Link Analysis

Andrea Corradini, 46953A

Algorithms for massive data, Data Science for Economics, Università degli Studi di Milano

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.

Abstract

This project develops a scalable implementation of the PageRank algorithm on the Amazon Books Review dataset. From millions of raw reviews, a co-occurrence graph of books is constructed by linking titles reviewed by the same user, and a cleaned core dataset is extracted to ensure a connected and meaningful structure. Two graph instances are derived to study how PageRank behaves as the network grows. The algorithm is implemented via power iteration in Python and extended to Apache Spark to assess distributed scalability. The analysis compares structural properties of the graphs, examines PageRank score distributions, and evaluates runtime as size increases. Results show that PageRank converges efficiently even on graphs exceeding 10,000 nodes and nearly 60,000 edges, while Spark allows scaling to larger datasets.

Introduction

The rapid growth of online platforms has produced vast networks of interconnected entities, making link-analysis techniques essential for ranking, recommendation, and information retrieval. PageRank, originally introduced as the foundation of the Google search engine, remains one of the most influential methods for quantifying node centrality in large graphs. Its power derives from interpreting importance as a random-walk process with teleportation, allowing centrality to emerge from global connectivity rather than local popularity alone. Applying PageRank to real-world data, however, requires addressing challenges related to graph construction, data sparsity, and the efficient computation of iterative algorithms on potentially massive networks.

This project examines how PageRank can be applied to the Amazon Books Review dataset by modeling relationships between books as a co-occurrence graph: two titles are linked whenever at least one user has reviewed both, capturing a behavioral notion of similarity. To reduce noise, only users and books with at least two reviews are retained, producing a core dataset from which two graph instances are derived: a smaller prototype for initial exploration and a larger variant containing more than ten thousand books and nearly sixty thousand edges. Comparing these structures highlights how graph size and density influence the behavior and convergence speed of PageRank.
Scalability is also a central focus. A Python implementation of PageRank provides fine-grained control over data structures, while Apache Spark serves as the distributed counterpart for constructing the co-occurrence graph on larger subsets. The contrast between the two approaches illustrates both the limitations of single-machine computation and the benefits of distributed processing. Finally, a set of scaling experiments evaluates how runtime grows with graph size, offering a practical assessment of PageRank's feasibility in increasingly large scenarios.

Dataset and Graph Construction

The empirical analysis is based on the "Amazon Books Review" dataset, obtained from Kaggle as a large collection of user-generated book ratings. The data are downloaded via the Kaggle API and stored locally as a CSV file, from which only the variables needed for graph construction are retained. The original fields (User_id, Id, Title, review/score) are standardized to the more interpretable user_id, book_id, book_title and rating. Because working directly with several million reviews is unnecessary for illustrating PageRank and its scalability, the project operates in a controlled "subsample mode". A random 5% sample (about 150,000 ratings) is extracted using a fixed seed. Even at this stage, sparsity is immediately apparent: the median user and the median book each occur only once, while a minority of highly active users or popular titles accumulate many reviews. Such extreme imbalance would lead to a fragmented graph dominated by degree-one nodes, preventing PageRank from propagating meaningfully. To address this, a core dataset is defined by retaining only users and books with at least two reviews. This filtering removes most one-off interactions and yields a workable foundation of 30,736 ratings, 11,104 users and 10,592 books. From this core, two graph instances are constructed. The small graph, used for prototyping and visualization, keeps the first 2,000 users and contains 5,875 ratings and 3,678 books. The big graph corresponds to the full core and includes all 11,104 users, producing a structure with more than ten thousand nodes and nearly sixty thousand edges. To run PageRank, books must be indexed as integers, so both subsets are mapped to consecutive user and book indices.

