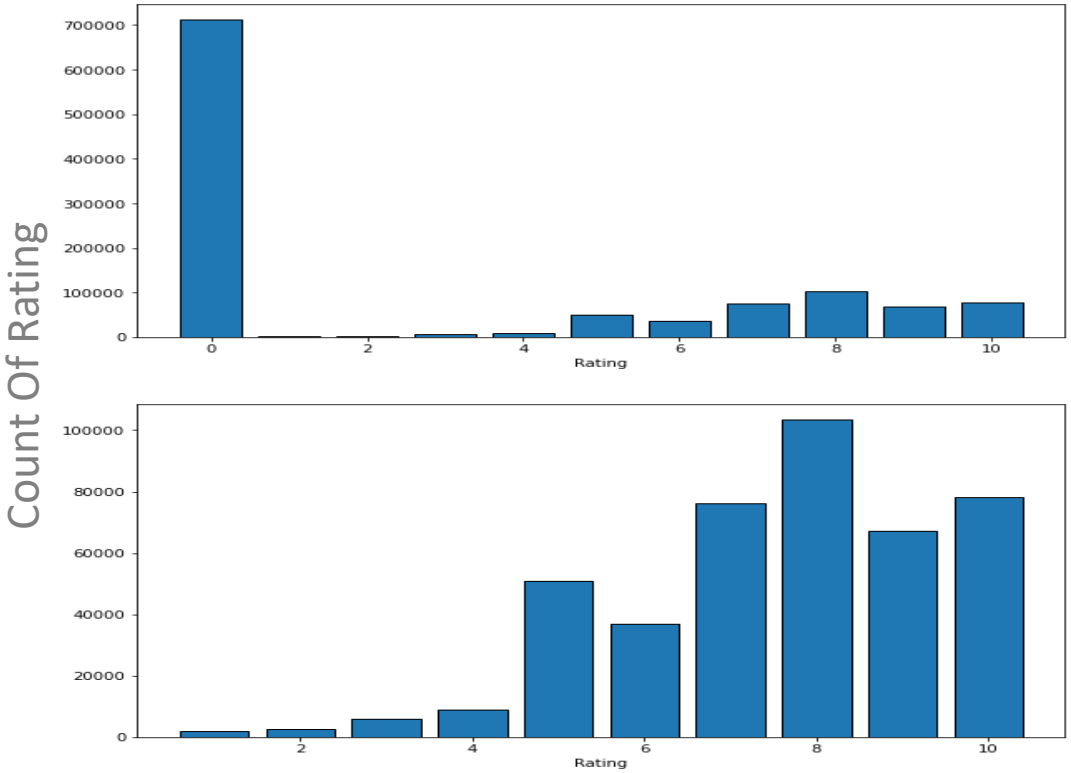
Ratings Statistic	Value
Total count of rated books	152,123
Maximum count of ratings on a book	505
Minimum count of ratings on a book	1
Median count of ratings on a book	1
Count of books with higher than or equal to 10 ratings	4,744
Count of books with higher than or equal to 5 ratings	11,952
Count of books with higher than or equal to 2 ratings	45,255

Observations

1. Most users have only ever rated any book once.
2. Of users who have rated a book once, 30% have given another rating.
3. There are ~12000 users who have given 5 or more ratings.
4. There are ~5000 users who have given 10 or more ratings.

Ratings Analysis

Unique Ratings Histogram With And Without 0 Rating

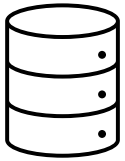


Observations

1. 62% of the rating data is 0 rating. This needs to be investigated further. This could be due to the design of data collection.
2. Median, mean and std of the nonzero ratings is 8.0, 7.6 and 1.8.
3. The zero ratings will be dropped for further analysis.

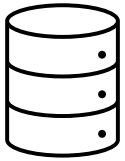
PROPOSED PREDICTIVE MODELS

Approaches Proposed



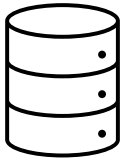
Collaborative Filtering

A closeness of each book is generated with all other books. The highest scoring books are recommended to the user.



