# PageRank-Based Book Recommendation Networks: Amazon Books Dataset

Kocheshkov Maksim
Master in Data Science for Economics
Università degli Studi di Milano

Academic Year 2024/25

**Abstract**

This project applies PageRank to construct a book recommendation system using the Amazon Books Review dataset. By modeling books as nodes connected through shared reviewers, I identify influential books that affect different reading communities. The approach addresses real-world challenges including title deduplication across 212,403 initial entries and memory-efficient processing of 1.4 million reviews. My implementation achieves sub-linear scaling ($O(n^{0.66})$). Results show that network centrality captures different signals than traditional metrics: while Harry Potter ranks first due to its role connecting diverse reader groups, the correlation between PageRank and average rating remains weak (0.082), showing that influence and quality represent distinct concepts in recommendation systems.

## 1 Introduction

Book recommendation systems need to identify which books to suggest from millions of options. Traditional approaches rely on metrics like review counts or average ratings, but these methods have limitations. Popular books from decades ago dominate review counts, while average ratings can be skewed by small groups of dedicated fans.

This project explores an alternative approach using PageRank [1], the algorithm behind Google's search rankings. The key insight is that books exist within networks of reader preferences. When readers enjoy multiple books, they create ”connections” between those titles. Books that cover different reading communities serve as valuable recommendations.

I implement this concept using the Amazon Books Review dataset, constructing a graph where books are nodes and edges represent shared readership. Through PageRank analysis, I identify books that occupy central positions in this preference network, regardless of their individual popularity or ratings.

# 2 Dataset Overview

## 2.1 Dataset Characteristics

The Amazon Books Review dataset from Kaggle provides a large-scale view of reader behavior:

- Total reviews: 3,000,000
- Unique users: 1,008,972
- Unique titles: 212,403 (before deduplication)
- Time period: 1995–2013
- Missing data: 561,787 reviews lack user IDs

## 2.2 Distribution Analysis

Initial exploration revealed extreme sparsity in the data:

- Reviews per book: median 3, mean 14.12, maximum 18,031
- Reviews per user: median 1, mean 2.42, maximum 5,795
- 75% of users reviewed only 1-2 books
- 90% of books received fewer than 10 reviews

This sparsity pattern significantly influenced my preprocessing decisions and algorithm design. Most books lack sufficient data for meaningful network analysis, while a small subset contains rich connection information.

# 3 Data Preprocessing

## 3.1 Basic Cleaning

The initial cleaning phase addressed standard data quality issues:

1. Removed 561,787 reviews missing user identifiers
2. Eliminated 208 reviews without book titles
3. Removed duplicate user-book pairs, keeping the first occurrence

These steps reduced the dataset from 3,000,000 to 2,115,811 unique user-book interactions.

### 3.2 Title Deduplication Challenge

The most complex preprocessing task involved handling title variations. The same book appeared under numerous titles in the dataset. For example, Orwell's ”1984” appeared as:

- ”1984”
- ”George Orwell 1984”
- ”Nineteen Eighty-four”
- ”1984 (Signet Classics)”
- ”1984: Nineteen Eighty-Four [Audiobook]”

I developed a multi-stage deduplication algorithm:

This process reduced unique titles from 212,403 to 67,892. After filtering for books with at least 10 reviews, I retained 33,957 books for analysis.

### 3.3 Semantic Deduplication Enhancement

For particularly challenging cases, I implemented semantic similarity using sentence transformers. This caught variations that string matching missed, such as:

- ”The Hobbit” → ”The Hobbit, or There and Back Again”
- ”HP and the Sorcerer's Stone” → ”Harry Potter and the Philosopher's Stone”

Due to computational constraints, I applied semantic matching only to the 5,000 most reviewed titles, achieving an additional 821 successful merges.

## 4 Network Construction

### 4.1 Graph Definition

I constructed an undirected graph $G = (V, E)$ where:

- $V$: Set of books (after deduplication)
- $E$: Set of edges between books
- Edge $(b_i, b_j) \in E$ if books $b_i$ and $b_j$ share at least $k$ common reviewers

I set $k = 2$ to filter noise while preserving meaningful connections.

