Goodreads

Books

Network Science project

by Riccardo Sturla 13457A

Overview

- Introduction
- Objectives
- The Network
- Analysis
- Node2Vec and RecSys
- References

Introduction

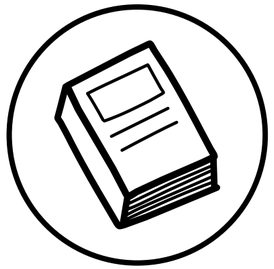
GOODREADS

- Goodreads is an American social cataloging website and a subsidiary of Amazon that allows individuals to search its database of books, annotations, quotes, and reviews.

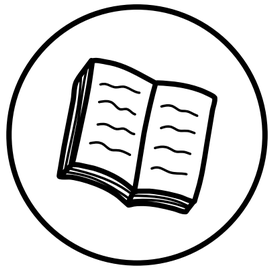
THE DATA

- Books and ratings data have been originally obtained through the Goodreads API, while I downloaded them from Kaggle.com.

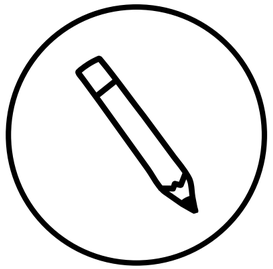
Project objectives



Create a network using the Goodreads database of books and ratings.



Analyse the network composition and its main features.



Create a **recommendation system** using the Node2Vec algorithm.

Creating the Network

Nodes

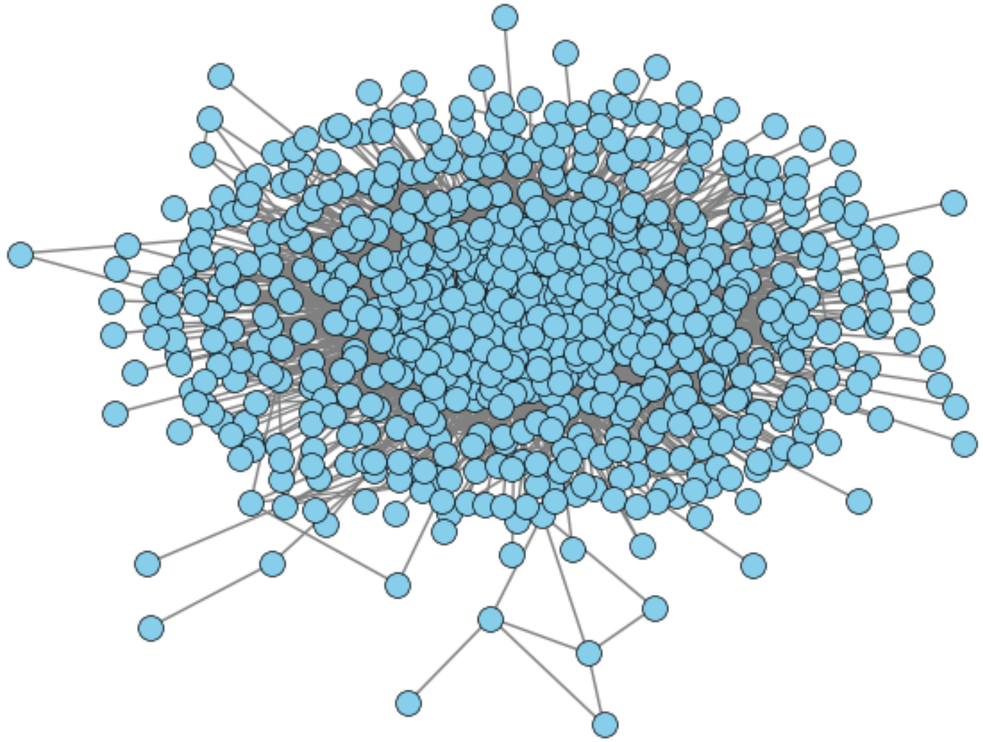
- First, we select only books rated 4 or 5 out of 5 by each user: that means that they have been liked.
- We compute a score representing the number of times two books are liked by the same user.
- Only books with a score bigger than 5 are taking into account.

