### **Final Report**

**About Dataset** - Goodreads dataset (<https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home>)

The dataset contains - (1) meta-data of the books, (2) user-book interactions (users' public shelves) and (3) users' detailed book reviews. These datasets can be merged together by matching book/user/review ids. The entire dataset contains -

* 2,360,655 books (1,521,962 works, 400,390 book series, 829,529 authors);
* 876,145 users;
* 228,648,342 user-book interactions in users' shelves (include 112,131,203 reads and 104,551,549 ratings).

For the purpose of this project, I propose to filter the dataset by genre of book pertaining to ‘Poetry’, then filter out data pertaining to top 13 publishers as the entire original dataset is too huge.

**Business use case**

Goodreads is like a social network platform like Facebook, but it is for books. Here, users review books, become friends, share books and keep status on their reading habits etc. Goodreads derives its revenues by promoting book campaigns, where it works with major publishers to promote titles. The company’s data services mainly center on an open API for utilizing its book data on third-party sites. Data from Goodreads could be utilized to understand how book characteristics, writers and other factors perform among certain reader sets. It would be interesting to apply the machine learning concepts to perform this task. But graph networks draw more insights from the data that are hidden to regular machine learning techniques.

Goodreads business model is again just like Facebook. It makes revenue out of book promotion campaigns, ad campaigns, book convention etc. So, keeping its user base active is the most challenging but rewarding when it comes to driving its revenue high.

Analytics team of Goodreads could use graph analytics and build implicit book-user graphs –

* to visualize and analyze networks of interactions between users, books, and authors via their reading preferences
* to identify and recommend books to users by engineering network features
* to identify influential users and user groups on the Goodreads social network
* to predict future user – user connections

### **Analysis Plan**

To help users be active and sharing in Goodreads it is essential to understand how to suggest books to the users to keep them reading and keep them writing reviews? And, to conduct other profitable activities, it is important to understand if there are influencers, if there are any communites within the network or if there is a scope for the network to expand.

To analyze this, the idea is

* to build cypher queries and cypher actions to understand the network and use it for data exploration
* to build a recommendation engine based on collaborative filtering using Jaccard similarity
* to find influencers in the network using page rank and other centrality algorithm
* to find predict future network connections between users

### **Value Addition**

Knowing that Goodreads generates its revenue by running Book promotion campaigns, running ads on books, it's important that it understands its users and their preferences well. Few of the benefits that can be derived from the analysis -

* Target influencers in book promotion campaigns can drive the campaigns more towards success.
* Communities can be grown via book clubs,
* Publishers/Authors can conduct targeted book conventions,
* Book bundle sales can be promoted through users’ book preferences.

In a nutshell, these results from this analysis can help to build a rewarding relationship with the book companies and other customers to generate profit.

### **Graph Data Model**

Having just 2 node types hinders full understanding of user’s behavior. So, in order to expand the scope of understanding of user characteristics, few other node types have been added.

**Diagram

Description automatically generated**

### **Graph Projections**

The mono-partite graphs are as below -

SHARE\_BOOK - two users are said to be connected if they have read (shared) the same books. Number of shares becomes the weight here.

**Application

Description automatically generated with medium confidence**

Secondly, if two users have read and reviewed the same book before 2015 then they are related by relationship – SHARE\_BOOK\_EARLY.

If two users have read and reviewed the same book after 2015 then they are related by relationship – SHARE\_BOOK\_LATE.

**Database Setup Queries**

CREATE CONSTRAINT ON (n:Book) ASSERT n.book\_id, n.title is UNIQUE;

CREATE CONSTRAINT ON (n:Publisher) assert n.publisher is unique;

create constraint on (n:Author) assert n.author is unique;

create constraint on (n:User) assert n.review\_id is unique;

LOAD CSV WITH HEADERS FROM 'file:///ebooks.csv' AS row

CREATE (m:Book:Ebook {book\_id: row.book\_id, text\_reviews\_count: row.text\_reviews\_count, country\_code: row.country\_code, is\_ebook: row.is\_ebook, average\_rating: row.average\_rating,

description:row.description, format:row.format,link:row.link,

url:row.url,image\_url:row.image\_url,

ratings\_count:row.ratings\_count,work\_id:row.work\_id,title:row.title,

title\_without\_series:row.title\_without\_series });