Social Network of Books (Graph Method)

Each book is considered a node of a graph, like a social network. Further, clusters are identified in the graph and recommendations are generated according to these clusters



Merged Approach

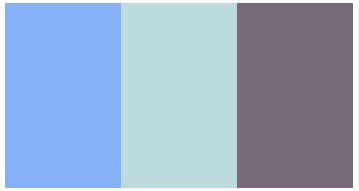
In the cluster identified, collaborative filtering is used to recommend books.

Collaborative Filtering - Technical Detail

STEPS FOR RECOMMENDING WITH COLABORITIVE FILTER

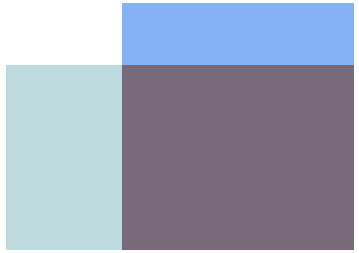
FILTER RATINGS DATA

We filter our ratings data by a threshold of rating count for the user and book. This is done to only keep statistically significant items.



PIVOT RATINGS DATA

We pivot the ratings data such that the columns are users, and the rows are the books with rating as value. This sparse matrix is a high dimensional vector space with the books as the vectors.



CALCULATE COSINE SIMILARITY

We calculate the cosine similarity of each book with all other books. The length of the resulting vector is an indicator of the similarity of book. We get a square matrix with book cosines scores as values.

PREDICT HIGHEST SIMILAR BOOK

For any input book, we go to its closest cosine similarity score and return the n highest scoring books.

Column	Color
User ID	
ISBN	
Rating	
Cos Score	

Core Principle Of Working

The collaborative filtering method works by projection of vectors. In a high dimension vector space made of each user, we create a vector with the length as score given by user. Then, we project each vector on any given vector. This gives us a measure of closeness of any two books .

Collaborative Filtering – Pros/Cons

Pros	Cons
Simplicity Cosine Collaborative Filtering is relatively easy to understand and implement. It doesn't require complex mathematical operations and can be implemented efficiently.	**Data sparsity** Cosine Collaborative Filtering relies heavily on the availability of user-item interactions or feature vectors. In situations where the data is sparse, meaning there are few interactions or features, the algorithm may struggle to find meaningful similarities and provide accurate recommendations.
User cold-start problem Cosine Collaborative Filtering can handle the cold-start problem, which occurs when there is limited or no information available about a new user. It can still make recommendations by finding similarities with existing users or items based on their feature vectors.	**Scalability** As the number of users or items grows, the computational complexity of computing pairwise similarities increases. This can become a bottleneck in large-scale systems with millions of users or items.
	Rare Items Problem/ Item cold-start problem In the solution, we generate a matrix only with statistically significant items. This leaves out books which are rarely read or not in DB. Furthermore, even for the filtered items, it may not show up for low user count.

Graph Method (Social Network of Books) - Technical Detail

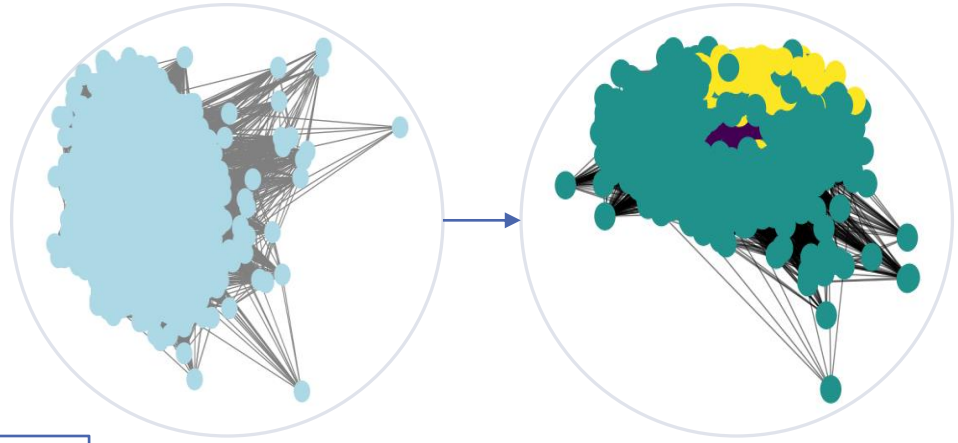
STEPS FOR RECOMMENDING WITH GRAPH METHOD