Links

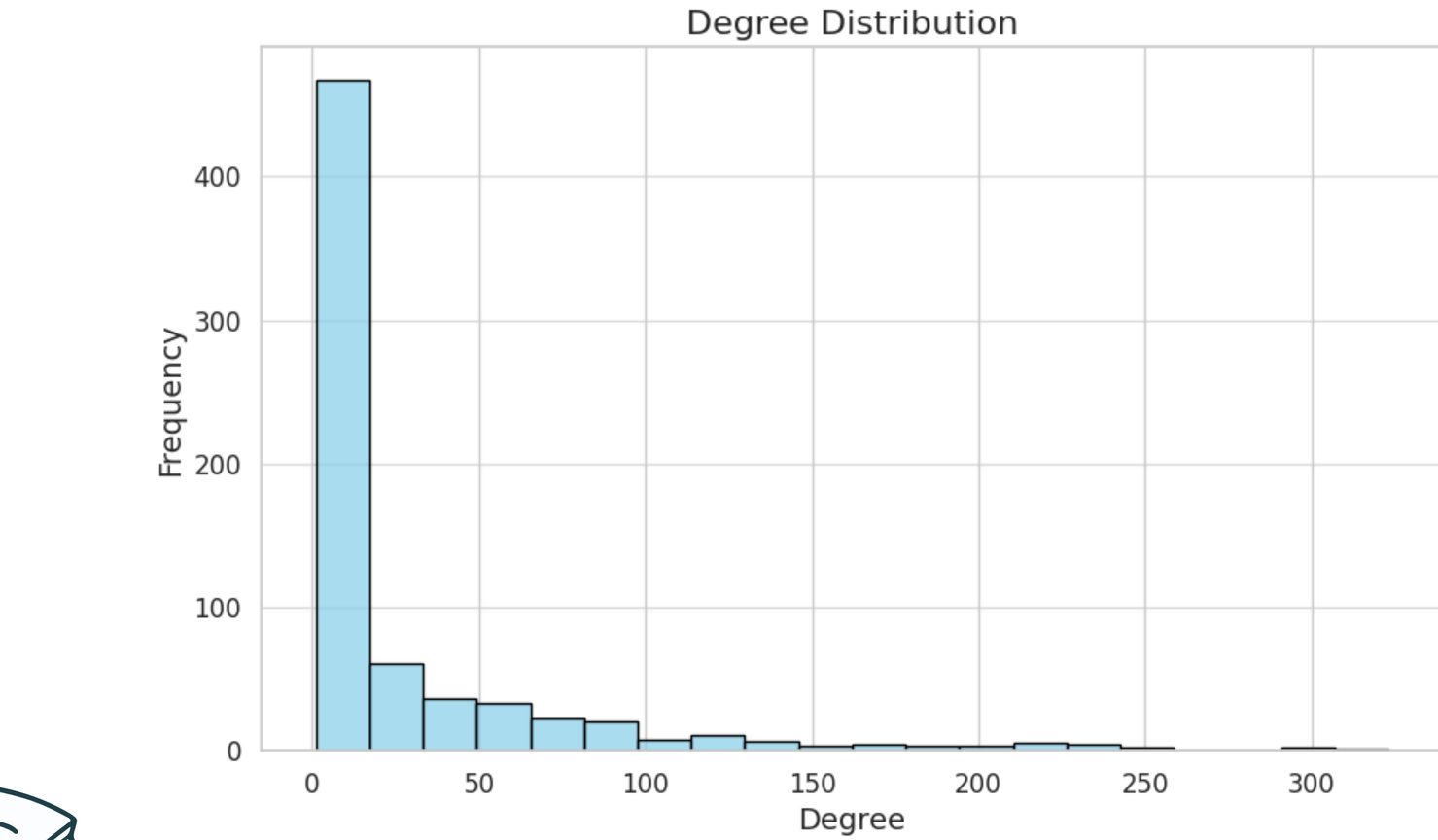
- Weighted links are added between two nodes based on their score parameter.
- For computational reasons only a random sample of the data is used.
- The result is a graph of 695 nodes and 9894 edges.

The Network

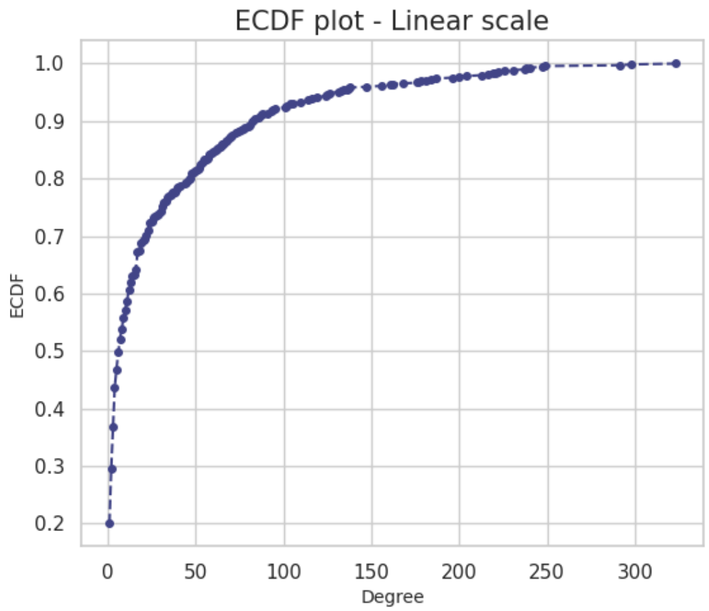
Metric	Value
Number of Nodes	695
Number of Edges	9894
Min Degree	1
Max Degree	323
Mean Degree	28.47
Median Degree	7.0
Standard Dev	48.96
Density	0.04