LOAD CSV WITH HEADERS FROM 'file:///not\_ebooks.csv' AS row

CREATE (m:Book:not\_Ebook {book\_id: row.book\_id, text\_reviews\_count: row.text\_reviews\_count, country\_code: row.country\_code, is\_ebook: row.is\_ebook, average\_rating: row.average\_rating,

description:row.description, format:row.format,link:row.link,

url:row.url,image\_url:row.image\_url,

ratings\_count:row.ratings\_count,work\_id:row.work\_id,title:row.title,

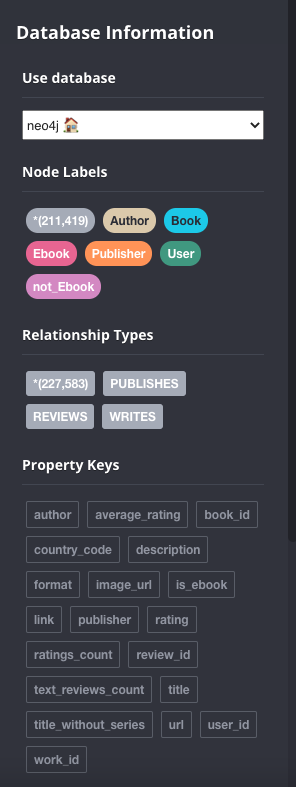
title\_without\_series:row.title\_without\_series });

LOAD CSV WITH HEADERS FROM 'file:///authors.csv' AS row MERGE(n:Author{author:row.author}) WITH row, n MATCH(t:Book {book\_id:row.book\_id}) MERGE (n) - [:WRITES] -> (t);

LOAD CSV WITH HEADERS FROM 'file:///publisher.csv' AS row MERGE(n:Publisher{publisher:row.publisher}) WITH row, n MATCH(t:Book{book\_id:row.book\_id}) MERGE (n) - [:PUBLISHES] -> (t);

LOAD CSV WITH HEADERS FROM 'file:///reviews.csv' AS row MERGE(n:User{user\_id:row.user\_id, review\_id:row.review\_id, rating:row.rating}) WITH row, n MATCH(t:Book{book\_id:row.book\_id}) MERGE (n) - [:REVIEWS] -> (t);

**Graph Model in Neo4j**

****

**Chart

Description automatically generated with medium confidence**

**CYPHER Queries**

1. **Find the books that has the maximum number of reviews**

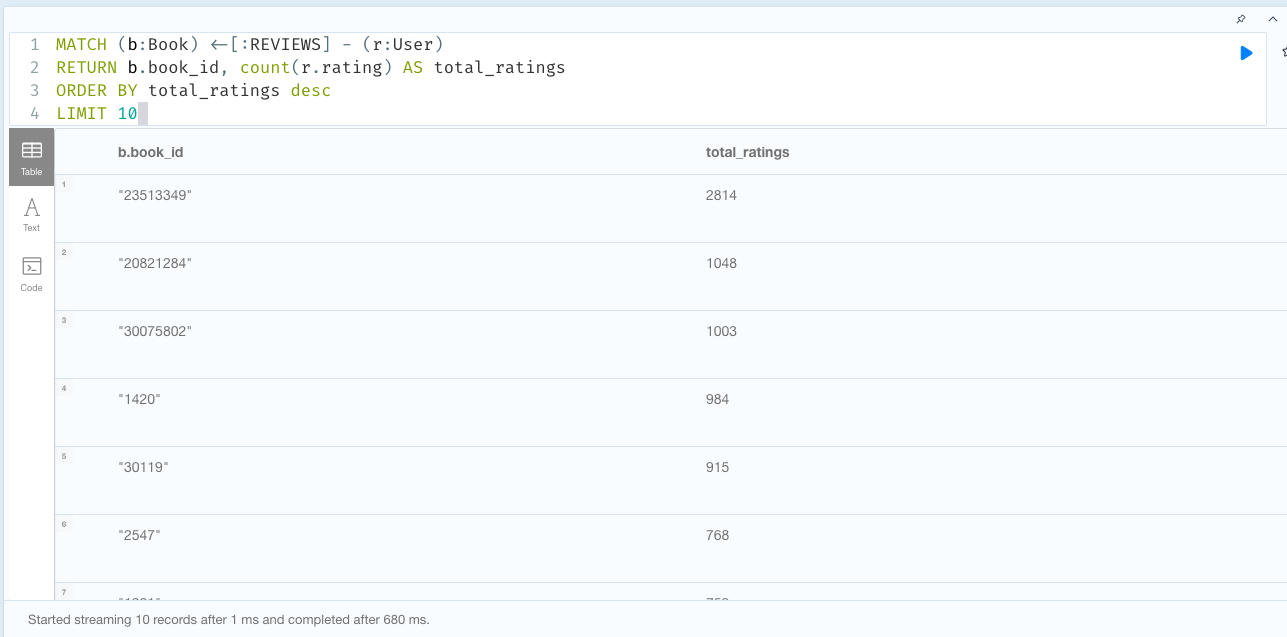
MATCH (b:Book) <-[:REVIEWS] - (r:User)