Relationships between books are modeled as a co-occurrence graph. For each user, the set of distinct books they reviewed is collected; users with fewer than two books are skipped, and extremely prolific users (more than 50 books) are capped to prevent a combinatorial explosion of edges. For each eligible user, all unordered pairs of books are generated, contributing one unit to the weight of the corresponding edge. Repeated co-occurrences across different users accumulate. Edges are stored in canonical order to maintain an undirected representation. Applying this procedure produces 13,367 edges among 3,678 books in the small graph, with weights almost entirely equal to one and only a few edges reaching weight five. The big graph contains 59,205 edges among 10,592 books, with weights occasionally rising to fourteen. In both cases, the network is sparse but non-trivial: most books connect to only a few others, while a small group of highly read classics form hubs with degrees reaching up to 182. These diagnostics confirm that the co-occurrence graphs accurately reflect underlying user behavior and provide a solid basis for the PageRank analysis that follows.

PageRank

The ranking phase of the project is based on PageRank, a centrality measure that interprets importance as the probability of visiting a node during a random walk with teleportation. The algorithm models a web-surfer who, at each step, either follows one of the outgoing edges of the current node, chosen uniformly at random, or "teleports" to any node in the graph with equal probability. This mechanism ensures the existence of a unique stationary distribution even in the presence of dead ends or disconnected components. Formally, let $G = (V, E)$ be a directed graph with $|V| = n$ nodes. For any node i, denote by $out(i)$ the set of its outgoing neighbors and by $d_i = |out(i)|$ its out-degree. PageRank defines the vector $r \in \mathbb{R}^n$ as the solution to the fixed-point equation

$$r_i = \frac{1-\alpha}{n} + \alpha \sum_{j:(j\to i)\in E} \frac{r_j}{d_j}$$

where $\alpha \in (0,1)$ is the damping factor, set here to the standard value $\alpha = 0.85$. The first term represents teleportation: with probability $1-\alpha$, the random walker jumps to any node uniformly. The second term represents the probability of arriving at node i by following an edge from a neighbor j. Nodes with high PageRank are those that receive substantial probability mass from important nodes, highlighting the recursive nature of the measure.

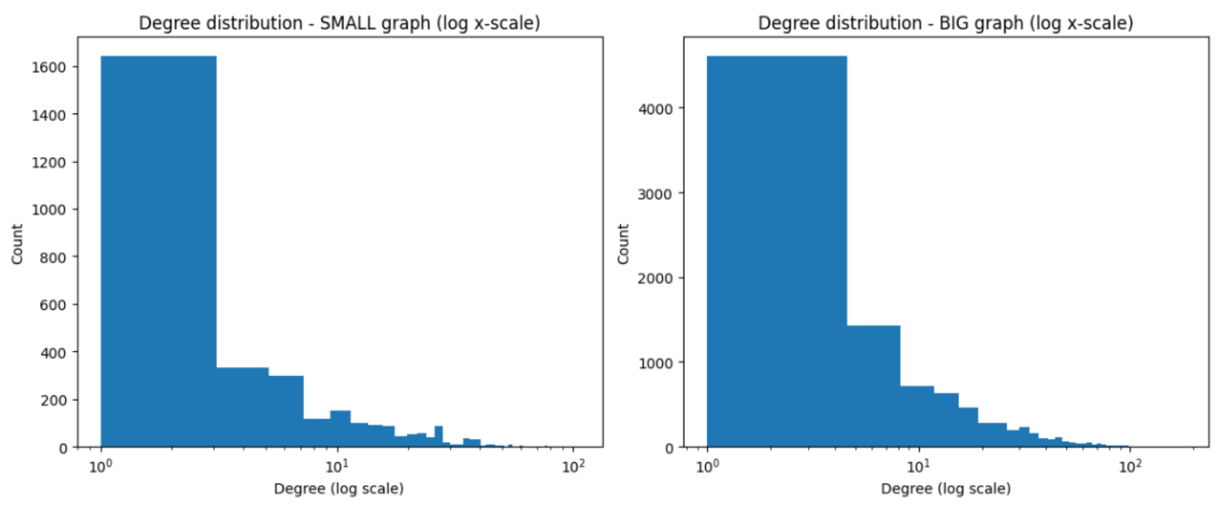