KEEP POSITIVE RATINGS

- Filter rating data by user rating count and book rating count.
- Scale ratings for every user. (MinMaxScaler)
- Only retain ratings with over a certain threshold of rating.

CREATE EDGE LIST OF BOOKS

Considering each book as a node, create an edge in the graph only if a certain threshold of users are in common. This is our network of graph.



RUN COMMUNITY DETECTION

- Run a community detection algorithm (Louvain, GN method, K-clique, et-cetera)
- Assign each book its community/communities.

PREDICT MOST POPULAR IN COMMUNITY

For a given book, return the most popular books in the community.

Graph Method (Social Network of Books) – Pros/Cons

Pros	Cons
Relational Understanding Similar items (books) have an edge generated in the graph. This gives knowledge of the orientation of	**Rare Items Problem/ Item cold-start problem** In the solution, we return only the top book. This poses a challenge of detecting rare items/highly similar items. Items not in the DB also remain missing.
Genre Detection/Overlapping Communities Genres can be detected using this method. Further, user input may lead to better recommendations.	**Resolution Limit** Finding niche communities is a challenge as it is tough to detect smaller communities. An unsupervised learning might fail to find niche communities.
Dynamic Networks Various different networks can be created of strong/weak affinity. Running analysis on each of these graphs may lead to different results.	**Scalability** Solution might have computation problems when data is scaled.

Graph Method With Collaborative Filter- Technical Detail

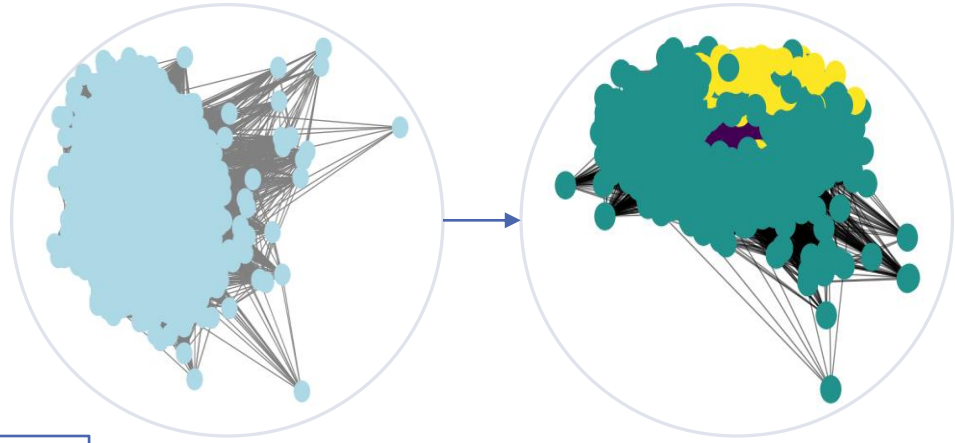
STEPS FOR RECOMMENDING WITH GRAPH METHOD

KEEP POSITIVE RATINGS

- Filter rating data by user rating count and book rating count.
- Scale ratings for every user. (MinMaxScaler)
- Only retain ratings with over a certain threshold of rating.

CREATE EDGE LIST OF BOOKS

Considering each book as a node, create an edge in the graph only if a certain threshold of users are in common. This is our network of graph.



RUN COMMUNITY DETECTION

- Run a community detection algorithm (Louvain, GN method, K-clique, et-cetera)
- Assign each book its community/communities.