RETURN b.book\_id, count(r.rating) AS total\_ratings

ORDER BY total\_ratings desc

LIMIT 10

Popular books tend to receive more reviews. Whether it is popular for negative or positive reasons, a popular book receives more reviews. It is important to know the top few popular books to understand user characteristics related to those popular books.



1. **Find the top 10 users who have read the most number of books of different authors.**

MATCH (p1:User)-[:REVIEWS]->(b:Book)<-[:WRITES]-(a:Author)

RETURN p1.user\_id AS user, count(distinct a.author) AS authors\_read

ORDER BY authors\_read DESC

LIMIT 10

Users who have read more books are important when it comes to finding the relationship between users' preference to authors. Users who have read different authors could also be related to other users through the books. So, understanding the relationship between users to users through books will help Goodreads better tune their recommendations.



1. **Find the top 10 users connected to users through books.**

MATCH (p1:User)-[:REVIEWS]->(b:Book)<-[:REVIEWS]-(p2:User)

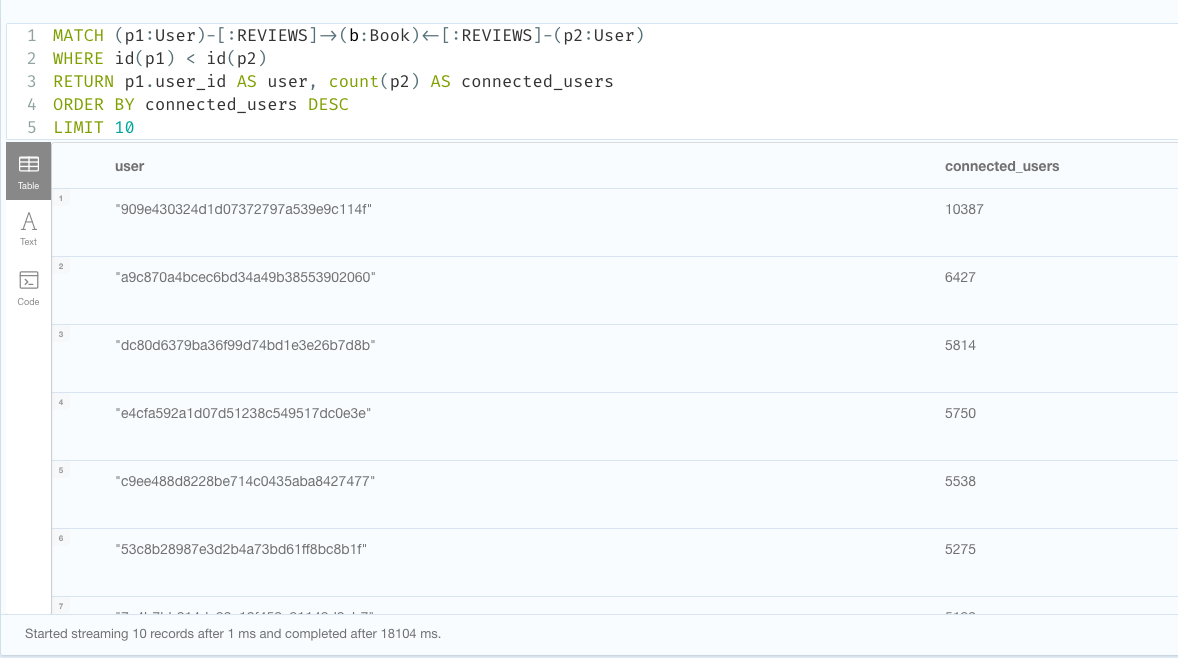
WHERE id(p1) < id(p2)

RETURN p1.user\_id AS user, count(p2) AS connected\_users

ORDER BY connected\_users DESC

LIMIT 10

Users with maximum connection to other users through books tend to be the influencers. So, targeting these users through campaigns could yield better results to the campaign. Knowing these top few user nodes that are maximum connected to other users is important to the use case to find influencers.