PageRank is computed iteratively. Starting from a uniform distribution $r^{(0)} = \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$, each iteration updates the scores according to the PageRank equation, producing a sequence $(r^{(t)})_{t\geq 0}$. Convergence is declared when the L1 difference between successive vectors becomes smaller than a chosen tolerance. In this project, the update is executed through a vectorized formulation that accumulates contributions using numpy.bincount, allowing efficient aggregation of inbound probabilities across all nodes. A special role is played by nodes with no outgoing edges. In a standard random walk, such “dangling nodes” would trap the walker; in PageRank, their mass is redistributed uniformly to all nodes at each iteration. This reflects the idea that a user encountering a page with no links would initiate a new search rather than terminate the process. In the implementation, dangling mass is computed at each iteration and added to the teleportation component, ensuring numerical stability and correctness. Since the co-occurrence graph is undirected in its raw form, each edge (i, j) is converted into two directed edges $i \to j$ and $j \to i$. This maintains symmetry while allowing PageRank to operate on a directed representation, which is essential for the stochastic interpretation. Before applying PageRank to the full graph, the algorithm is tested on a toy example with four nodes and a small set of directed edges arranged so that one node receives inbound links from three others while another receives none. This configuration highlights the behavior of the method: the node with multiple incoming links systematically receives the highest rank, while a node with no inbound edges converges to the lowest score. The sum of PageRank scores remains equal to one throughout the iterations, confirming correct normalization. Applying PageRank to the small co-occurrence graph yields a ranking over 3,678 books. The distribution of PageRank values is highly skewed: most books receive scores close to the baseline $\frac{1}{n}$, while a small number obtain noticeably larger values. These outliers typically correspond to books with high degree and strong connectivity, such as widely reviewed bestsellers or books that frequently co-occur with many others in users’ histories. This mirrors the behavior observed in real-world web graphs, where a handful of nodes dominate centrality measures. The same procedure is then applied to the big graph, which contains over 10,000 nodes and nearly 60,000 edges. Despite the substantial increase in size, the algorithm converges reliably within a reasonable number of iterations. The distribution of PageRank scores remains consistent with the small graph, though more concentrated due to the larger number of nodes. The top-ranked books again correspond to highly connected hubs. A high PageRank score reflects the fact that a book frequently appears alongside many others in diversified user histories, not necessarily that it is “better” or more highly rated.

Experimental Results

Starting from the core dataset described in the previous sections, two graph instances are considered: a smaller prototype derived from the first 2,000 active users and a larger instance based on the full core. While the construction details and exact sizes are introduced earlier, the comparison here emphasizes how scaling up the graph amplifies heterogeneity. In both cases, degree distributions display a familiar pattern: the vast majority of books connect to only a handful of others, while a limited number of hubs concentrate a disproportionate share of links. Moving from the small to the

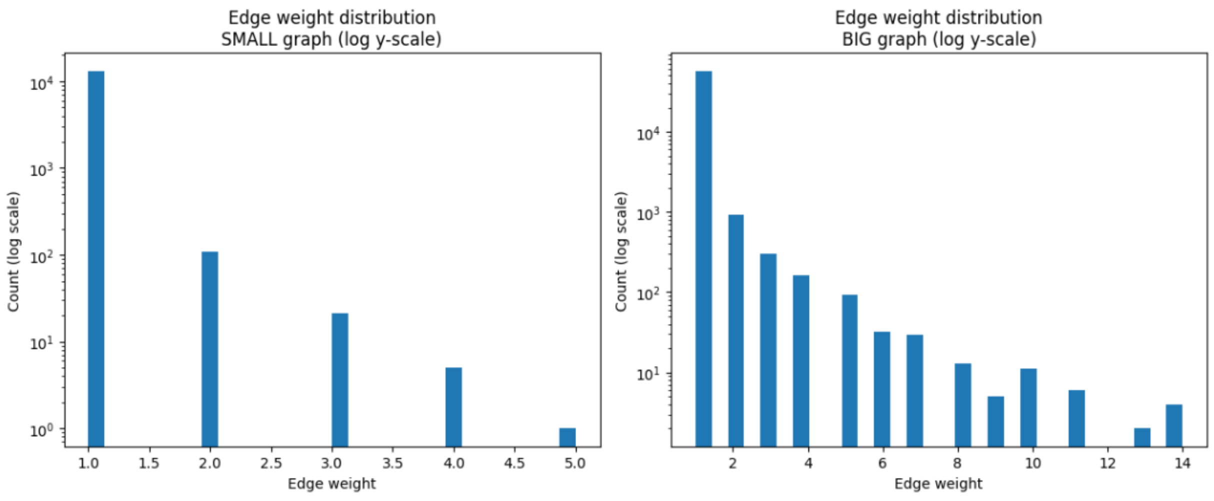
big graph strengthens this effect, with highly connected titles becoming structurally more prominent and the right tail of the degree distribution visibly thickening when plotted on logarithmic scales. A similar picture emerges when focusing on edge weights, which encode how often two books appear together in users' histories. Single co-occurrences dominate overwhelmingly, while edges supported by multiple users represent only a marginal share of the total. Increasing the number of users slightly broadens the tail of the weight distribution, allowing for a few stronger ties, but the overall pattern remains one of extreme sparsity: almost all book pairs are observed together only once. Histograms on log scales clearly show that the mass of the distribution is concentrated at the minimum value, with the heavier weights forming a thin tail that becomes visible only at finer resolution.
A logarithmic transformation on the x-axis makes the full degree spectrum visible (Figure 1), ranging from books with a single neighbor to hubs approaching 10^2. On a linear scale, almost all books collapse into the lowest bin, and the few highly connected titles become visually negligible. The log-scale representation thus exposes both the dominance of low-degree nodes and the thickening of the right tail in the big graph, signalling the emergence of increasingly central books as the network grows.

Fig.1



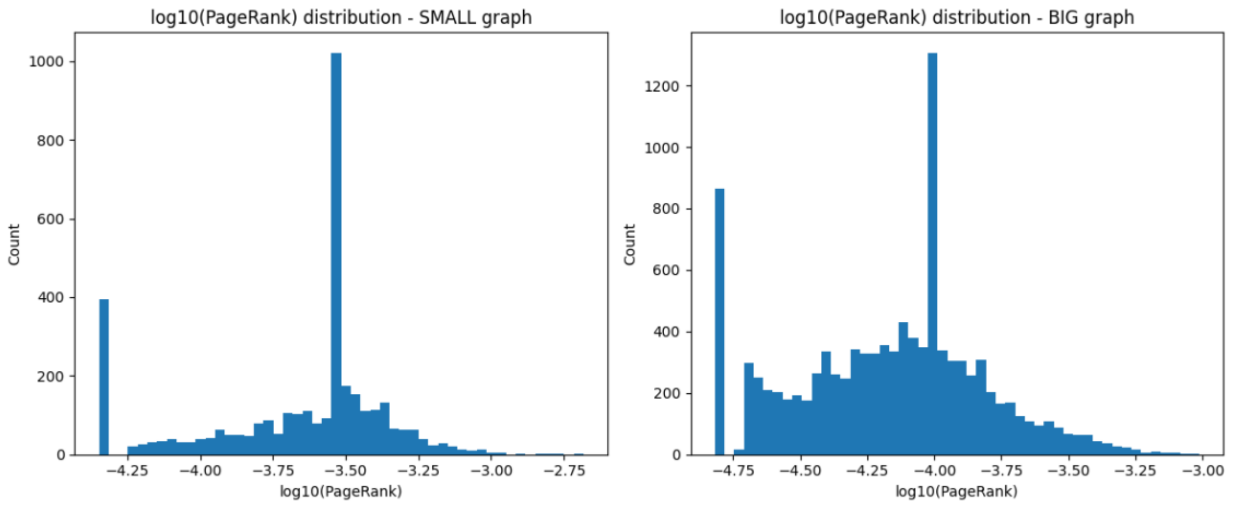
For edge weights (Figure 2), it is the frequencies that span several orders of magnitude. Weight 1 occurs tens of thousands of times, while larger weights appear only sporadically. Applying a logarithmic scale to the y-axis compresses this imbalance and makes the decay of frequencies legible, revealing the rarity of multi-user co-occurrences. Without this transformation, all weights greater than one would visually collapse at the bottom of the plot.

Fig.2



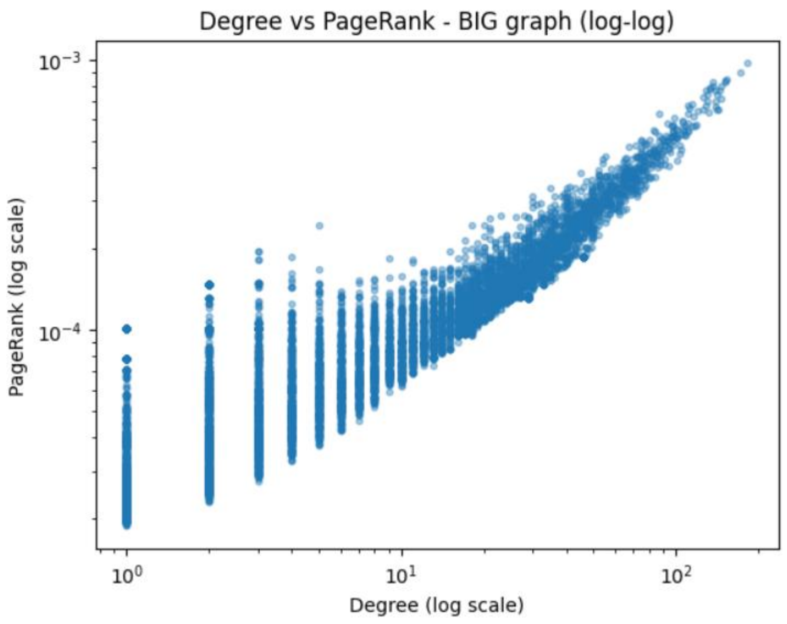
Because PageRank scores are probabilities below one, their $\log_{10}$ values are negative, ranging roughly from -4.25 to -2.75 in the small graph and from -4.75 to -3 in the big one. These intervals correspond to centrality values spanning multiple orders of magnitude. Representing PageRank on a logarithmic axis (Figure 3) reveals the internal structure of the distribution while a linear scale would collapse all probability mass near zero. No information is lost, as logarithms preserve ordering and relative magnitude while making variation visible.

Fig.3



Placing both degree and PageRank on logarithmic axes (Figure 4) provides the most interpretable view of their relationship. This analysis is carried out only for the big graph, which spans several orders of magnitude in both variables, contains a markedly heavier right tail, and exhibits more heterogeneous neighborhoods. Only at this scale do the multiplicative relationships captured by logarithmic scaling become analytically meaningful, revealing deviations from the main trend and illustrating how PageRank depends not just on the number of connections but also on their structural quality.

Fig.4



Node-level statistics help clarify how these structural features interact. In the small graph, the correlation between degree and strength (defined as the sum of incident edge weights) is practically perfect: the Pearson coefficient is extremely close to one. This means that adding new neighbors almost always corresponds to adding single-weight links, and edges with higher weights remain exceptional. When strength is normalized by degree, the resulting average edge weights cluster tightly around one, with just a small group of books displaying substantially higher values. These outliers correspond to titles that systematically co-occur with the same partners across different users, forming compact clusters of strongly associated works and standing out in the ranking by average weight.
This structure of weights also explains why a weighted version of PageRank is not employed in the analysis. Weighted PageRank would adjust transition probabilities so that edges with larger weights exert proportionally stronger influence on the random walk. However, the empirical distribution of weights reveals that the graph is, for all practical purposes, almost unweighted: more than 97 percent of edges in the big graph and nearly 99 percent in the small one have weight exactly equal to one. The remaining edges show only modest deviations, rarely exceeding a weight of two or three and reaching higher values only in an extremely small number of cases. Because the random-walk process is dominated overwhelmingly by single-weight contributions, incorporating weights into PageRank would have produced results nearly indistinguishable from the unweighted variant. The negligible variance in edge intensities offers little meaningful signal to exploit, and a weighted scheme would