PREDICT MOST SIMILAR BOOK IN COMMUNITY

For a given book, return the most similar books in the community.

Key Advantages

Instead of predicting the most popular book in the community, we predict the most cosine similar book.

1. More similar item are suggested.

2. Space complexity goes down.

METRIC OF PERFORMANCE

Metric of performance

1. It is tough to generate any metric for such unlabeled data.
2. One indicator of performance can be developed by following the below steps:
    1. Pick random frequent users from the ratings table.
    2. Get top 9 predictions for 10% of the positive rating books for the users.
    3. Find the intersection of the recommendations with the 90% of the remaining books.
    4. Generate an average score.
3. This is only an indicator metric. The numbers can be tuned to perform better.
4. These numbers do not have any tangible meaning. They as a representation of the recommendation engine's ability to mimic user behavior.

DEPLOYMENT TO PRODUCTION

Deployment Details

Collaborative Filtering

RUN MODEL AS A BATCH JOB

- Run the cos similarity model in a distributed computing data structure (py-spark dataframes).

→

SAVE SIMILARITY TUPPLES

- Save as tuples in a py-spark RDD to apply map reduce further.

→

RUN SQL QUERY/MAP REDUCE ON USER INPUT

- Run map reduce to find the highest pair of the searched item.

Graph Method/ Graph Method + Collaborative Filter

STORE GENERATED GRAPH

Storing the graph generated in distributed graph DB (HDFS, Apache Cassandra) would be ideal

→

ADD INCREMENTS TO GRAPH AS A BATCH JOB (Batch-1)

In fixed intervals, add data to the graph by computing the last highest index in the ratings table.

→

GENERATE COMMUNITY FOR EACH ITEM(Batch-2)

With spark, run community detection algorithm to generate community. Join the results in the ratings.

→

RECOMMEND TOP ITEM IN COMMUNITY

Recommend the top item in the community

↓

RUN COLABORATIVE FILTER FOR EACH COMMUNITY(Batch-3)

For each community generate a similarity score matrix and save as tuples. Since the has a O(n^2) complexity, this is computationally cheaper.

→

RECOMMEND CLOSEST ITEM IN COMMUNITY

- Query community of search
- Get closest item in community by map reduce.