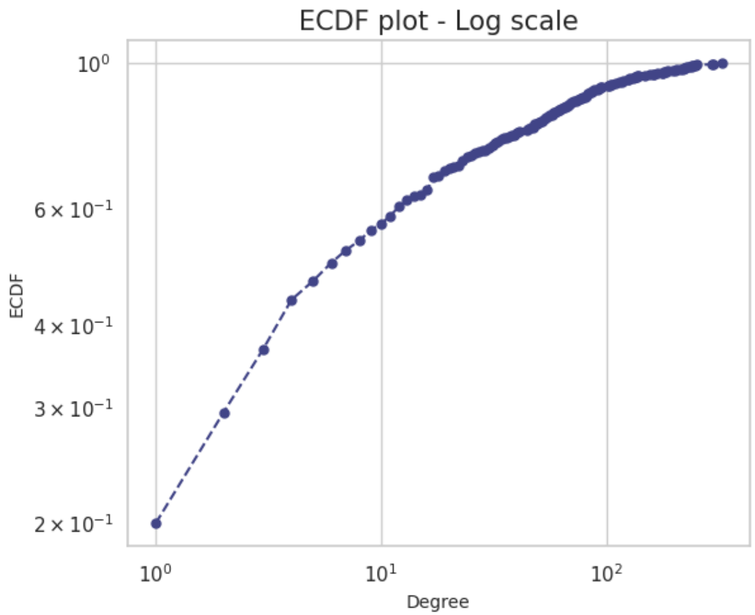


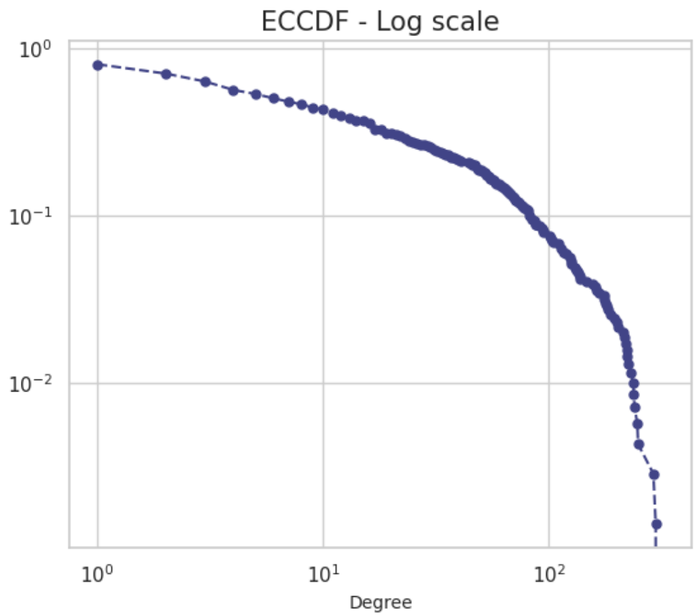
The Degree distribution



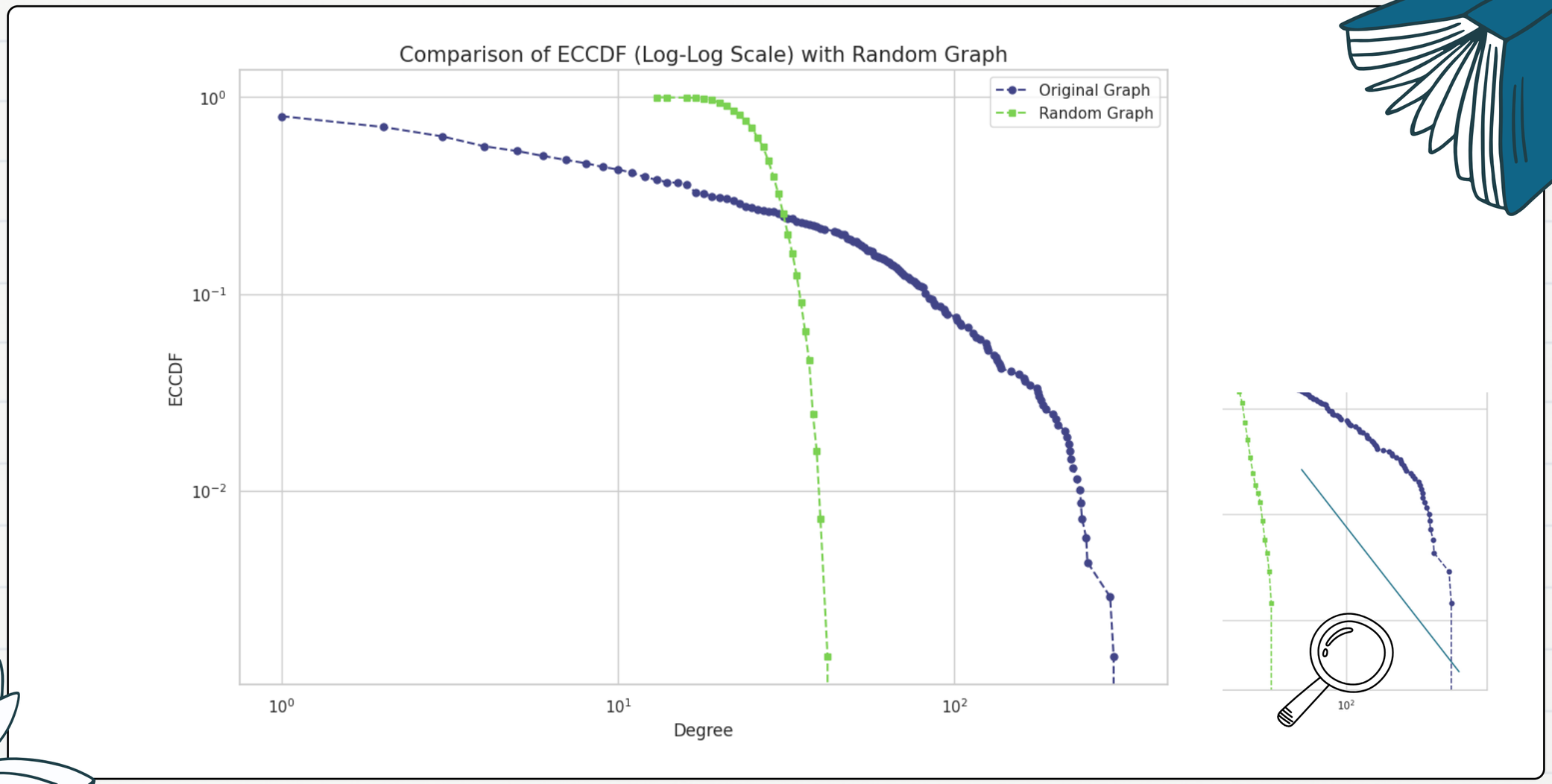
The Degree distribution



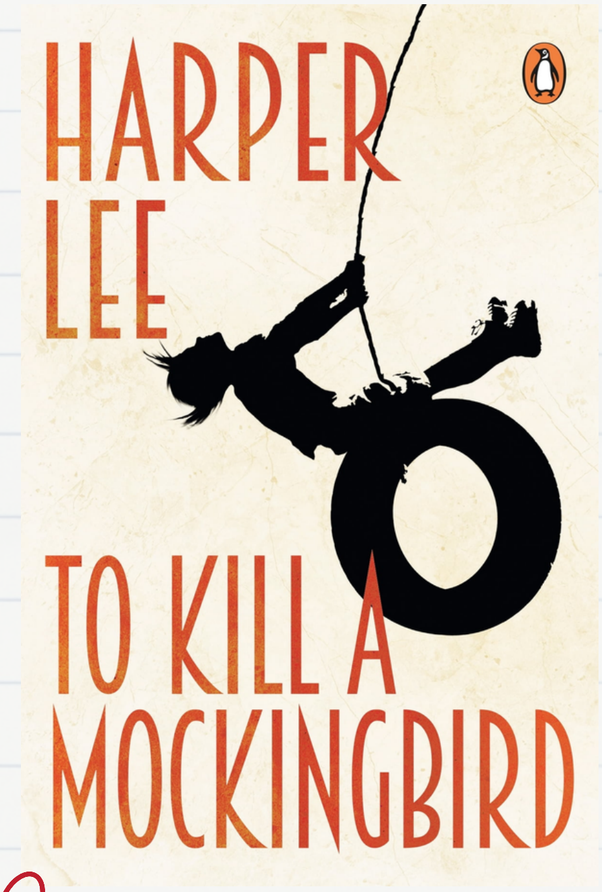




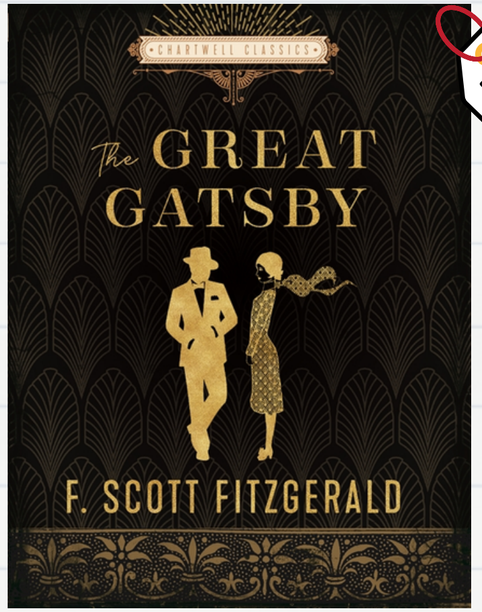
Comparison with Random Network



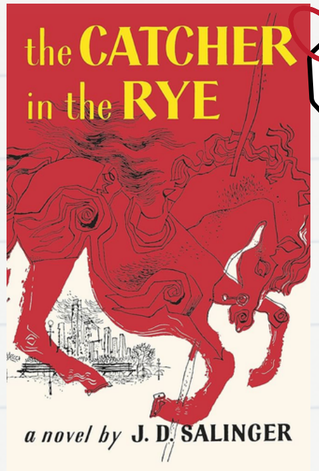
Highest degree books – Hubs



323



298

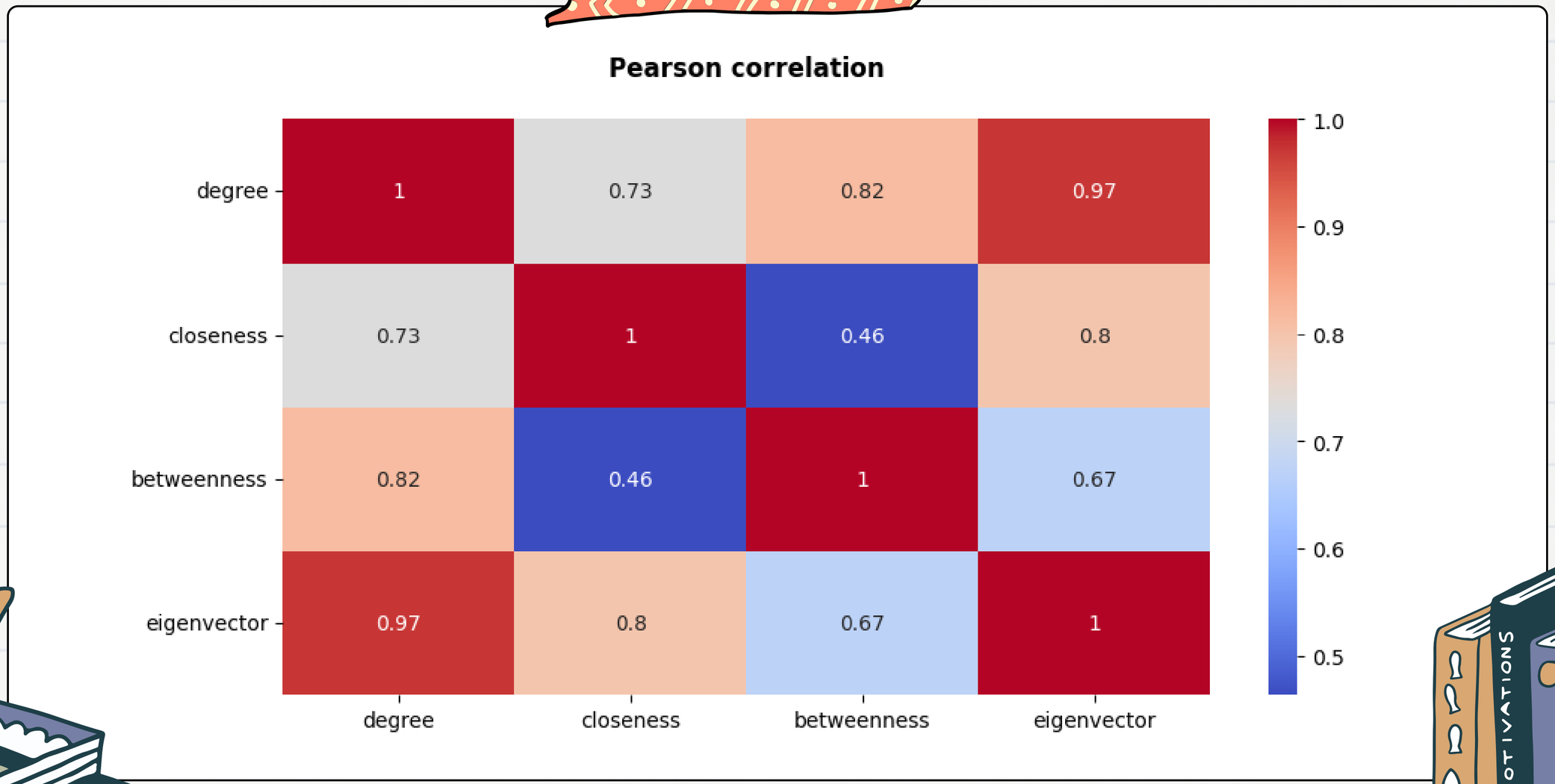


291

The books with the highest degrees seem to be some of the all time classics. That makes sense because more people read them and rate them higher.