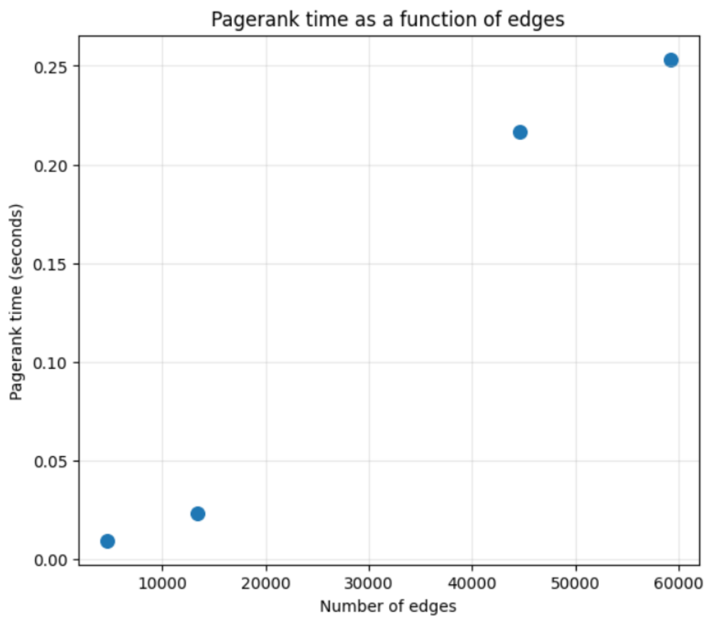
mainly add computational overhead without introducing substantive differences in the ranking. For these reasons, and in line with the goal of isolating structural rather than frequency-based centrality, the unweighted formulation is retained.

On this structural foundation, PageRank produces coherent and interpretable rankings for both graph sizes. In the small instance, the distribution of PageRank scores centers around the theoretical uniform baseline of approximately 2.7×10^{-4}, with most books lying close to this value. Nonetheless, the algorithm identifies a clear right tail: the highest-ranked book reaches about 2.08×10^{-3}, nearly eight times the baseline, indicating a moderate concentration of centrality on a subset of titles. The big graph exhibits an analogous shape, but with a more compressed distribution, which is natural given the larger number of nodes. Here, the median PageRank value lies just above 7.6×10^{-5}, while the top book reaches 9.70×10^{-4}, roughly ten times the uniform reference level. This stability across graph sizes suggests that the random-walk process captures enduring patterns of reader behavior rather than artefacts of sampling.

The identities of the top-ranked books further support this interpretation. In both the small and big graphs, PageRank consistently selects widely read classics that appear across diverse user histories. Works such as *1984*, *Pride and Prejudice*, *The Hobbit*, *Fahrenheit 451*, *The Catcher in the Rye*, *Animal Farm*, *The Great Gatsby*, *The Grapes of Wrath* and several study editions of canonical titles emerge as the most central nodes in the co-occurrence network.

Scalability is assessed through a sequence of experiments in which increasingly large subsets of users are used to construct co-occurrence graphs. The smallest configuration, built from the first one thousand active users in the core dataset, produces a compact network with 2,076 nodes and 4,620 edges. Despite its limited size, graph construction already reveals clear structural patterns while requiring only about 0.19 seconds. PageRank converges almost instantaneously, stabilizing in roughly 0.01 seconds. Expanding to two thousand users produces the small graph used throughout the analysis. At this scale, the graph reaches 3,678 nodes and 13,367 edges, and construction time increases modestly to 0.27 seconds. PageRank remains extremely fast, converging in approximately 0.02 seconds. The medium configuration, based on the first eight thousand users, marks the first substantial growth step: the resulting graph contains 8,847 nodes and 44,582 edges, and construction time rises to about 1.34 seconds. PageRank, however, still converges efficiently, requiring only 0.22 seconds. The largest configuration corresponds to the full core dataset of 11,104 users, producing a graph with 10,592 nodes and 59,205 edges. At this scale, graph construction becomes noticeably heavier, reaching roughly 5.45 seconds, reflecting the quadratic growth of potential book pairs within users' reading histories. In contrast, PageRank remains computationally lightweight. Even on the big graph, convergence is achieved in approximately 0.25 seconds, confirming the efficiency of the vectorized power-iteration implementation. A scatter plot relating the number of edges to PageRank runtime (Figure 5) illustrates this behavior clearly: while graph construction time grows superlinearly due to the combinatorial nature of co-occurrence generation, PageRank exhibits a sublinear trend. These results indicate that PageRank itself is unlikely to become the bottleneck in larger scenarios; rather, the limiting factor is edge construction. Nonetheless, the observed runtimes show that networks substantially larger than those examined here could still be processed comfortably on a single machine before memory constraints begin to dominate.

