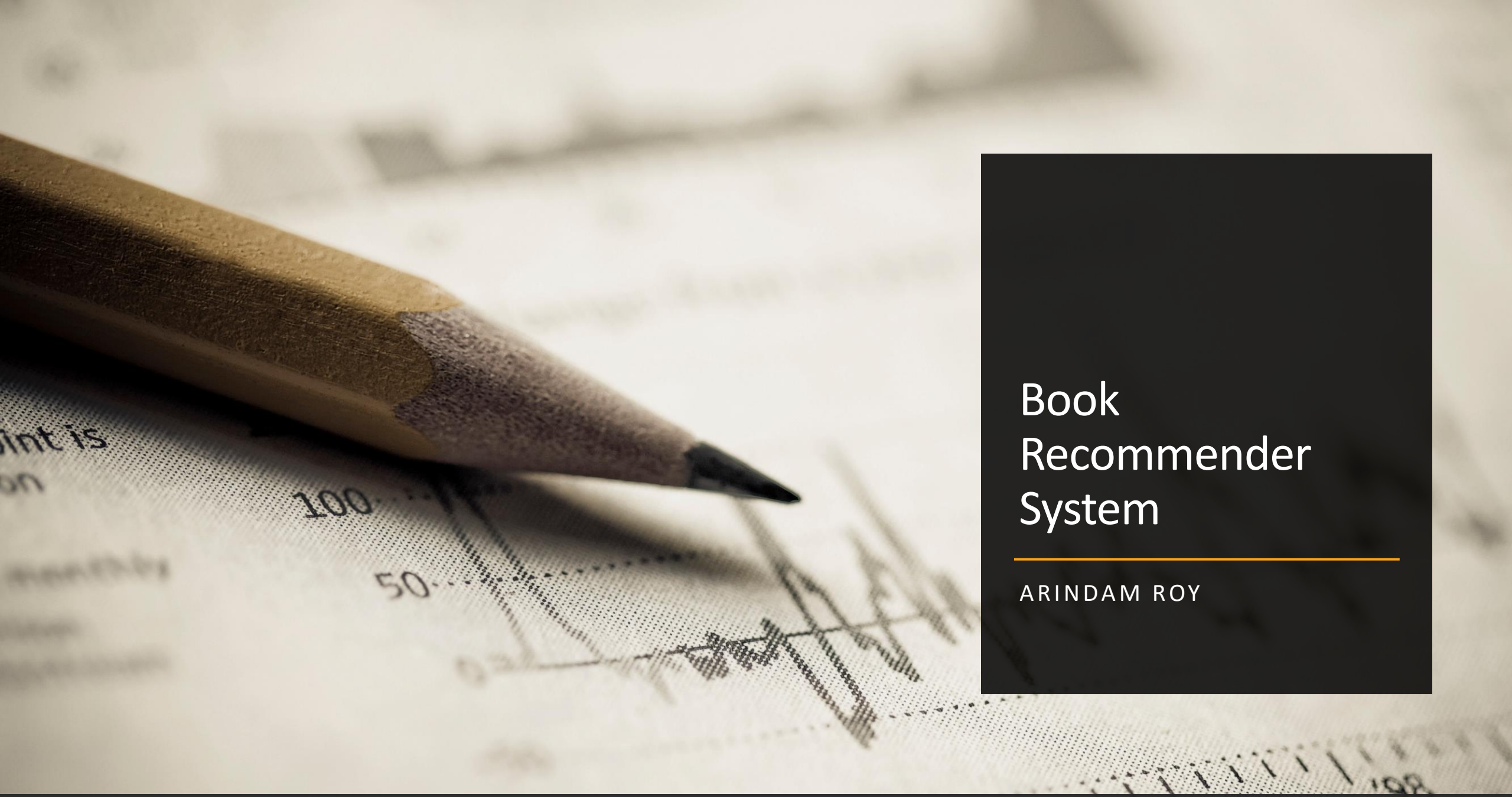


Book Recommender System

ARINDAM ROY

Introduction

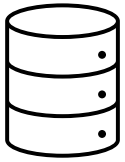
Problem Statement

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

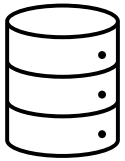
Data Description



Books

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

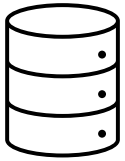
Columns: *isbn, book_title, book_author, year_of_publication, publisher,img_s, img_m, img_l, book_title_identifier*



