

PageRank-based Link Analysis on Books

Linh Chi HOANG

Data Science for Economics
33141A

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.

1 INTRODUCTION

This project was carried out within the course *Algorithms for Massive Data* taught by Professor Dario Malchiodi. Based on the publicly available Amazon Book Review dataset on Kaggle, the objective was to develop a ranking system using PageRank-based Link Analysis to identify the most influential books. Specifically, the books were linked if they were rated with high scores by at least two different users. In other words, the importance of a book depended on how many other books that shared a significant number of high-rating users with it.

Since inconsistencies in book titles could influence the link structure, the project also investigated a second approach, in which similar titles were identified and merged before applying the PageRank algorithm. Particularly, MinHash signatures combined with Locality Sensitive Hashing were employed to efficiently generate candidate pairs of potentially similar titles, which were subsequently verified using exact Jaccard similarity. In the end, the project compared the results obtained with and without merging duplicated book titles.

2 DATASET

The original dataset consists of two files: *Books-rating.csv* and *books-data.csv*, as illustrated in Figure 1. These datasets refer to the same collection of books and can be linked via the *Title* attribute. The *books-data.csv* file provides metadata for each book, such as title, authors, description, and image, while *Books-rating.csv* contains user IDs, review scores, ratings' helpfulness, and timestamps. Together, these datasets allow us to combine user rating behavior with book-level descriptive attributes.

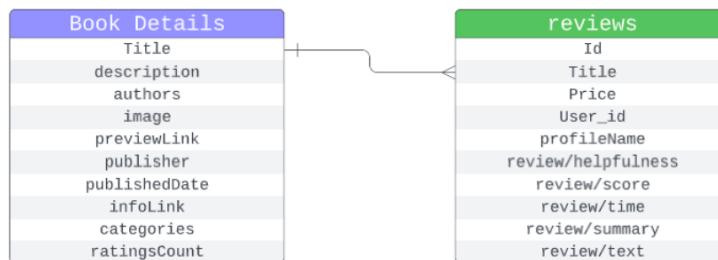


Figure 1. Columns of Books-data.csv and books-rating.csv and their relationship (source: Kaggle).

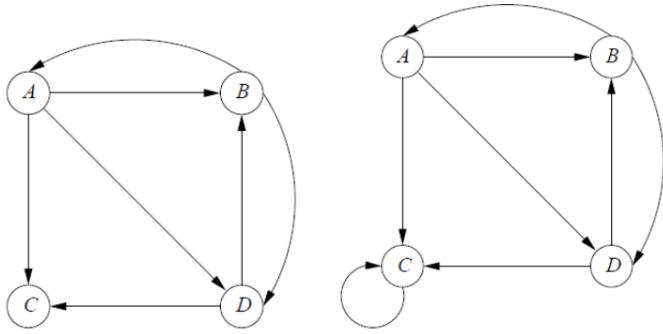


Figure 2. Node C is a dead end (left) and a spider trap (right) (source: Textbook *Mining of Massive Datasets*)

2.1 Data preprocessing

For the purpose of the project, *Books-rating.csv* dataset was mainly used to extract three attributes *Title*, *User-id* and *review/score*. The file contains in total 3 million rating entries, corresponding to 212,404 unique book titles, and 3 million users giving reviews.

To prepare the data for PageRank-based link analysis, several preprocessing steps were applied. First, *Books-rating.csv* dataset was filtered to retain only reviews with a score of at least 4, which contributed approximately 2.5 million entries of the entire file. Checking for NAN values was the next step. A total of 208 missing entries were found in the *Title* attribute and 561,787 in *User-id*. All records containing missing values in either column were then removed.

For the second approach, a further preprocessing step was also applied, where book titles were first normalized by converting them to lowercase and removing spaces, punctuation, and special characters before a Jaccard similarity measure was computed to detect near-duplicate titles. As a result, the number of unique titles decreased from 380 to 378.

Besides, because of RAM limitations, the analysis was further restricted to books with at least 81 unique users who rated them. The filtered dataset, consequently, consisted of a total of 75,786 rows with 60,081 unique users and 380 unique book titles.

3 METHODOLOGY

3.1 PageRank Algorithm

The PageRank algorithm computes an important score for each node in a graph by analyzing how these nodes are connected to each other.

In this project, books were modeled as nodes, and an edge between $book_i$ and $book_j$ was created if at least two users rated both books with a high score ($review/score \geq 4$). The expected outcome of PageRank algorithm is a column vector v_k , whose j^{th} component is the probability representing how likely a user is to land on that book by following edges in the graph.

The PageRank vector v_k is computed iteratively by repeatedly multiplying the transition matrix \mathcal{M} with the distribution vector from the previous iteration. After k iterations, the updated vector is

$$v_k = \mathcal{M}v_{k-1} \quad (1)$$

The convergence is reached when applying the transition matrix no longer changes the distribution vector.

However, in most real-world graphs, two issues commonly arise in link analysis: *dead ends* and *spider traps*. A dead end is a node with no outgoing edges, while a spider trap is a group of nodes that link only to each other and have no edges leading outside the group, as shown in Figure 2. In other words, a spider trap is a set of nodes with no dead ends but no arcs out. To avoid these problems, the calculation of PageRank is modified by introducing a probability $(1 - \beta)$, which allows a user to randomly teleport

to a random node. The damping factor β was set 0.85 in this project. Therefore, the update rule becomes

$$v_k = \beta \mathcal{M} v_{k-1} + (1 - \beta) \frac{1}{n} \mathbf{1} \quad (2)$$

where:

- n is the number of nodes in the graph
- $\mathbf{1}$ is a column vector of all ones (the number of the entries is equal to the number of the nodes)
- $\frac{1}{n} \mathbf{1}$ is the uniform probability distribution.

3.1.1 Transition matrix \mathcal{M}

The transition matrix \mathcal{M} describes how the probability distribution of the random surfer evolves after one iteration. In this project, \mathcal{M} represented the book-book transition probabilities. Each entry m_{ij} denoted the probability of moving from book j to book i .

To construct this matrix, we first computed, for every pair of books, the number of users who rated both books with a high score. These counts formed a weighted adjacency matrix. Each column was then normalized by its column sum so that

$$\sum_i m_{ij} = 1,$$

ensuring that \mathcal{M} was column-stochastic. The resulting transition matrix has shape (380, 380), corresponding to the 380 unique book titles included in the final dataset.

3.2 MinHash Signatures combined Locality Sensitive Hashing (LSH)

In simple settings, Jaccard similarity can be computed directly between all pairs of sets. However, this approach has quadratic complexity, $O(n^2)$, which becomes impractical when dealing with large-scale datasets. For this reason, the project was extended by incorporating MinHash signatures combined with Locality Sensitive Hashing (LSH) to generate candidate pairs of potentially similar items. These candidate pairs were then verified using exact Jaccard similarity to eliminate false positives introduced by the approximate hashing step.

In this project, each book title was represented as a set of words extracted from the cleaned title text. MinHash signatures and LSH index were constructed using libraries `datasketch.MinHash` and `datasketch.MinHashLSH`, respectively, with 128 hash permutations. The LSH threshold was set to 0.5, meaning that pairs with Jaccard similarity around or above this value have a high probability of being selected as candidates.

3.2.1 MinHash Signatures

Given a *Characteristic Matrix* M , where each column represents a title as a set of words and each row corresponds to a unique word, the entries of the matrix are binary, taking value 1 if the word is contained in the title and 0 otherwise. MinHash constructs hash functions based on random permutations of the rows of M . For a given permutation, the MinHash value of a column is defined as the index of the first row (according to the permutation) in which the column contains a 1.

Therefore, by using n hash functions, it is possible to construct a *signature matrix* whose number of columns is the same as that of the *characteristic matrix*, while the number of rows is equal to n , each corresponding to a distinct hash function rather than to individual elements. In this way, it becomes unnecessary to explicitly consider all row permutations. Each row of the *signature matrix* can thus be interpreted as the outcome of a single MinHash experiment.

3.2.2 Locality Sensitive Hashing (LSH)

Although MinHash provides a compact representation of sets, comparing all pairs of MinHash signatures can still become infeasible when the number of items is large. Locality Sensitive Hashing (LSH) addresses this issue by further hashing the signature matrix in a structured way to efficiently identify candidate pairs.

Specifically, LSH divides the signature matrix into b bands, each consisting of r rows, as illustrated in Figure 3. For each band, a

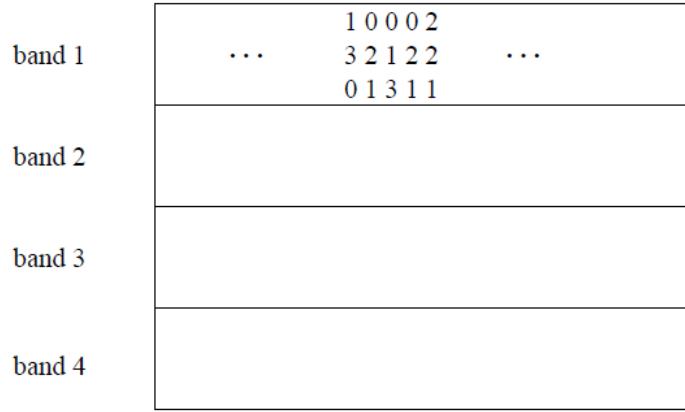


Figure 3. Example for the signature matrix divided into 4 bands consisting 3 rows each (source: Textbook *Mining of Massive Datasets*).

Verified similar pairs from Jaccard: 10			
	Title1	Title2	Sim
0	the picture of dorian gray classic collection ...	the picture of dorian gray the classic collection	0.777778
6	the picture of dorian gray	the picture of dorian gray the classic collection	0.714286
2	lord of chaos turtleback school library bindin...	shadow rising turtleback school library bindin...	0.692308
1	harrington on hold em expert strategy for no l...	harrington on hold em expert strategy for no l...	0.687500
9	a crown of swords turtleback school library bi...	shadow rising turtleback school library bindin...	0.642857
3	a crown of swords turtleback school library bi...	lord of chaos turtleback school library bindin...	0.642857
7	exodus turtleback school library binding edition	the lorax turtleback school library binding ed...	0.625000
8	exodus turtleback school library binding edition	maniac magee turtleback school library binding...	0.625000
5	the lorax turtleback school library binding ed...	the tao of pooh turtleback school library bind...	0.600000
4	up from slavery	up from slavery an autobiography	0.600000

Figure 4. Pairs of book titles with corresponding Jaccard similarity values (source: *Source code*).

hash function is applied to the corresponding portion of the signature, mapping columns to a set of buckets. If two columns are identical within at least one band, they are hashed into the same bucket and are therefore selected as candidate pairs, regardless of their similarity in the remaining bands.

Consistent with the intuition described above, the MinHash signatures combined with Locality Sensitive Hashing, implemented using the *datasketch.MinHashLSH* library, produced 53 candidate pairs of potentially similar book titles.

3.3 Jaccard Similarity

Mathematically, for two sets S_1 and S_2 , the Jaccard similarity is defined as

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}. \quad (3)$$

In this project, Jaccard similarity was used to verify the candidate pairs generated by MinHash combined with Locality Sensitive Hashing (LSH), since the latter provides only an approximate similarity measure and may produce false positives. As a result, a total of 10 title pairs were retained together with their corresponding Jaccard similarity values, as shown in Figure 4.

A threshold of 0.6 was applied to the Jaccard similarity values, meaning that only title pairs with a similarity of at least 60% were retained for merging. Titles identified as similar were grouped into clusters, and each cluster was assigned a canonical representative title. These canonical titles were then used to replace all titles within their respective clusters, ensuring that duplicated or highly

index	PageRank
Jane Eyre (Large Print)	0.018189757209258944
Jane Eyre (New Windmill)	0.018189757209258944
Wuthering Heights	0.016202152498141113
A Tale of Two Cities - Literary Touchstone Edition	0.015097259873860253
Great Expectations	0.014423246195390326
The Scarlet Letter (Lake Illustrated Classics, Collection 2)	0.012686965659563397
The Plot Against America	0.012635089064003333
Emma (CH) (Jane Austen Collection)	0.01241471636053881
A Christmas Carol (Classic Fiction)	0.012386286817263061
A Christmas Carol, In Prose: Being a Ghost Story of Christmas (Collected Works of Charles Dickens)	0.012386286817263061
Persuasion	0.01122225326964406
Good to Great	0.010967370342232037
Sense And Sensibility (CH) (Jane Austen Collection)	0.010844517447304188
the Picture of Dorian Gray	0.010797405247428669
The Picture of Dorian Gray (Classic Collection (Brilliance Audio))	0.010797405247428669
The Picture of Dorian Gray	0.010797405247428669
The Picture of Dorian Gray (The Classic Collection)	0.010797405247428669
Treasure Island	0.010749024037624356
Little Women	0.009952895917828695
Little Women (Junior Classics)	0.009952895917828695

Figure 5. Book rankings without applying Jaccard similarity computation (source: *Source code*).

index	PageRank
jane eyre new windmill	0.019044439001278533
jane eyre large print	0.019044439001278533
wuthering heights	0.01716479417589845
a tale of two cities literary touchstone edition	0.0158092262841564
great expectations	0.01532970494885647
the plot against america	0.013786661913383253
the scarlet letter lake illustrated classics collection 2	0.013584556610374783
emma ch jane austen collection	0.013505372618656683
a christmas carol in prose being a ghost story of christmas collected works of charles dickens	0.0129856588228505
a christmas carol classic fiction	0.0129856588228505
the picture of dorian gray	0.012615145023031168
persuasion	0.012137141182166923
good to great	0.0117158697942823
sense and sensibility ch jane austen collection	0.011688508955185067
treasure island	0.011373934907107638
little women	0.010824946516233254
little women junior classics	0.010824946516233254
silas marner the classic collection	0.009858693497170216
silas marner classic collection brilliance audio	0.009858693497170216
the killer angels turtleback school library binding edition	0.009668996058524934

Figure 6. Book rankings applying Jaccard similarity computation (source: *Source code*).

similar books were represented consistently. The resulting cleaned set of titles was subsequently used as input to the PageRank algorithm for further analysis.

4 DISCUSSION AND CONCLUSION

Figure 5 and Figure 6 present the final book rankings obtained without and with title merging, respectively. In the case without merging, several books with nearly identical titles appeared as separate entries and received identical PageRank values, which was not informative for link analysis and artificially inflated their perceived importance. After applying the clustering procedure, notable differences in the rankings emerged. For example, *The Picture of Dorian Gray* had its PageRank increasing from 0.0107 to 0.0126, indicating that its earlier importance was partly due to duplicated editions reinforcing one another.

Therefore, combining the PageRank algorithm with a similarity detection method like Jaccard could lead to more reliable rankings, as it helps consolidate duplicated or highly similar titles. However, Jaccard similarity based solely on raw text is not always an ideal solution. It cannot distinguish between books that differ in meaningful ways, such as editions, languages, formats, or publication conditions, which may influence readers' preferences and result in different rating opinions. For example, different versions of *The Picture of Dorian Gray*, such as audiobook and printed editions, are merged despite potentially reflecting distinct user experiences. Thus, more sophisticated similarity measures, for example applying machine-learning-based models, could be used to capture deeper semantic relationships between titles.

APPENDIX

The full implementation of this project is provided as a Jupyter notebook and can be executed on Google Colab. The source code is available at: https://github.com/chihoang811/chihoang811/tree/feea3326c704e15b1abc1dd6bf9ee2c02feea674/PageRank_based_Link_analysis_on_Books