Fig.5



A final experiment explores the distributed construction of the big graph using Apache Spark. Starting from the indexed ratings, Spark aggregates book pairs across users to produce an edge list that closely parallels the structure obtained in Python. The resulting graph contains 59,161 edges before filtering, a count that remains extremely close to the 59,205 edges produced locally and signals that both pipelines are implementing the same co-occurrence logic at scale. The first edges printed by Spark match the structure of the Python output: identical book pairs appear with the same weights, including examples where multiple users reinforce the same pair, such as the pairs (946, 3223), (3223, 3909) or (261, 284). This alignment shows that Spark correctly reconstructs both single-weight and multi-weight edges, preserving the underlying semantics of user co-occurrences. The slight discrepancy in the total number of edges arises from minor technical differences in the distributed pipeline. In particular, Spark requires explicit casting of user and book indices from strings to integers, and this conversion step may drop a very small number of malformed rows. In addition, the ordering of group-by and reduction operations in a distributed setting can marginally affect how duplicate or invalid combinations are handled. Crucially, these implementation-level nuances do not alter the empirical meaning of the graph: Spark reproduces the same sparsity patterns, the same dominance of weight-one edges, and the same thin tail of stronger co-occurrences that characterize the Python version. Although Spark introduces some workflow overhead for datasets of this size—chiefly due to the distributed shuffle required when grouping by user and by book pairs—the experiment demonstrates that the distributed construction behaves consistently with the local approach and scales naturally to substantially larger datasets. The fact that Spark accurately recovers the co-occurrence signal, even under parallel execution and with millions of intermediate key–value pairs, confirms its suitability as a foundation for link-analysis tasks on genuinely massive data, far beyond the limits of single-machine memory.

Conclusions

This project shows how a classical link-analysis method such as PageRank can be adapted to the structure and constraints of a large-scale recommendation setting. Starting from the Amazon Books Review dataset, a simple co-occurrence model between books is used to construct sparse but informative graphs, where a handful of canonical titles act as hubs connecting otherwise fragmented user histories. The analysis demonstrates that, even in this highly unbalanced environment, PageRank converges rapidly and produces rankings that are both interpretable and stable across graph sizes, consistently highlighting books that bridge diverse reading patterns rather than merely reflecting raw popularity. From a computational perspective, the experiments reveal a clear separation between the cost of graph construction and the cost of ranking. Generating co-occurrence edges grows superlinearly with the number of users and quickly becomes the dominant component, while PageRank itself remains comparatively inexpensive, with subsecond runtimes even on graphs with more than 10,000 nodes and nearly 60,000 edges. The Spark implementation further confirms that the same co-occurrence logic can be reproduced in a distributed environment, preserving sparsity patterns and edge weights while providing a viable path toward genuinely massive datasets that exceed single-machine memory. The co-occurrence graph deliberately ignores review scores focusing instead on a purely structural notion of proximity. Future work could enrich the model by integrating rating information, time-aware dynamics or genre metadata, and by exploring personalized or topic-specific PageRank variants. Nonetheless, within its chosen scope, the project provides a transparent and scalable blueprint for applying PageRank to large recommendation datasets and for assessing how implementation choices affect both interpretability and computational feasibility.