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

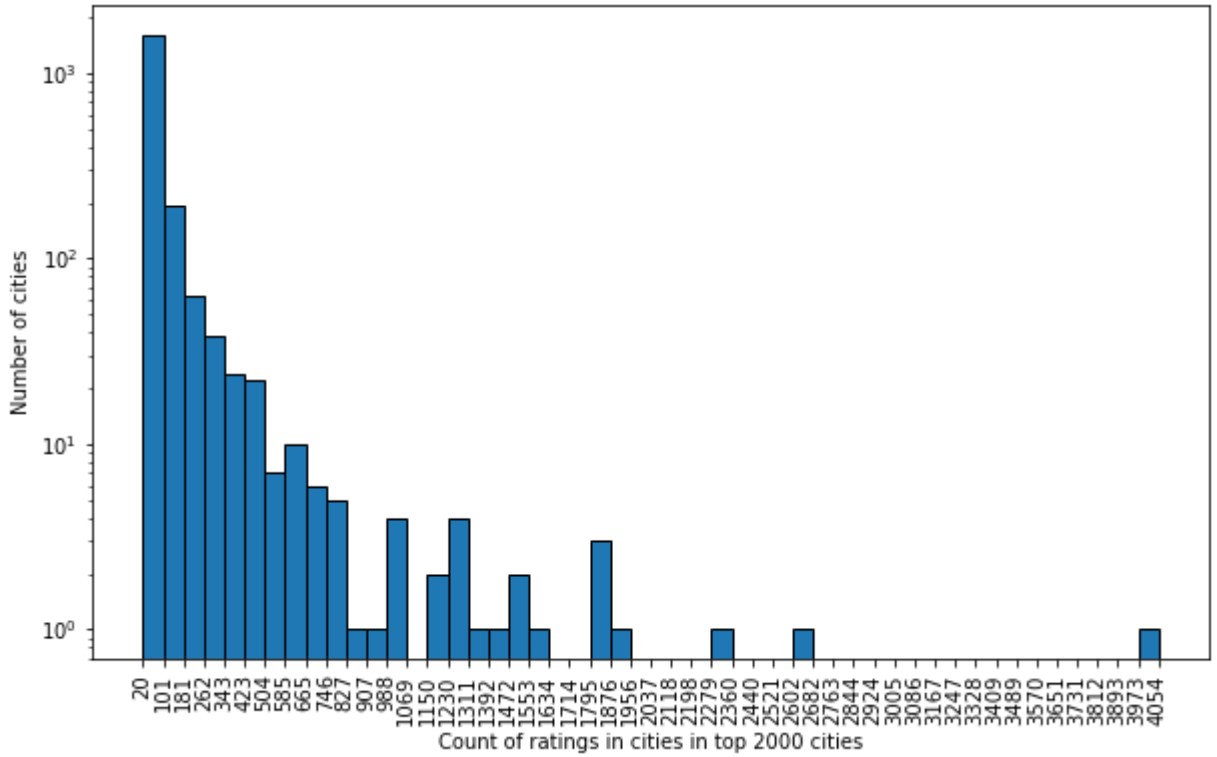
Statistic	Count
Total number of books	271,360
Total number of users	278,858
Total number of ratings	1,149,780
After Data Fixes	
Total number of books	242,505
Total number of users	278,858
Total number of ratings	1,144,516