**Projections and Algorithms**

Before jumping to identify influencers in the graph, it is necessary to first define and lay out the assumptions.

So, starting with a definition for influencer -

*A Social Media Influencer is a user on social media who has established credibility in a specific industry. A social media influencer has access to a large audience and can persuade others by virtue of their authenticity and reach.*

Next step would be to create a projection such that it captures the relationship between users based on the book they have read. So, user 1 is said to be related to user 2 if they both have read the same book.

As this dataset does not provide any information on friendship or follower-followee relationship between users, for the sake of convenience it is assumed that people who have reviewed the same book are friends.

Below is the cypher query to create the projection discussed.

**Projection 1 -**

CALL gds.graph.create.cypher(

'user-to-user-relationship',

'MATCH (p:User) RETURN id(p) AS id',

'MATCH (p1:User)-[:REVIEWS]->(b:Book)<-[:REVIEWS]-(p2:User) WHERE id(p1) < id(p2) RETURN id(p1) AS source, id(p2) AS target')

**Algorithm 1 - PageRank**

From the projection, the idea is to find people who are influencers. But, to identify influencers, what makes a user influential? One can say -

* A book is relevant for him/her if other similar people liked it.
* A person is similar to him/her if they like books that are relevant to him/her.

So, to find these influencers, centrality algorithms can be used.

But if there are two people who are equally similar to a given user and say if both the people like the book this user likes. But if one of these people likes every single book in the market, and the other one likes only a handful of books, the set of books from the second person is much more informative about the first person’s interests. Which is why it is more important to use PageRank instead of other centrality approaches, as the [PageRank algorithm](https://neo4j.com/docs/graph-algorithms/current/algorithms/page-rank/) measures the transitive influence or connectivity of nodes.

If Goodreads wants to post some content or do some marketing campaign on books and have it spread across a large number of users/accounts, the users that rank highest for PageRank would be the best place to post that content.

**Cypher query-**

CALL gds.pageRank.stream('user-to-user-relationship')

YIELD nodeId, score

RETURN gds.util.asNode(nodeId).user\_id AS user, score

ORDER BY score DESC, user ASC

****

**Algorithm 2 - Betweenness Centrality**

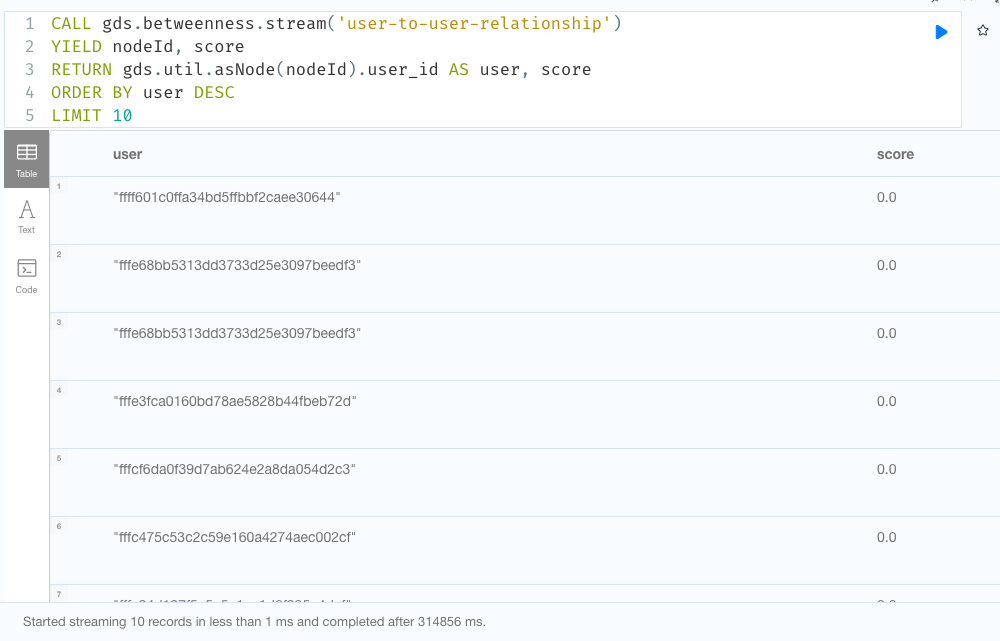
PageRank can give us influencers. But there can be a disjoint set of groups that have users with disjoint interests, connected by very few users that connect groups. These users are important to know of to maximize the spread of book campaign success. Such users can be found using betweenness centrality as betweenness centrality is a way of detecting the amount of influence a node has over the flow of information in a graph. So, this algorithm can be used to find nodes that serve as a bridge from one part of a graph to another.