Centrality

Degree Centrality	Betweenness Centrality	Closeness Centrality	Eigenvector Centrality
To Kill a Mockingbird	To Kill a Mockingbird	To Kill a Mockingbird	To Kill a Mockingbird
The Great Gatsby	The Catcher in the Rye	The Great Gatsby	The Great Gatsby
The Catcher in the Rye	The Great Gatsby	The Catcher in the Rye	The Catcher in the Rye
Me Talk Pretty One Day	Me Talk Pretty One Day	The Kite Runner	The Kite Runner
The Kite Runner	Slaughterhouse-Five	Me Talk Pretty One Day	Me Talk Pretty One Day

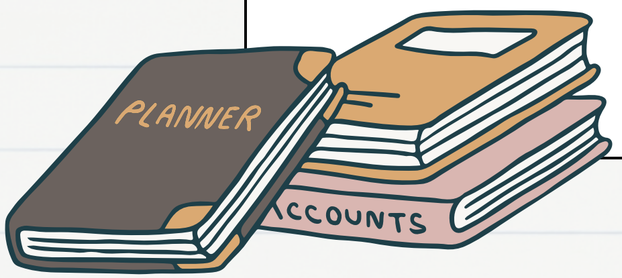

Pearson correlation
degree
closeness
betweenness
eigenvector
1
0.73
0.82
0.97
0.73
1
0.46
0.8
0.82
0.46
1
0.67
0.97
0.8
0.67
1
degree
closeness
betweenness
eigenvector
1.0
0.9
0.8
0.7
0.6
0.5


MOTIVATIONS
FLAVORS OF YOGURT

Assortativity and Clustering Coefficient

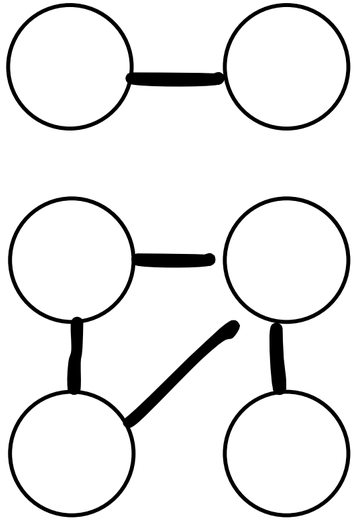
- **Assortativity** of the degree: -0.38
- **Disassortative network:** nodes with many connections are primarily connected to nodes with fewer connections.