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

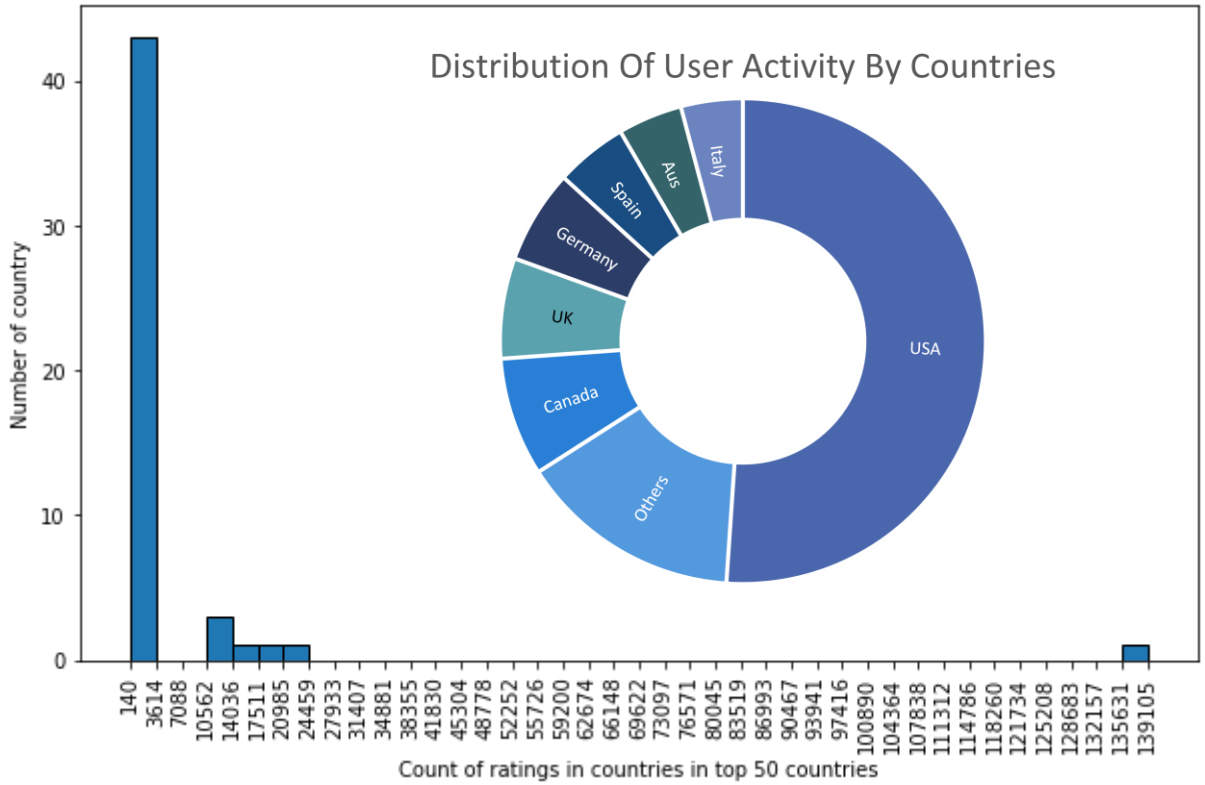


Observations

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

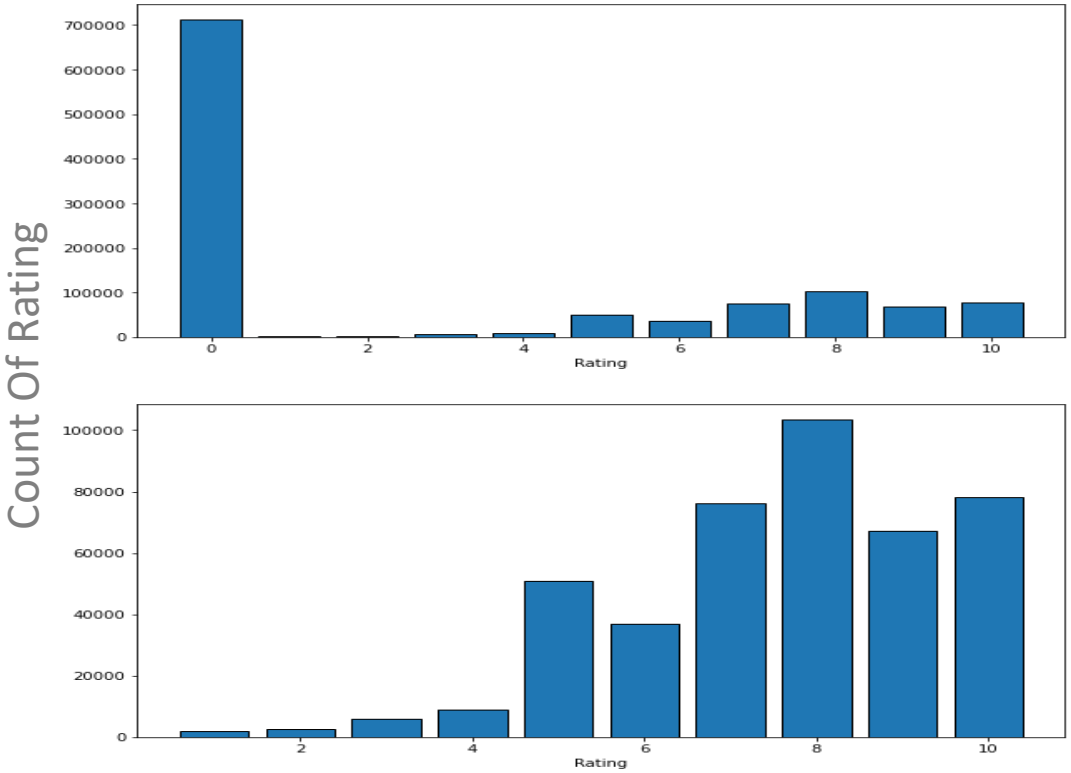
Ratings Statistic	Value
Total count of rated books	152,123
Maximum count of ratings on a book	505
Minimum count of ratings on a book	1
Median count of ratings on a book	1
Count of books with higher than or equal to 10 ratings	4,744
Count of books with higher than or equal to 5 ratings	11,952
Count of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

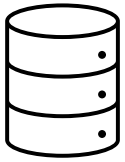


Observations

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.

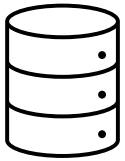
PROPOSED PREDICTIVE MODELS

Approaches Proposed



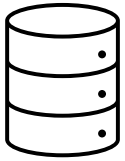
Cosine Similarity Collaborative Filtering

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



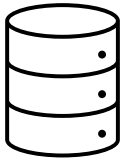
Graph Method (Social Network of Books)

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



Content Based Filtering

A LLM(BERT) is used to generate encoding on the titles. These encodings are used to predict the closest book.



Merged Approach

In the cluster identified, collaborative filtering is used with content-based filtering to recommend books.

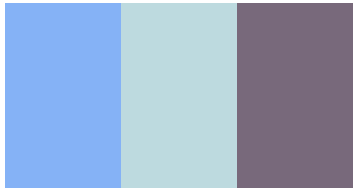
COLLABORATIVE APPROACHES

Cosine Similarity Collaborative Filtering - Technical Detail

STEPS FOR RECOMMENDING WITH COLABORITIVE FILTER

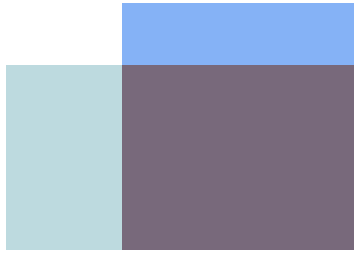
FILTER RATINGS DATA

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 Sim	

Core Principle Of Working

The collaborative filtering method works by finding cosine similarity 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 & Scalability** 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.	**Large Vector Problem** Since the vectors are root squared addition of ratings, books that are popular will create very large vectors in multiple dimensions. This in turn would result in these vectors showing up in similarity, even though this might not be the ideal solution.
	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