CALL gds.betweenness.stream('user-to-user-relationship')

YIELD nodeId, score

RETURN gds.util.asNode(nodeId).user\_id AS user, score

ORDER BY user ASC

****

**Algorithm 3 - Louvain Modularity**

How are the users similar? Are there any groups in between users? Answers to such questions might be important to recommend books or to make a book launch/marketing campaign successful. Users in the same community might react the same way as the other user in the group towards a book. To know and capture this user's reaction towards the book would be necessary and reasonable before launching a book campaign as part of market research.

To do this, community detection algorithms can be used to evaluate how groups of nodes (users) are clustered or partitioned, as well as their tendency to strengthen or break apart. The Louvain Modularity algorithm detects communities in networks, based on maximizing a modularity score, where the modularity quantifies the quality of an assignment of nodes to communities. This means evaluating how much more densely connected the nodes within a community are, compared to how connected they would be in a random network.

This algorithm can be used to find sub communities in the larger graph community.

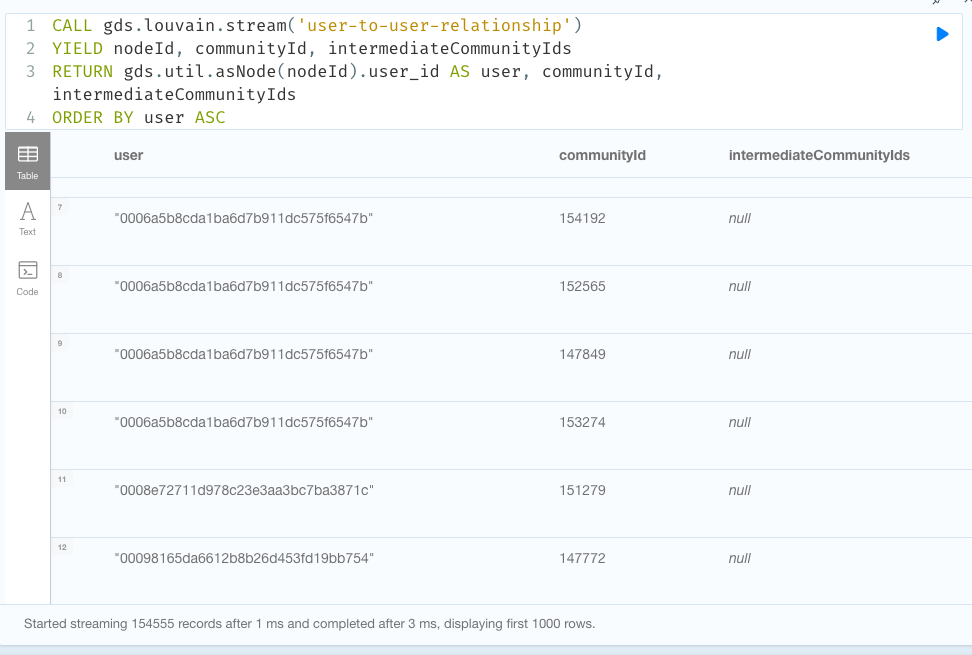
**CYPHER query -**

CALL gds.louvain.stream('user-to-user-relationship')

YIELD nodeId, communityId, intermediateCommunityIds

RETURN gds.util.asNode(nodeId).user\_id AS user, communityId, intermediateCommunityIds

ORDER BY user ASC

****

**Projection 2 -**

Projection 1 connects users to users folding on the edge- book. Similar projection can be done by folding on Authors. It is generally true in the book readers community that readers are divided by authors they follow, and fans/followers of the same author are so closely knit. To understand the influencers and their characteristics based on Authors, in order to make recommendations based on authors, we can use this projection.