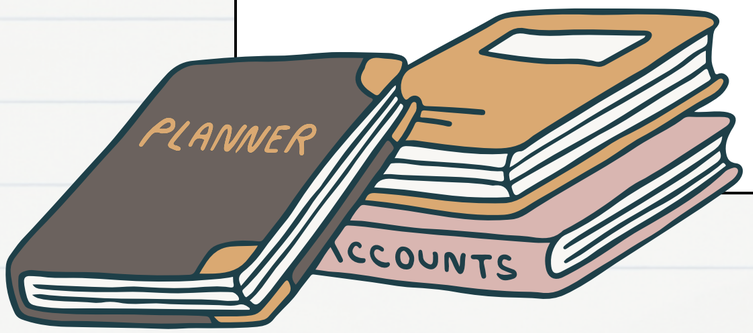
- **Global and Average Local Clustering coefficients:** 0.62
- Highly clustered network



Connected components

- **Disconnected graph**
- **3** Connected Components
- **Size** of the components: 689, 4, 2
- One large component
- **Diameter** of largest component: 7
- Percentage of **bridges** among edges: 1.41%





Community detection

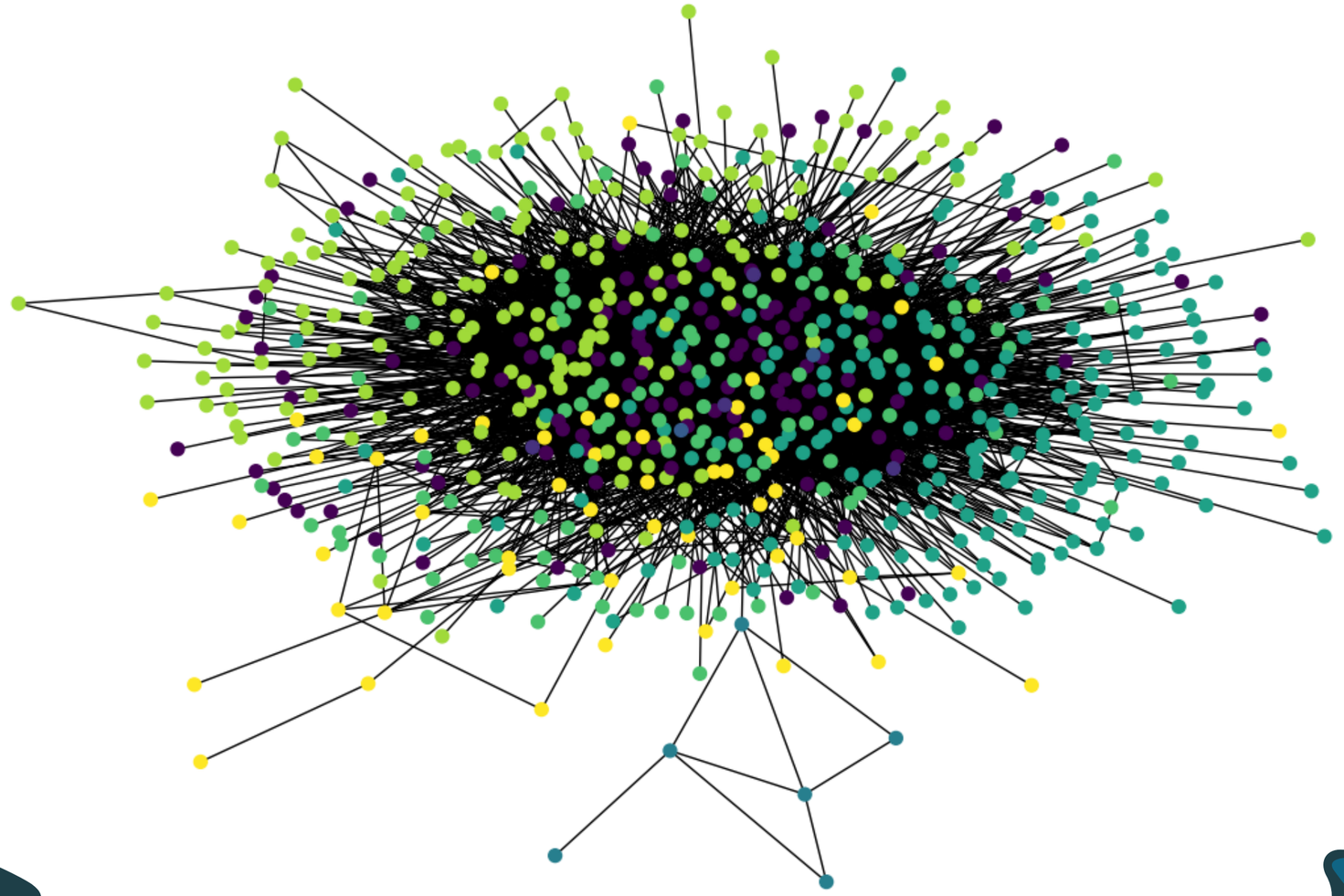
- The partition is made using the Louvain's algorithm.
- Found 8 different communities.
- 3 very small communities (size of 2, 4 and 6)
- Size 2 and 4 communities are the two connected components seen before