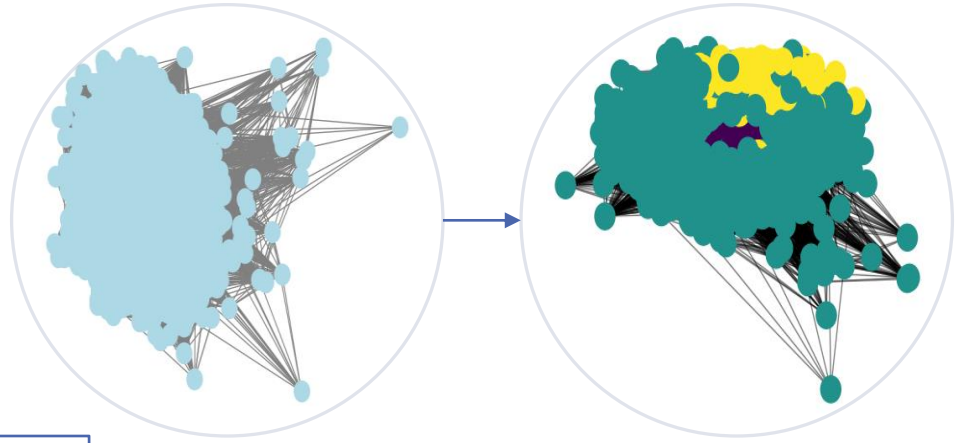
STEPS FOR RECOMMENDING WITH GRAPH METHOD

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 With Collaborative Filter- Technical Detail

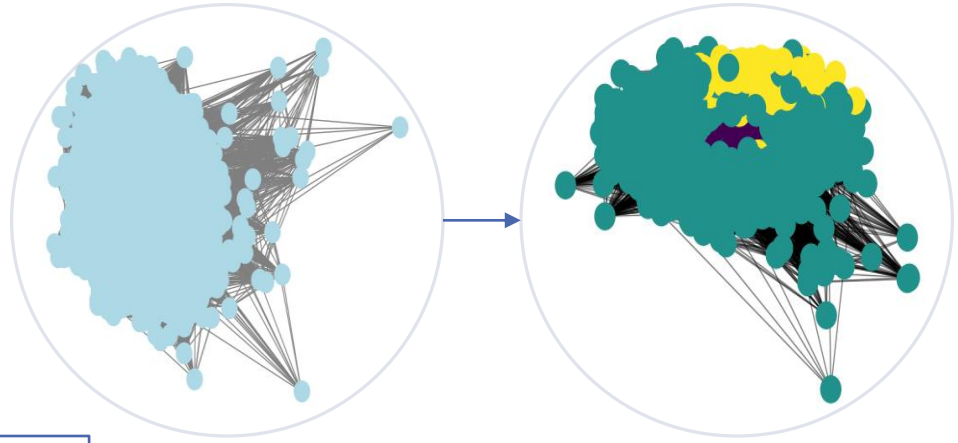
STEPS FOR RECOMMENDING WITH GRAPH METHOD

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 using the cosine similarity.

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 of computation and storage goes down.

Graph Method – 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.

CONTENT BASED APPROACH

Content Based Approach : Key Ideas

Current Challenges

- The collaborative approaches do not consider the content of the product.
- Dealing with out of distribution queries is not possible
- Additionally, we currently have extremely limited amount of text data to work with.

Why a LLM encoder?

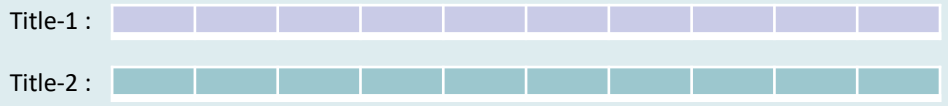
While a LLM encoder might take higher times to train and evaluate on, there are three key advantages:

1. Context: Since we have highly textual data, the context recognition is extremely critical
2. Out of bag: Such a model can help us deal with unseen things as it has past training to leverage.
3. Text Mining Scope: Addition to the existing text data will lead in much better results.

Outline of Proposed Solution

Step-1: Representation

Represent the available text titles as vectors of a high dimensional space using an encoder only LLM (BERT), keeping in consideration the literal and contextual meaning.