**Algorithm 1** Title Deduplication Algorithm

```
Input: Set of raw titles
Output: Mapping to canonical titles

Phase 1: Preprocessing
for each title in dataset do
    Remove format indicators: [Audiobook], (Kindle), -Paperback
    Strip edition information: "1st edition", "revised"
    Remove ISBN patterns
    Delete publisher information in parentheses
    Normalize whitespace and punctuation
end for

Phase 2: Similarity Grouping
Create empty groups dictionary
for each processed title do
    matched ← false
    for each existing group do
        if Jaro-Winkler similarity %¿ 0.85 then
            Add title to group
            matched ← true
            break
        end if
    end for
    if not matched then
        Create new group with this title
    end if
end for

Phase 3: Canonical Selection
for each group do
    Count reviews for each title variant
    Select variant with highest review count as canonical
    Map all variants to canonical title
end for
```

## 4.2 Memory-Optimized Implementation

The approach of creating all possible book pairs for each user is hard. A user who reviewed $n$ books generates $\binom{n}{2}$ potential edges. With one user having reviewed 5,795 books, this would create over 16 million edge candidates from a single user.

I implemented several optimizations:

```python
def build_book_graph_optimized(df, min_common_reviewers=2, chunk_size=100000):
    # Initialize edge counter
    book_pairs = Counter()

    # Process users in chunks
    for chunk_start in range(0, n_users, chunk_size):
        chunk_users = all_users[chunk_start:chunk_start + chunk_size]
        chunk_df = df[df[’User_id’].isin(chunk_users)]

        # Group by user
        user_books = chunk_df.groupby(’User_id’)[’Title’].apply(list)

        for user_id, books in user_books.items():
            # Limit books per user to prevent memory explosion
            if len(books) > 50:
                books = books[:50]

            # Generate pairs
            for book1, book2 in combinations(sorted(books), 2):
                book_pairs[(book1, book2)] += 1

        # Periodic garbage collection
        if chunk_idx % 5 == 0:
            gc.collect()

    # Build graph from pairs exceeding threshold
    G = nx.Graph()
    for (book1, book2), count in book_pairs.items():
        if count >= min_common_reviewers:
            G.add_edge(book1, book2, weight=count)

    return G
```

Listing 1: Optimized graph construction approach

The 50-book limit per user prevents memory exhaustion while minimally impacting results, as users with hundreds of reviews often include bulk purchases or systematic reviewing patterns.

## 4.3 Resulting Network Properties

The final graph exhibits interesting structural properties:

- Nodes: 33,957 books
- Edges: 410,303 connections
- Average degree: 24.17
- Network density: 0.000712

- Connected components: 12,765
- Largest component: 20,192 nodes (59.4% of total)

# 5 PageRank Implementation

## 5.1 Algorithm Adaptation

PageRank models a random walk process on the graph. For my book network, this represents a reader randomly selecting their next book based on current reading patterns. The PageRank score $PR(v)$ for book $v$ is computed as:

$$PR(v) = \frac{1-d}{|V|} + d \sum_{u \in N(v)} \frac{PR(u)}{deg(u)} \tag{1}$$

where $d$ is the damping factor (0.85), $|V|$ is the total number of books, $N(v)$ represents books connected to $v$, and $deg(u)$ is the degree of book $u$.

## 5.2 Power Iteration Method

I implemented PageRank using power iteration on a sparse transition matrix:

```python
def pagerank_power_iteration(G, damping=0.85, max_iter=100, tolerance=1e-6):
    # Focus on largest connected component
    largest_cc = max(nx.connected_components(G), key=len)
    G_sub = G.subgraph(largest_cc).copy()

    # Build sparse transition matrix
    n = len(G_sub)
    row_ind, col_ind, data = [], [], []

    node_to_idx = {node: i for i, node in enumerate(G_sub.nodes())}

    for i, node in enumerate(G_sub.nodes()):
        neighbors = list(G_sub.neighbors(node))
        if neighbors:
            prob = 1.0 / len(neighbors)
            for neighbor in neighbors:
                j = node_to_idx[neighbor]
                row_ind.append(j)
                col_ind.append(i)
                data.append(prob)

    M = csr_matrix((data, (row_ind, col_ind)), shape=(n, n))

    # Initialize uniform distribution
    pr = np.ones(n) / n
    teleport = (1 - damping) / n

    # Iterate until convergence
    for iteration in range(max_iter):
        pr_new = damping * M.dot(pr) + teleport

        # Check convergence
        if np.abs(pr_new - pr).sum() < tolerance:
            break

        pr = pr_new

    # Normalize scores
    return pr / pr.sum()
```