The smallest communities are composed by books of the same saga

Community 6: 'The Drawing of the Three (The Dark Tower, #2)', 'The Gunslinger', 'Wolves of the Calla (The Dark Tower, #5)', 'Wizard and Glass (The Dark Tower, #4)', 'The Dark Tower (The Dark Tower, #7)', 'Song of Susannah (The Dark Tower, #6)'

Community 7: 'Seven Up (Stephanie Plum, #7)', 'Hot Six (Stephanie Plum, #6)', 'Three to Get Deadly (Stephanie Plum, #3)', 'High Five (Stephanie Plum, #5)'

Community detection



Node2Vec and RecSys

Node2Vec

- The goal of Node2Vec is to produce high-quality feature representations of nodes in an unsupervised way.
- Use **biased random walks** to generate meaningful sequences of nodes
- Biased promoting nodes closer to the previous one or promoting nodes that are not connected to the previous one: p, q parameters

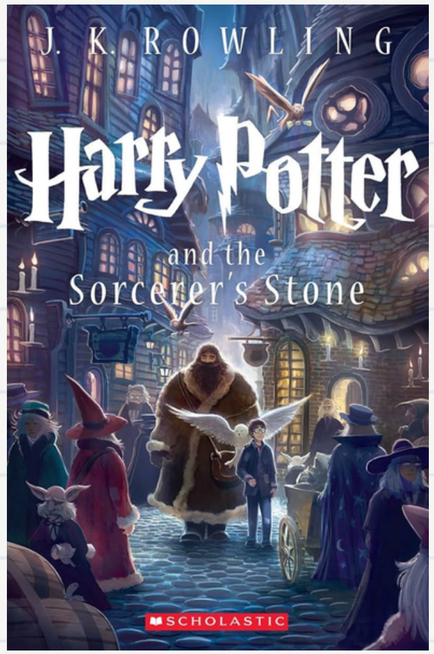
Reccomendation System

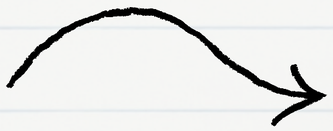
- Choose p and q.
- Use embeddings to train a model that suggests similar books.
- Create “reccomend” function which uses the model.

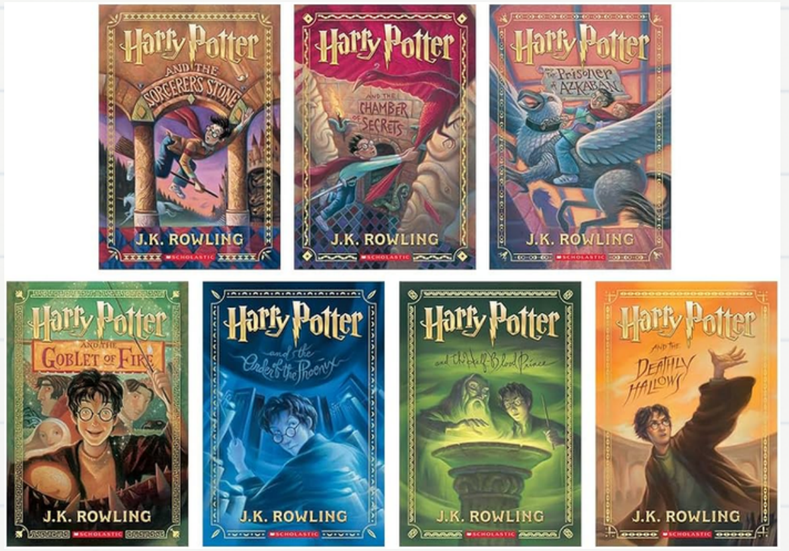
Reccomendation System

```
recommend("Harry Potter and the Sorcerer's Stone (Harry Potter, #1)")
```

```
Harry Potter and the Half-Blood Prince (Harry Potter, #6): 0.72
Harry Potter and the Order of the Phoenix (Harry Potter, #5): 0.71
Harry Potter and the Prisoner of Azkaban (Harry Potter, #3): 0.68
Harry Potter and the Goblet of Fire (Harry Potter, #4): 0.67
Harry Potter and the Deathly Hallows (Harry Potter, #7): 0.62
```