Step-2 : Query Representation

Any existing or non existing title can be represented in the same dimension

Step-3 : Scoring on query

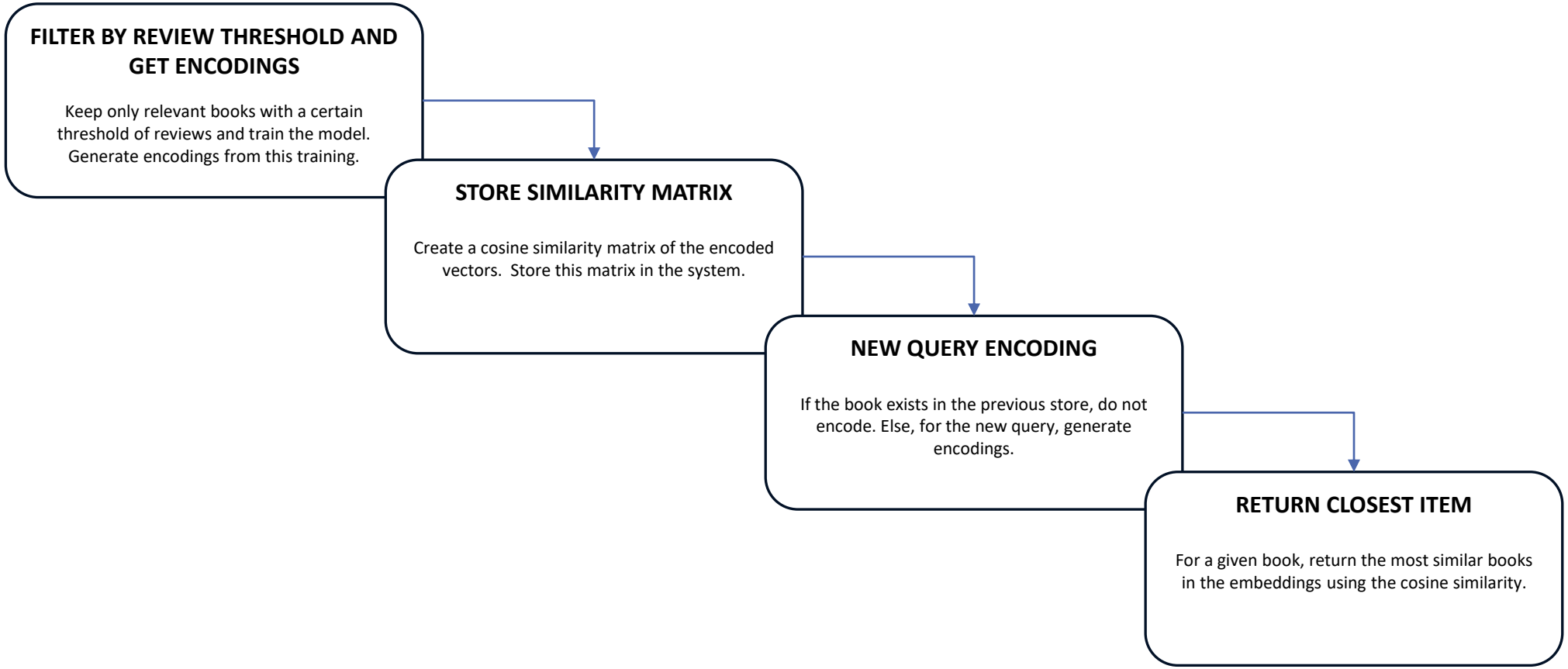
Similar to the collaborative filtering method with cosine similarity, we find the similarity between the different vector representations.

(Query, Title-1) : 0.8

(Query, Title-2) : 0.6

Content Based Recommendation- Technical Detail

STEPS FOR RECOMMENDING WITH LLM ENCODER



MERGED APPROACH

Merged Approach - Technical Detail

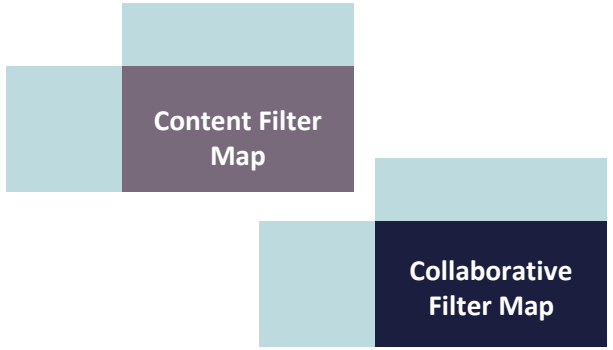
STEPS FOR RECOMMENDING WITH MERGED APPROACH

GET COMMUNIT(IES) FOR EACH BOOK

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

GENERATE CONTENT & COLABORATIVE COS SIMILARITY

Create cosine similarity matrices of the encoded vectors by both methods.



TAKE WEIGHTED AVERAGE OF BOTH MATRICES

Add the two matrices with optimal weights

RETURN CLOSEST ITEM

For a given book, return the most similar books in the embeddings using the cosine similarity on the joint filter map.

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 *N* users from the ratings table.
    2. Get top *K* predictions for *I* fraction of the positive rating books for the users.
    3. Find the intersection of the recommendations with the set of books the user has rated positively.
    4. Generate an average score.
    5. *N, K* and *I* are parameters that can be tuned to represent different user demographics, behaviors and confidence.
3. This is only an indicator metric. The numbers can be tuned or appended 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.

Legend

N: *Users: top, most frequent, highly positive, et cetera*

K: *Number of predictions to be generated*

I: *Fraction of the set of positive reviews to be predicted on*

Metric of performance: First Prediction-Top10

Method	Hit Rate	Average Test Set Length	Average Queried Filter Count	Average Set Length Returned	True Positive Rate	False Positive Rate
Cosine Similarity	**38.69**	78.9	30.5	28.1	**48.98**	51.02
Cosine Similarity With Graph	**24.55**	78.9	19.3	19.1	**44.23**	55.77
Content Based	**99.79**	78.9	78.7	73.6	**12.53**	87.47
Content Based With Cosine Similarity	**15.62**	78.9	12.4	12.4	**50.28**	49.72

Legend

N: (Top 10) *Number of users: top, most frequent, highly positive, et cetera*

K: (1) *Number of predictions to be generated*

I: (0.1) *Fraction of the set of positive reviews to be predicted on*

DEPLOYMENT TO PRODUCTION

Deployment Details: Collaborative Model

Collaborative Filtering

RUN MODEL AS A BATCH JOB

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

SAVE SIMILARITY SCORES IN A SPARSE TABLE

- Save the similarity matrix in a system capable of handling such data (HDFS, MongoDB, etc)

RUN QUERY ON USER INPUT

- Run map reduce enabled computation for every new user input to find the most similar 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 most popular in the community

RUN COLABORATIVE FILTER FOR EACH COMMUNITY(Batch-3)

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

RECOMMEND CLOSEST ITEM IN COMMUNITY

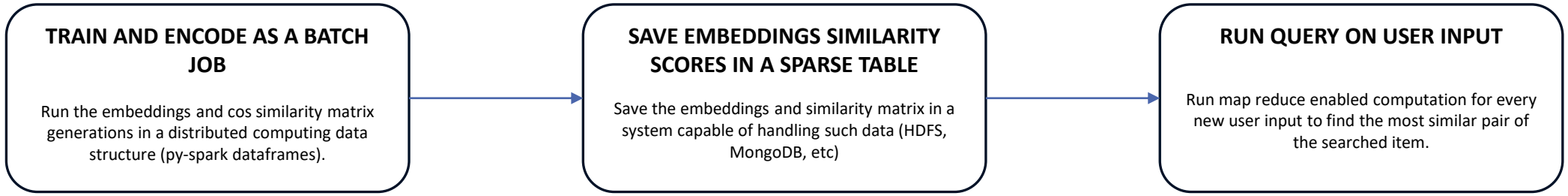
- Query community of search
- Get closest item in community by map reduce.

Graph Method

Graph Method + Collaborative Filter

Deployment Details: Content-based Model & Data Stream Build

Content Based Filtering



Data Streams

When traffic increases on the platform, it would be interesting to look into implementing data stream solutions. A few reasons for this are as follows:

- **Real-time Insights**: Data streams can catch onto trends, bugs, glitches et cetera faster than most methods
- **Cost Effective**: Instead of storing all data generated, we could look at storing only the relevant data items
- **Ad-Hoc Processing**: Models can be trained and appended in an ad hoc manner, creating the ability to optimize storage and performance.
- **De-Duplication**: Similar queries and responses can be generated without processing required with process like locality sensitive hashing. This would take of load from the computation.

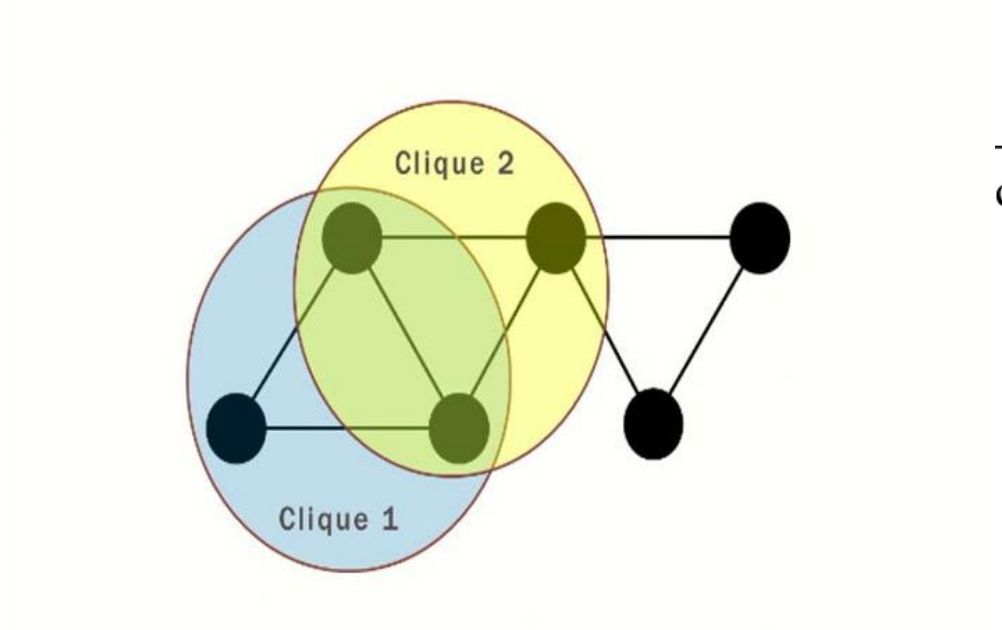
Summary

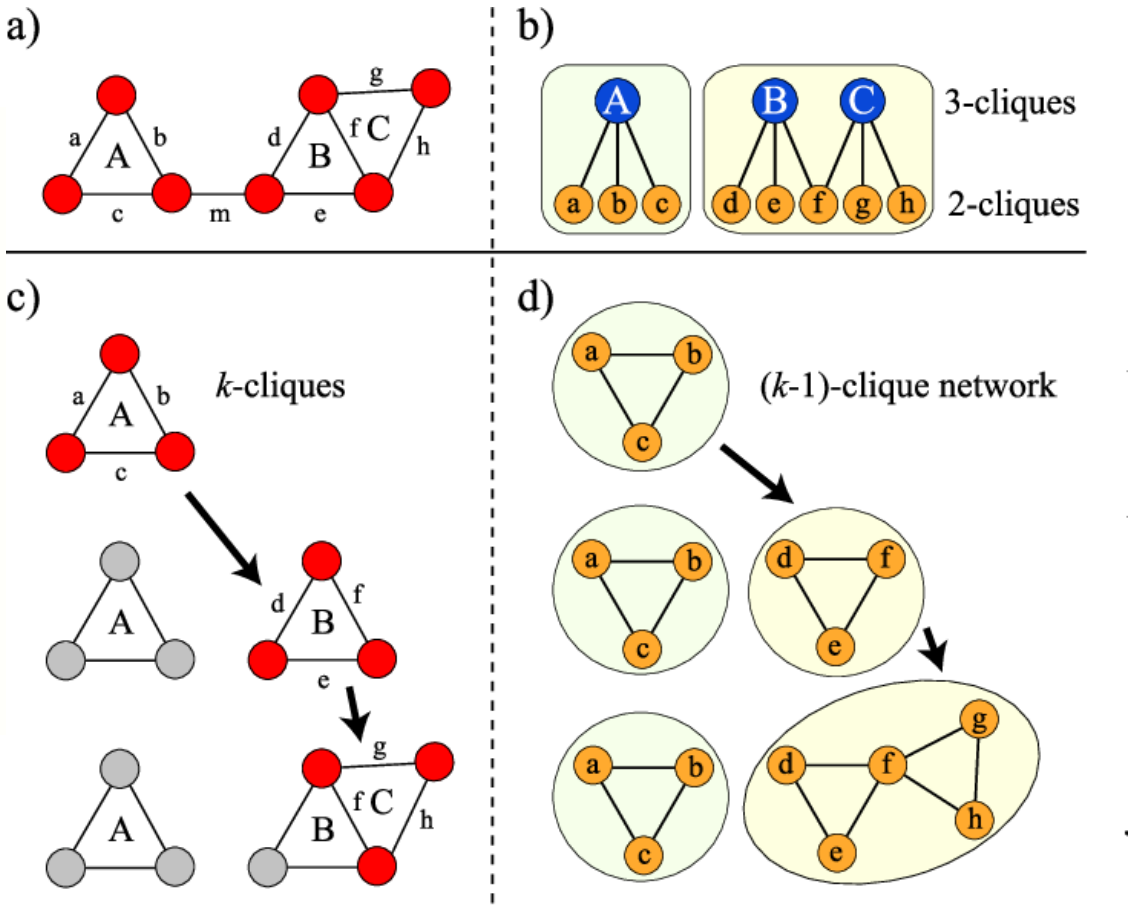
1. While the data received is quite large, the volume of true information points are significantly smaller. Data cleaning pipeline and the model selection parameters remove a large chunk of the data, implying that the data store needs to be cleaned. Furthermore, data collection and storage design methods need to be reviewed closely.
2. The four models built perform well qualitatively. In the quantitative test, we see a clear tradeoff between hit rate and precision of the models. On the current configurations, the content-based model has the highest hit rate while the joint model has the highest precision. The cosine similarity collaborative filter seems to be at the best tradeoff.
3. Deploying to production should be done after careful consideration of volume of data in future.

Future Scope

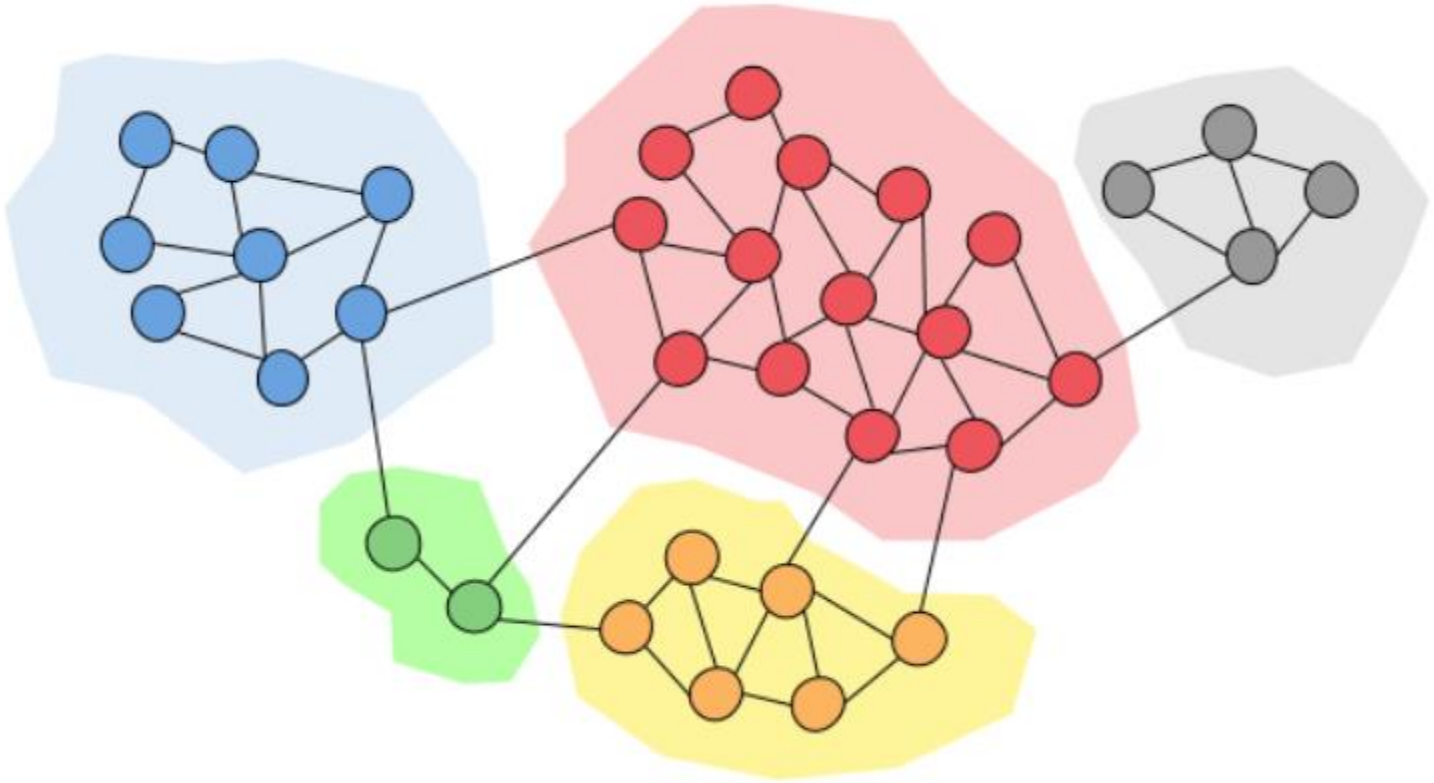
1. Extensive hyperparameter (grid search, randomized search, et cetera) tuning on each model for different performance metric configurations.
2. Data addition to the warehouse. This includes user data and product data. Few models will hugely benefit from textual data about the product.
3. Dimensionality reduction on the collaborative approaches to reduce computational complexity.
4. Image data extraction and user behavior relation with different methods like convolutional networks.
5. Directional referenced graph of books, Ego networks of books after book reference collection.

Appendix – Clique Percolation method

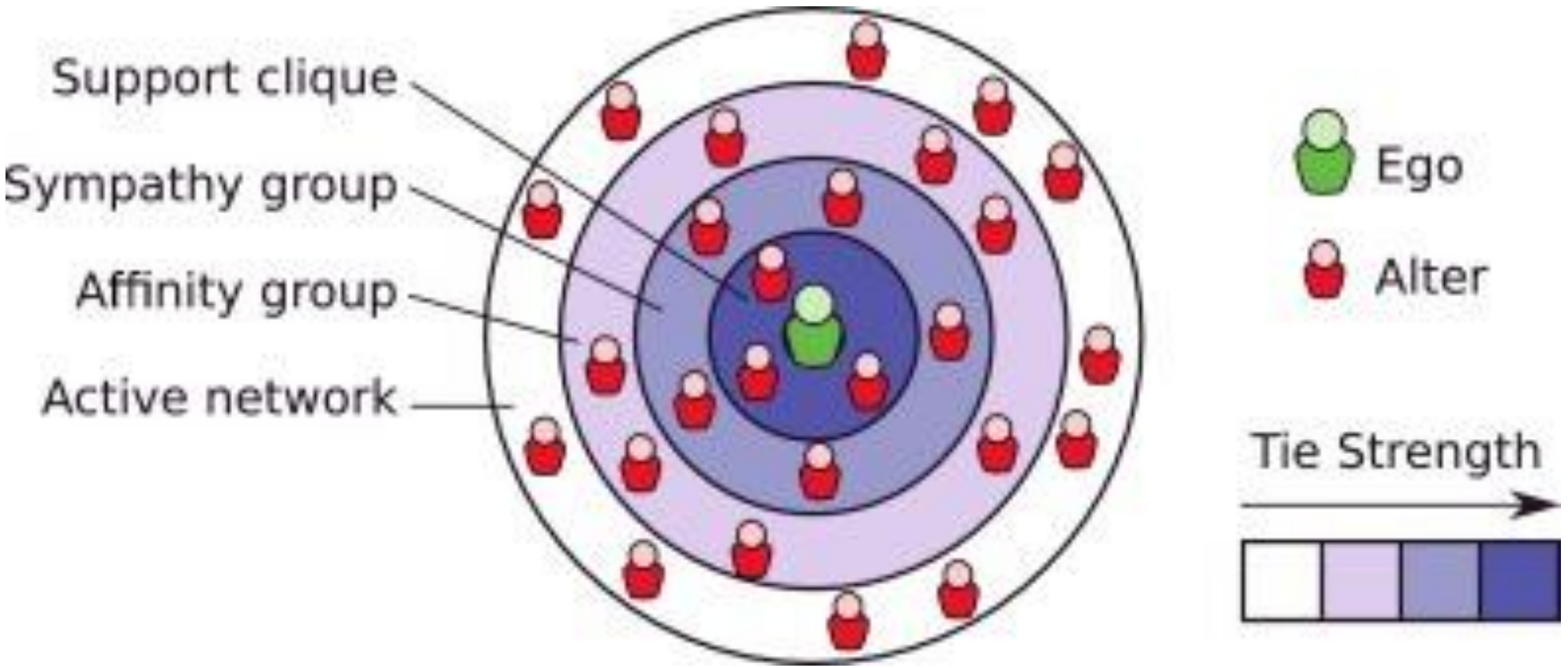




Appendix – Louvain Algorithm



Ego nets


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