CALL gds.graph.create.cypher(

'user-to-author-relationship',

'MATCH (p:User) RETURN id(p) AS id',

'MATCH (p1:User)-[:REVIEWS]->(b1:Book)<-[:WRITES]-(a:Author)-[:WRITES]->(b2:Book)<-[:REVIEWS]-(p2:User) WHERE id(p1) < id(p2) RETURN id(p1) AS source , id(p2) AS target')

**Algorithm 4 - Closeness Centrality**

Closeness centrality is a way of detecting nodes that are able to spread information very efficiently through a graph. The closeness centrality of a node measures its average farness (inverse distance) to all other nodes. Nodes with a high closeness score have the shortest distances to all other nodes.

So, this algorithm can be used to find users that have a high closeness centrality score to influence the fan followers of that author. It would come handy when the book campaigns or book releases are planned. During book release the influencers with high scores from this algorithm can be specifically identified and used to propagate the information within the community. As the communities are tightly knit and well trusted among its followers, it would serve as a good tool to reach each and every member of the community.

**CYPHER query**

CALL gds.alpha.closeness.stream('user-to-author-relationship')

YIELD nodeId, centrality

RETURN gds.util.asNode(nodeId).user\_id AS user, centrality

ORDER BY user DESC

LIMIT 10



**CYPHER ACTIONS**

1. **Find top 10 books of a publisher based on user ratings**

CYPHER ACTION QUERY –

MATCH (p:Publisher {publisher:$publisher\_name}) - [r:PUBLISHES] -> (b:Book) <- [w:REVIEWS] - (u:User)

WITH p,r,b,sum(u.rating) AS total\_rating

ORDER BY total\_rating DESC

RETURN p,b,r

LIMIT 10

Graphical user interface, text, application, Teams

Description automatically generated

Diagram

Description automatically generated

1. **Find the author and publishers of book with title name**

CYPHER ACTIONS QUERY –

MATCH (a:Author) - [r1:WRITES] -> (b:Book{title:$title}) <-[r2:PUBLISHES]- (p:Publisher)

RETURN a,r1,b,r2,p

Graphical user interface, text, application

Description automatically generated

Chart

Description automatically generated with low confidence

**GRAPH VISUALISATIONS**

1. **Shortest Path Between the reviews from extremes of community IDs –**

After running Louvain Community Detection algorithm in the ‘user-book-user’ projection, which is created by folding on the edge book, the community Ids of the nodes are stored in community variable in User node. Now it can be used to see how far the users are from the first and last community. This helps to visualize as many nodes are in between the extremities.

In this below example, the first community is 10 and the last community is 154554. This graph can help understand which author or book or common node that connects the extremes. So, in order to reach both the users for a marketing campaign the idea would be target the common node that connects both the users.

A picture containing chart

Description automatically generated

1. **User nodes with highest page rank –**

After running pagerank algorithm on the projection ‘user-book-user’ folded on edge book, pagerank is stored in ‘pagerank’ variable on User node. It is important to understand how the users with high page rank are connected to target only those influencers with high page rank. So, when the users with high page rank are visualized, it can be noticed that a single author connects all these users. So, if these users were targeted, the marketing campaign can better increase its probability of success.

Background pattern

Description automatically generated

1. **User nodes with highest and lowest degree of centrality**

Users with highest degree of centrality are connected to maximum of users and users with the lowest degree of centrality are connected to very few users. It is hard to reach to people with the lowest degree of centrality in general. In order to make a book marketing campaign a big success, it is essential to reach out to people with even low degree of centrality. So just like it is done in the first example, the shortest distance between users with high and low degree of centrality would help the publishers to reach out to and influence people who are mostly disconnect to the rest.

In this example user ‘20fd7b31a6445c3959594b61d3659274’ has the highest degree of centrality - 2813.0 and user ‘7074b9bd7e86e3eb85dfe2d5d4a85f8e’ has the lowest degree of centrality 0

Chart

Description automatically generated with low confidence

**RECOMMENDATION SYSTEM**

The two major recommendation approaches are content filtering and collaborative filtering. For this project the best fit was collaborative filtering.

Collaborative filtering approaches mainly rely on the user behavior data. For example, if say Alice and Bob loves the book ‘Lord of the Rings (LOR)’, which was liked by both of them, respectively. Based on the records, it can be inferred that these two users may share similar preferences. Now considering that Bob loves ‘The Hobbit’, we can expect similar behavior from Alice, and recommend ‘The Hobbit’ to Alice. K-nearest neighbors ([KNN](https://docs.tigergraph.com/v/2.6/dev/gsql-examples/common-applications#example-1-collaborative-filtering)) is a typical collaborative filtering approach.

For finding the similarity between the users, Jaccard similarity was used. Then, used k-nearest neighbors’ approach to find the users that are close to the user that needs a recommendation. Then, the top few books were filtered from the list of the books read by the neighbors.

Users like good recommendations and good recommendations can help them be active on the network, which can be profitable in generating revenue.

**CYPHER QUERY–**

MATCH (u1:User {user\_id:$uid})-[:REVIEWS]->(Book1)

WITH u1, collect(id(Book1)) AS u1Books

MATCH (u2:User)-[:REVIEWS]->(Book2) WHERE u1 <> u2

WITH u1 as us1, u2 as us2, (gds.alpha.similarity.jaccard(u1Books, collect(id(Book2))))\*100 AS similarity

ORDER BY similarity DESC, us2.user\_id

WITH us1, COLLECT(us2)[0..$k] as neighbours

WHERE size(neighbours) = $k

UNWIND neighbours as neighbour

WITH us1, neighbour

MATCH (neighbour)-[:REVIEWS]->(b:Book)

WHERE not (us1)-[:REVIEWS]->(b:Book)

WITH us1,b,COUNT(DISTINCT neighbour) as cnt

ORDER BY us1.user\_id, cnt DESC

RETURN us1.user\_id as user, COLLECT(b.title)[0..$n] as recos

(Please refer to Python code for implementation in - RecoSystem&LinkPrediction.ipynb)

**LINK PREDICTION WITH MACHINE LEARNING**

Can future network expansions be predicted? There can be several subnetworks in Goodreads, but can it be predicted if a particular sub-network or community is going to expand in the future? Knowing this can help Goodreads utilize this information to conduct community specific book events in the future and plan for it ahead of time or utilize it for other profitable activities like book bundle sales.

Link prediction although is a straightforward technique, it involves a number of parts that include machine learning and use of link prediction algorithms.

There are several link prediction algorithms, but the effectiveness and their prediction power depend on the network. For this project, Adamic Adar, Common neighbors, and total neighbors’ algorithms were used and compared.

Below were the steps followed -

* To start with, the dataset was divided into training and test dataset. Any pair of user nodes that have both reviewed the book before 2015 have the relationship SHARE\_BOOK\_EARLY and any pair of user nodes that have both reviewed the book after 2015 have the relationship SHARE\_BOOK\_LATE.
* Any pair of nodes that share a relationship has a LABEL 1 and any pair that does not have a relationship has LABEL 0
* Then link prediction algorithms were used on training and testing dataset nodes to find the closeness value between the pairs.
* Machine learning algorithm – Random Forest – was trained on the training dataset, such that the closeness parameter from link prediction algorithm is the training variable and the labels were the classes to be predicted.
* Trained Random Forest model was used to predict the labels in the test dataset.

**CYPHER QUERY used –**

UNWIND $pairs AS pair

MATCH (u1) WHERE id(u1) = pair.node1

MATCH (u2) WHERE id(u2) = pair.node2

RETURN pair.node1 AS node1, pair.node2 AS node2, gds.alpha.linkprediction.commonNeighbors(u1,u2,{relationshipQuery:"SHARE\_BOOK\_LATE"}) AS CN, gds.alpha.linkprediction.adamicAdar(u1, u2, {relationshipQuery:"SHARE\_BOOK\_LATE"}) AS aa, gds.alpha.linkprediction.totalNeighbors(u1, u2, {relationshipQuery:"SHARE\_BOOK\_LATE"}) AS tn

(Please refer to Python code for implementation in - RecoSystem&LinkPrediction.ipynb)

**CONCLUSION**

In social media platforms like Goodreads having customers active can be used for a great deal of profitable ventures. But at the same time, it is also a very challenging task. So, to tackle this problem, few solutions like building a recommendation engine, finding influencers in the network, finding communities in the network, predicting future user-user connections were suggested. These were successfully implemented for a small network but can be scaled to a larger scale network to accommodate growing network in the future.

The suggested solutions can be leveraged for many profitable activities to earn revenue. The link prediction solution can be used to organize book-based conventions, make book club suggestions, predict community expansions that can be leveraged for book bundle sales etc., Community detection can be used for community testing to gauge the interest on a book before its full release, recommendation systems to effectively recommend books to users based on their reading preferences, and if found influencers can be used as targets for book campaigns and ad campaigns.