Listing 2: PageRank implementation core logic

The algorithm typically converged within 40 iterations, requiring less than 2 seconds for the 20,192-node component.

### 5.3 Handling Disconnected Components

The presence of 12,765 connected components posed a challenge. I computed PageRank only on the largest component containing 59.4% of books. Books in smaller components received zero PageRank, representing a limitation discussed in Section 8.

## 6 Results and Analysis

### 6.1 Top-Ranked Books

Table 1 presents the books with highest PageRank scores:

Table 1: Top 15 Books by PageRank Score

| Rank | Title | PageRank | Degree | Reviews |
|---:|---|---:|---:|---:|
| 1 | Harry Potter and The Sorcerer's Stone | 0.00277 | 2,516 | 3,663 |
| 2 | Guns, Germs, and Steel | 0.00201 | 1,635 | 1,156 |
| 3 | Five People You Meet in Heaven | 0.00185 | 1,684 | 1,752 |
| 4 | To Kill a Mockingbird | 0.00178 | 1,897 | 1,621 |
| 5 | 1984 | 0.00178 | 1,840 | 1,441 |
| 6 | Atlas Shrugged | 0.00176 | 1,652 | 2,644 |
| 7 | The Hobbit | 0.00175 | 1,813 | 3,562 |
| 8 | The Catcher in the Rye | 0.00174 | 1,895 | 2,134 |
| 9 | The Great Gatsby | 0.00170 | 1,914 | 1,435 |
| 10 | Night | 0.00165 | 1,618 | 1,578 |
| 11 | Memoirs of a Geisha | 0.00165 | 1,651 | 1,629 |
| 12 | The Hobbit (alternate edition) | 0.00164 | 1,730 | 3,577 |
| 13 | Great Gatsby (Everyman) | 0.00164 | 1,842 | 1,423 |
| 14 | Fahrenheit 451 | 0.00156 | 1,685 | 1,164 |
| 15 | Nineteen Eighty-four | 0.00146 | 1,545 | 1,440 |

### 6.2 Network Structure Analysis

The network exhibits several notable characteristics:

**Degree Distribution:** Follows a power law, indicating a scale-free network. Most books have few connections while a small number serve as hubs.

**Clustering Coefficient:** The average clustering coefficient of 0.5513 indicates strong local community structure. Books form tight clusters, likely corresponding to genres or themes.

**PageRank Inequality:** The Gini coefficient of 0.6213 reveals high concentration of PageRank scores. The top 10% of books capture 55.0% of total PageRank mass.

### 6.3 Correlation Analysis

I examined relationships between PageRank and traditional metrics:

Table 2: Correlation Analysis Results

| **Metric Pair** | **Pearson Correlation** | **Interpretation** |
|---|---:|---|
| PageRank vs Degree | 0.989 | Nearly perfect positive correlation |
| PageRank vs Review Count | 0.758 | Strong positive correlation |
| PageRank vs Average Rating | 0.082 | Negligible correlation |

The weak correlation with average rating confirms that PageRank captures network importance rather than perceived quality.

### 6.4 Temporal Patterns

Review distribution over time shows significant variation, with a dramatic spike in 2012-2013. This temporal concentration may bias results toward books popular during this period, though PageRank's reliance on network structure partially mitigates this effect.