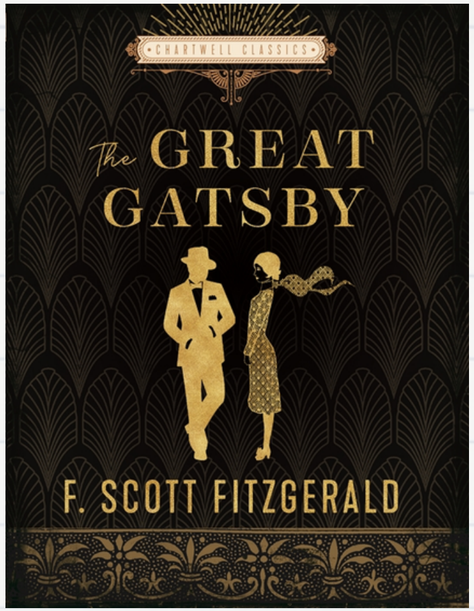


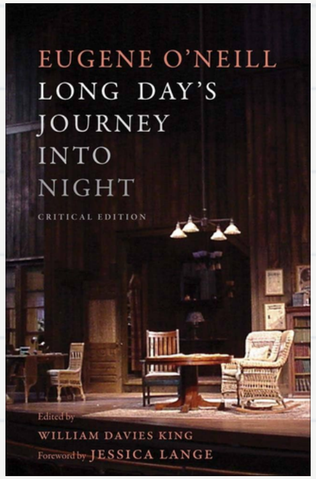
A first simple test of the RecSys shows that it suggestes highly similar and related books.

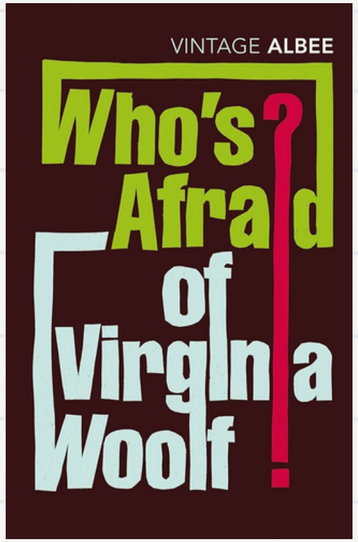
Reccomendation System

```
recommend("The Great Gatsby")
```

```
Long Day's Journey into Night: 0.78
Who's Afraid of Virginia Woolf?: 0.77
Equus: 0.76
The Hunt for Red October (Jack Ryan, #3): 0.73
South of the Border, West of the Sun: 0.73
```







Testing a book that does not belong to a saga, the recsys suggestes books of the same genre and style (Tragedies, theatrical plays).

Conclusions

01

Work with large networks is computationally intense.
Sampling is necessary, but some books may not be present in the graph because of that. Reducing the score for links would lead to a bigger graph, but less significative relations.

02

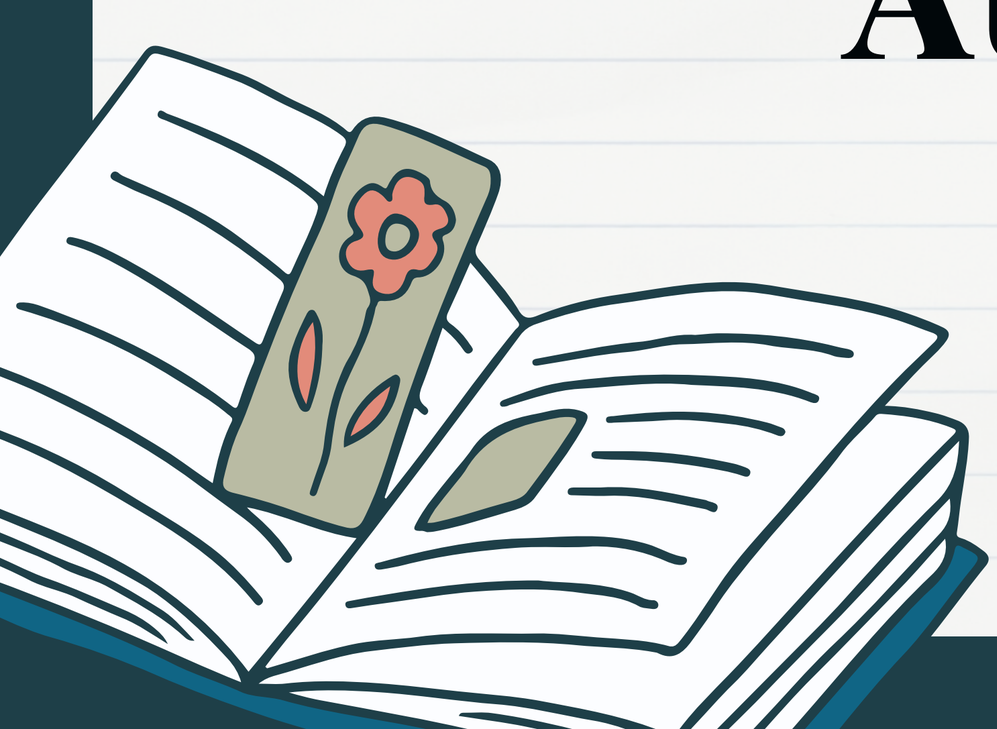
As expected, genre-similar books and books of the same saga seem to be liked together the most.

03

The RecSys works very well. However, it could be improved:

- add more books (bigger graph)
- add features for similarity (genre, author...)
- using a more complex ML technique. (trade-off complexity and efficiency)

Thanks For Your Attention



References

- *Hands-on Graph Neural Networks with python,* Maxime Labonne
- *Network Science,* Albert-László Barabási
- Code: Github