→ **Graph Method**

→ **Graph Method + Collaborative Filter**

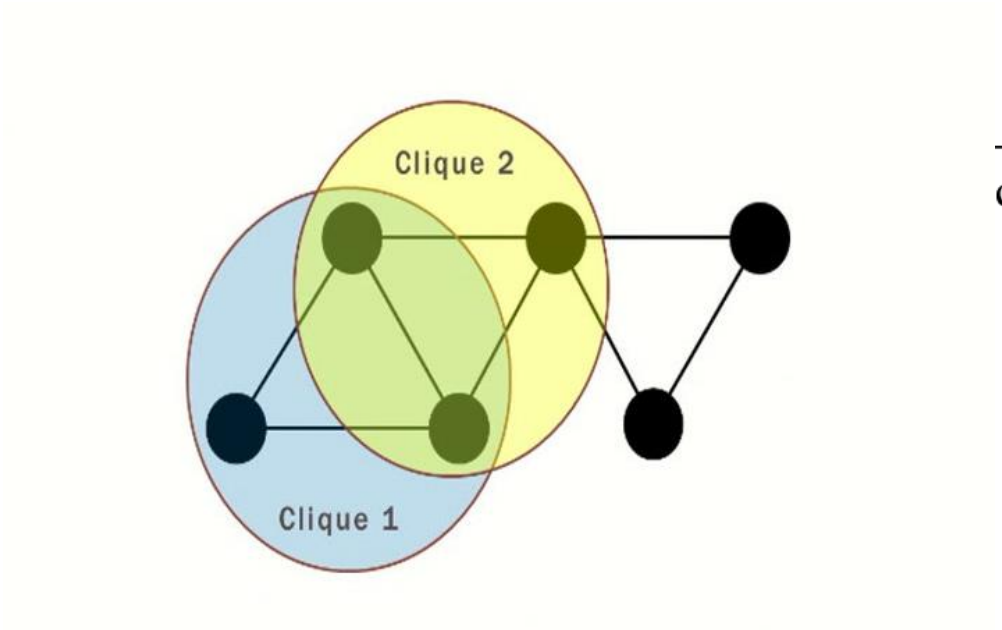
Summary

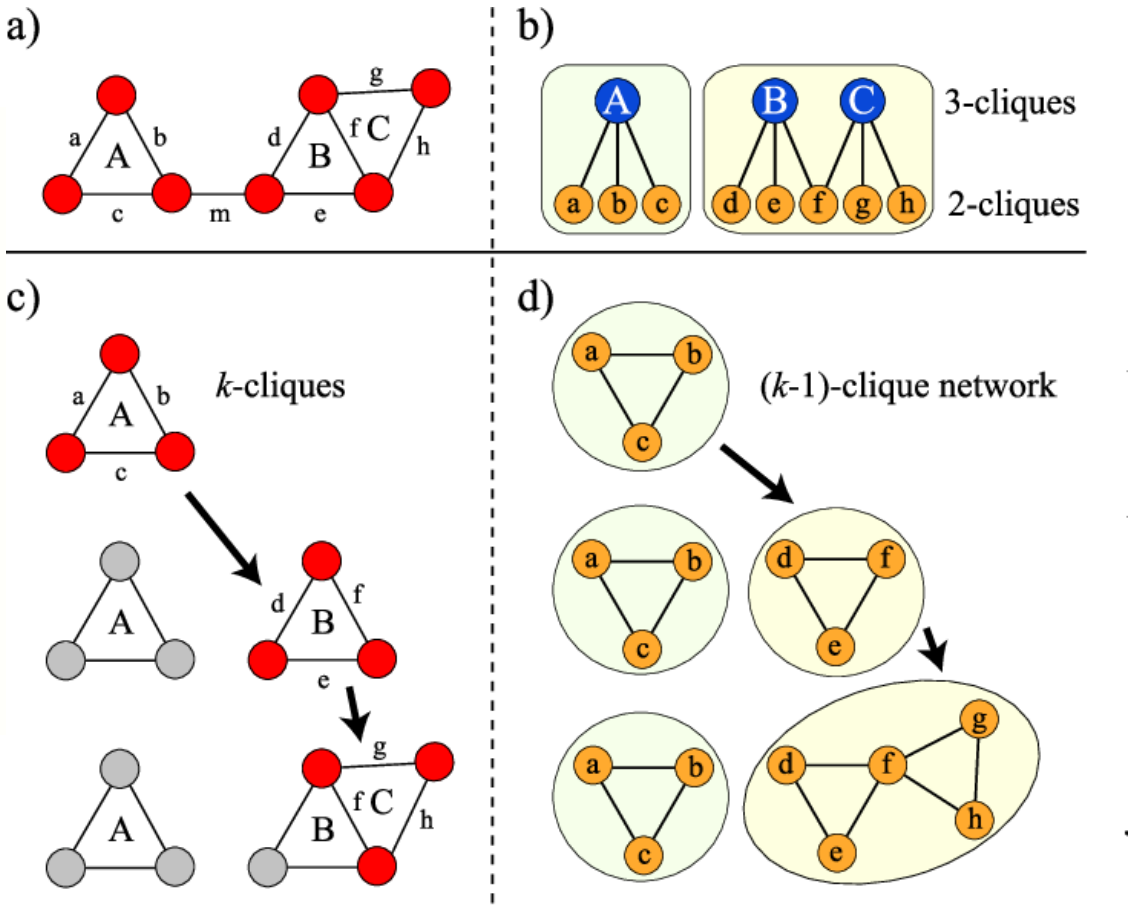
1. The data store needs to be cleaned. Furthermore, data collection method needs to be reviewed closely.
2. While the data stored in tables is quite large, the volume of true information points are is significantly smaller.
3. Most of the active users come from USA and EU.
4. The three models built perform well qualitatively. While tough to generate ,the quantitative analysis is pending on these models.
5. Deploying to production should be done after careful consideration of volume of data in future.

Future Scope

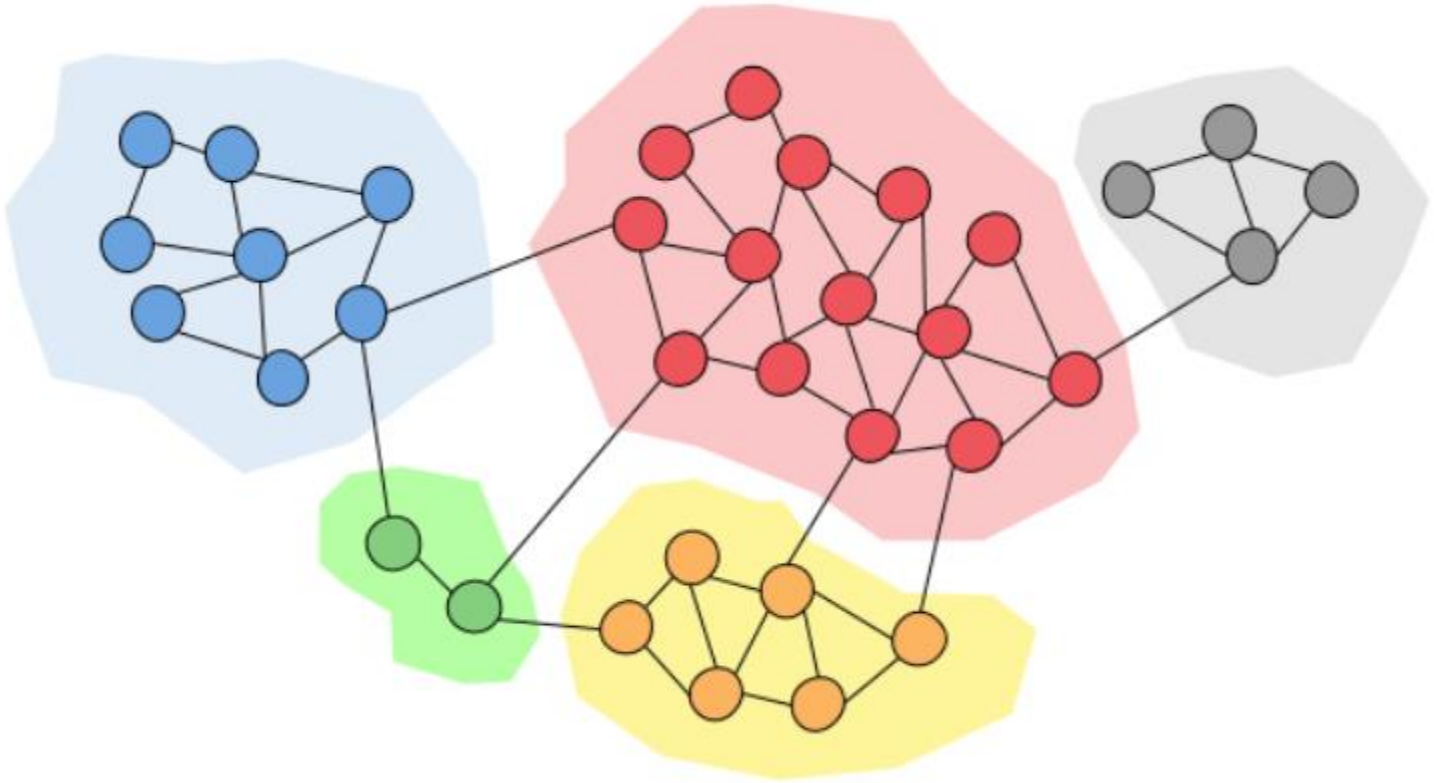
1. Successive user input for overlapping communities.
2. Ego networks of books after book data collection.
3. User behavior tracking and meta data generation (Clicks, page time, gender, education ..).
4. Deep learning-based recommender system.
5. Customized news-letter with rare/new picks.

Appendix – Clique Percolation method

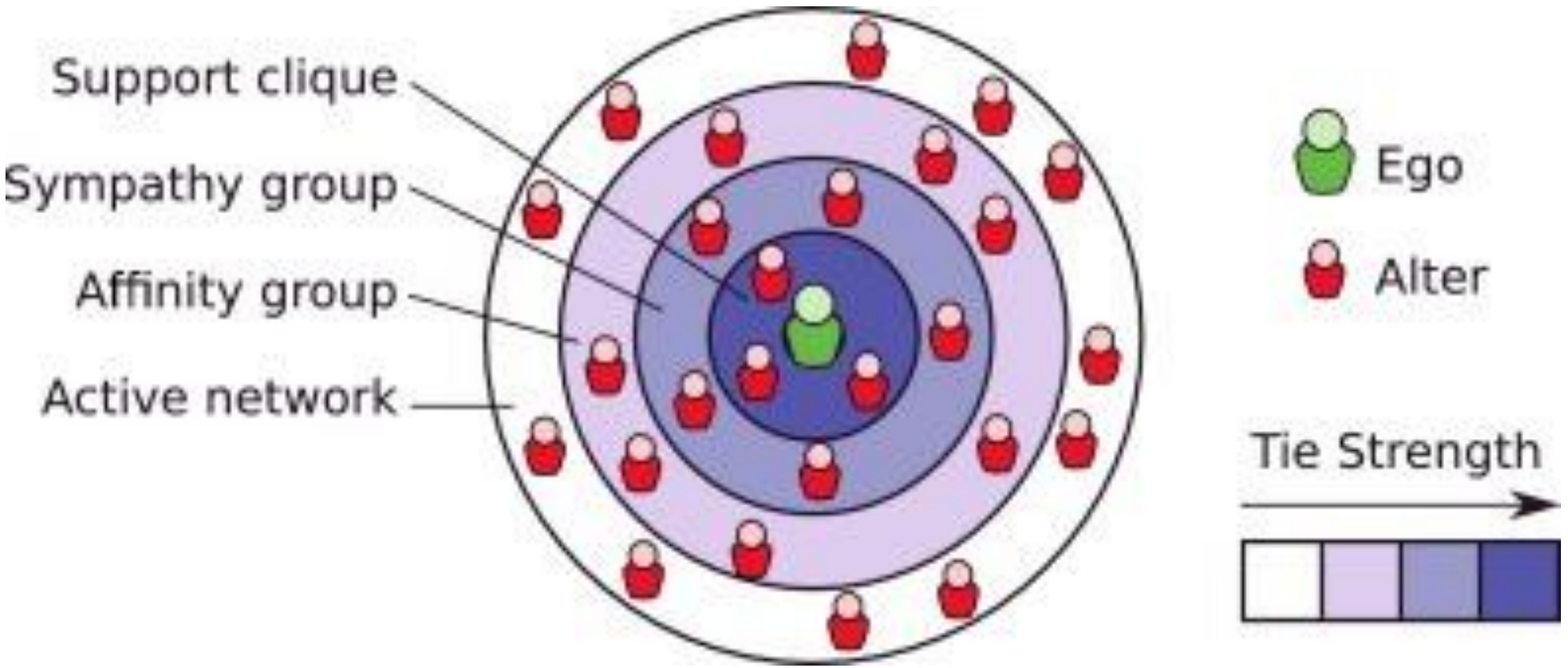




Appendix – Louvain Algorithm



Ego nets


Support clique
Sympathy group
Affinity group
Active network
Ego
Alter
Tie Strength