## 7 Scalability Analysis

### 7.1 Computational Performance

I evaluated scalability by processing increasingly large subsets:

Table 3: Scalability Benchmarks

| **Dataset Fraction** | **Books** | **Edges** | **Time (s)** |
|---:|---:|---:|---:|
| 10% | 3,396 | 4,812 | 0.45 |
| 25% | 8,489 | 28,934 | 1.82 |
| 50% | 16,979 | 106,445 | 5.64 |
| 75% | 25,468 | 237,891 | 10.23 |
| 100% | 33,957 | 410,303 | 20.89 |

Regression analysis yields time complexity of approximately $O(n^{0.66})$, demonstrating sub-linear scaling. This efficiency stems from:

- Sparse matrix operations scaling with edges rather than nodes squared
- Faster PageRank convergence on larger networks due to better connectivity
- Efficient memory management preventing

### 7.2 Memory Efficiency

Peak memory usage remained under 10MB for the graph structure itself, achieved through:

- Chunked user processing (100,000 users per batch)
- Sparse matrix representation for transition matrices
- Aggressive garbage collection between chunks
- Book-per-user limits preventing quadratic explosion

## 8 Discussion

### 8.1 Interpretation of Results

The top-ranked books reveal interesting patterns about reading behavior:

**Series Entry Points:** First books in popular series (Harry Potter, Lord of the Rings) rank highly, suggesting they serve as gateways to extended reading experiences.

**Cross-Genre Bridges:** "Guns, Germs, and Steel" ranks second despite being non-fiction, indicating it appeals to readers across traditional genre boundaries.

**Enduring Classics:** Books like "1984" and "To Kill a Mockingbird" maintain high centrality decades after publication, confirming their continued relevance in contemporary reading networks.

**Quality vs. Influence:** The weak correlation between PageRank and average rating demonstrates that network importance differs from perceived quality. A book can be influential without being universally loved.

### 8.2 Methodological Limitations

Several limitations affect my analysis:

**Component Isolation:** Only 59.4% of books belong to the giant component where PageRank operates. The remaining 40.6% receive zero PageRank regardless of their importance within smaller communities. This particularly affects specialized academic texts and non-English books.

**Binary Edge Weights:** I treat all connections equally—two common reviewers create the same edge weight as two hundred. While this prevents popular books from completely dominating rankings, it loses information about connection strength.

**Temporal Dynamics:** The static network ignores how reading patterns evolve over time. A book's influence likely varies as cultural contexts shift, but our approach captures only an aggregate view.

**User Behavior Assumptions:** Capping users at 50 books assumes that extremely prolific reviewers provide less reliable signals. While preventing computational issues, this may undervalue genuine reading enthusiasts.

**Deduplication Errors:** Despite careful implementation, some incorrect merges occurred. The algorithm merged "Golden Retrievers For Dummies" with "Labrador Retrievers For Dummies"—distinct books that happen to share similar titles and readership patterns.

# 9 Conclusions and Future Work

This project successfully applied PageRank to book recommendation, revealing the network structure underlying reader preferences. By treating books as nodes connected through shared readership, we identified influential titles that traditional metrics might overlook.

Key technical contributions include:

- An effective title deduplication pipeline handling real-world data variations
- Memory-efficient graph construction enabling full dataset processing
- Empirical demonstration of sub-linear scaling behavior
- Analysis revealing how network-based and traditional metrics capture different signals

The results suggest that network analysis can complement existing recommendation approaches by identifying books that connect diverse reading communities. While Harry Potter's top ranking might seem obvious, its position reflects not just popularity but its unique role bridging multiple reader groups.

Future research directions include:

- Implementing weighted PageRank to capture connection strength
- Developing methods to incorporate disconnected components
- Adding temporal analysis to track influence evolution
- Creating personalized PageRank variants for individual recommendations
- Combining network metrics with content features for hybrid systems

The project demonstrates that even well-studied algorithms like PageRank can yield new insights when applied to novel domains with appropriate adaptations for real-world constraints.

# Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work, and including any code produced using generative AI systems. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

# References

[1] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web*. Stanford InfoLab.

[2] Amazon Books Reviews Dataset. (2013). *Kaggle Datasets*. `https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews`