Link Analysis with Topic-Sensitive PageRank: finding the most authoritative users on Amazon Books Reviews dataset

Emanuele Giavardi
Department of Computer Science, University of Milan

June 9, 2025

1 Introduction

This report aims to present the work carried out for the final project of the Algorithms for Massive Datasets course, taught by Professor Dario Malchiodi at the University of Milan. The focus is on the implementation of the Topic-Sensitive PageRank algorithm and its application to a graph built from the "Amazon Books Reviews" dataset, publicly available on Kaggle, which contains reviews and related information about a wide variety of books. The data used refers to version 1 of the dataset, which was last updated in 2022.

The core idea behind this work is to identify the most authoritative or influential users within a specific literary genre. To achieve this, the original dataset was processed to consider pairs of users who reviewed the same book. Based on this information, a graph was constructed to represent the network of user-to-user review relationships.

Although the project was developed on Google Colab, the implementation was designed to run in a distributed environment, where operations can be performed across the physical nodes of a cluster. For this reason, both the preprocessing operations and the ranking computation were carried out using the Spark Engine [1], specifically through the PySpark library.

2 Graph construction

The dataset includes two tables: the first one, *books_ratings*, contains information about user reviews. Among the most relevant attributes for this project are the user's ID who wrote the review, the book being reviewed, and a helpfulness score for the review. The second table, *books_data*, contains information related to books, including the various literary categories associated with each one of them.

The graph on which the ranking scores were computed was built from these two tables according to the following logic: nodes represent users, and an edge connects two nodes if the corresponding users reviewed the same book. More specifically, the graph is directed: an edge from node $u2$ to node $u1$ exists if $u1$ and $u2$ both reviewed the same book and the helpfulness score of $u1$'s review is higher than that of $u2$ for the same book.

It should be noted, however, that only reviews related to a subset of all available books were considered: the literary categories in the *books_data* table include:

- in many cases, a string representing a list of genres (enclosed in square brackets), such as `"['Religion', 'Politics', ...]"`
- in some cases, invalid or irrelevant data, such as the value `'None'` or links to the Google Book Store

Since each user was assigned the literary genre they reviewed the most as the topic for the computation of Topic-Sensitive PageRank, it was important to ensure the use of meaningful data for this aspect. Therefore, only those reviews for which the "literary category" attribute could be parsed into a valid string of values were considered, in order to exclude noisy or irrelevant data.

The pseudocode for graph creation is as follows:

```
for each book b (with well-formatted literary categories):
    for each (u1, u2) such that both u1 and u2 reviewed b:
        if (helpfulness(u1, b)) > (helpfulness(u2, b)):
            add edge from u2 to u1
```

The helpfulness scores of each review are not on a common scale, as they appear in formats like `"0/0"`, `"4/5"`, `"78/82"`, and so on. For the purpose of this project, we simply convert each `"x/y"` string into a float value by evaluating the fraction. However, a more refined interpretation would consider `"x/y"` as "number of people who found the review helpful/total number of voters". Under this assumption, it is important to account not only for the ratio itself but also for how many people voted, as a high ratio based on a small number of votes may be less reliable than a slightly lower ratio based on many votes.

3 Data preprocessing

3.1 Subsampling

Since the project was carried out on a single machine rather than in an actual cluster, to ensure a reasonable code execution time, the dataset was sampled by retaining a fixed fraction p of rows from the total. Experiments were conducted with $p = 0.01$, using 1% of the available data, resulting in 30000 rows out of the initial 3000000, from which a graph with 15435 nodes and 2411934 edges was built.

In the original dataset, the reviews are ordered by the reviewed book. This means there are K rows related to book B, followed by K' rows associated with book B', and so on. Therefore, there are two possible strategies for sampling the dataset:

- Selecting the first total_rows$\cdot p$ rows without any shuffling: this approach favors the inclusion of reviews for the same book, resulting in a smaller number of distinct books but a higher number of user-user connections.
- Sampling rows uniformly at random, where each review is included in the sample with probability p (using a random seed for reproducibility): this method leads to a broader variety of books in the sample, but it becomes less likely that two users have reviewed the same book, thus reducing the number of connected users.

Although the second approach may be more statistically appropriate in terms of representative sampling, the first strategy is chosen for this project to ensure a denser graph with a more substantial number of user connections, since the goal of this work is to explore links between users who have reviewed the same book.

3.2 Tables manipulation

Preparing data for graph construction, the following procedure was applied: given R, the *books_rating* table, stored as a Spark DataFrame, where only the relevant columns (book title, user, and helpfulness score) were retained, a self-join was performed on R to extract all review pairs where two distinct users $u1$ and $u2$ reviewed the same book and the helpfulness score of $u1$ was greater than that of $u2$.

The resulting table was then filtered to keep only those reviews for which the book's category matched a valid format, i.e., a list of strings.

At this point, the N unique users in the resulting table were mapped to contiguous integer values from 0 to $N-1$. This operation is motivated by two factors: firstly, PageRank values can be stored in an array V of N elements, such that $V[i]$ is the PageRank score of the user associated with the integer i. Secondly, by iterating through the table mentioned above, it is possible to represent each edge from $u2$ to $u1$ simply by storing the integer pair (i, j), where i is the integer associated with user $u2$ and j is the one associated with $u1$.

As a final preprocessing step, starting from *books_data* table (also stored in a Spark Dataframe), a vector was created such that position k of this vector contains the most frequently reviewed genre by the user mapped to integer k. This step is crucial to ensure that each node in the graph is associated with a topic that can be taken into account during PageRank computation. In this specific context, the idea is that each user has a "favorite literary genre", or at least a genre in which they are more authoritative.

All these operations rely on an important assumption: the number of unique users N has to be small enough to allow both the data structure (specifically, a dictionary) that maps user IDs to integers, and the array storing each user's preferred genre, to be held in main memory.

4 Ranking Computation

PageRank [2] is an algorithm originally developed for the web, designed to measure the importance of web pages by leveraging the structure of hyperlinks between them.

It models the behavior of a random surfer who jumps from one page to another. Each page is represented as a node in a directed graph, with edges corresponding to hyperlinks. The algorithm works with a transition matrix M, where each element represents the probability of moving from one page to another by following a link, obtained by normalizing the columns of the adjacency matrix. More formally, given a generic element m_{ij} of the matrix:

$$m_{ij} = \begin{cases} \frac{1}{\alpha}, \text{ if there is an edge from } j \text{ to } i \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

where α is the number of outgoing edges from node j.

The ranking scores associated with the N nodes of the graph are stored in a vector v of N elements, initialized with the value $\frac{1}{N}$ to ensure an initial uniform score distribution. The algorithm then proceeds iteratively by multiplying the matrix M by the vector v obtained from the previous iteration. Also in this case, we assume this vector to be sufficiently small to be stored in main memory, so that it can be broadcasted to all the nodes of the cluster, which will therefore have it entirely available during processing.

In this specific project, it is common to have multiple edges from a given node j to another node i, since there may be several books for which a user has written better reviews than another user. For this reason, all the b possible edges from j to i are collapsed into a single edge with weight b, and the corresponding transition matrix element m_{ij} is set to $\frac{b}{\alpha}$.

This small adjustment does not alter the probabilistic interpretation of the matrix columns, which retain the crucial property of being *column-wise stochastic*, since the values in each column sum to one. This characteristic is fundamental because it ensures that, after a certain number of iterations, the product $M \cdot v$ converges and the algorithm reaches a valid stopping condition.

It is well known that the basic version of PageRank is prone to certain issues due to structural patterns that may arise within the graph. It is therefore important to take into account that in this specific graph:

- **Dead Ends**, which are nodes with no outgoing edges, may appear. This occurs when a user has written better reviews than all other users with whom they shared books, never being outperformed. In such cases, the "literary authority" score could quickly concentrate solely on that user.
- **Spider Traps** may also exist. In the simplest case (involving just two nodes), two users may have mutually better reviews on different books, and there are no edges to or from other users. This forms an isolated cycle in which the PageRank score gets trapped

Within the data sample used for testing, no dead ends and just one two-node spider trap were identified, but such structures may likely occur more frequently when using the full dataset.

To address this issue, a mechanism known as *taxation* was introduced, which involves a damping factor β. Under this mechanism, the random surfer, figuratively

placed on a certain node, follows one of the outgoing links with probability β and "jumps" randomly to another node with probability $1 - \beta$.

In the **Topic-Sensitive** version of PageRank, this very last step is modified: with probability complementary to the damping factor, the random surfer does not choose a new node uniformly among all nodes, but rather among those associated with a specific topic. This contributes to increasing the PageRank score of those nodes. In the context of this project, if the goal is to evaluate, for example, the most authoritative reviewers in the "Politics" genre, the PageRank score will favor those users who have reviewed political books more than any other genre.

Considering all of this, the operation at the core of the PageRank version implemented in this work is:

$$\begin{cases} v(0) = \frac{1}{N}\underline{1} \\ v(t+1) = \beta M v(t) + (1-\beta)\frac{e_S}{|S|} \end{cases} \tag{2}$$

where S is a given literary genre, $|S|$ is the cardinality of the set of users interested in S, and e_S is a vector such that

$$e_S[i] = \begin{cases} 1 & \text{if the user associated with } i \text{ value has } S \text{ as most reviewed genre} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

5 Results

The ranking scores were computed both using the base version (without topics) of PageRank and on the three different topics: "Education", "Religion", and "Fiction". For each of the four cases, a maximum number of 100 iterations was set, along with a tolerance threshold of 10^{-5}, below which the difference between the PageRank vector at time $t-1$ and that at time t becomes negligible, allowing the algorithm to meet the stopping condition. The damping factor β was set to 0.8.

Table 1 shows the number of iterations and convergence times for each run, while Table 2 lists the top 5 users with the highest ranking scores in each of the four scenarios.

Topic	**Iterations**	**Convergence time (s)**
None	5	28.8
Fiction	5	28.7
Education	6	31.8
Religion	7	36

Table 1: Number of iterations and convergence time for PageRank computation in each experimental setting

Evaluating the quality of the ranking is rather challenging, since there is no definitive ground truth to which to compare the results. However, it is still possible to attempt to infer some information based on external knowledge, which may indicate whether the results are reasonable or not.

	None	Fiction
1	Midwest Book Review	Midwest Book Review
2	E. A Solinas	E. A Solinas
3	Terri J. Rice	Harriet Klausner
4	Harriet Klausner	Terri J. Rice
5	BooksForABuck	BooksForABuck
	Education	**Religion**
1	Lois P. Miller	D. M. Childs
2	George Zee	R. Okarski
3	R. Fitzgerald	Nathan Shattuck
4	Midwest Book Review	K.H.
5	PhD student	theotokos

Table 2: Top five users for each experimental setting

For instance, the user with the highest authority score in the Non-Topic-Sensitive PageRank (who also appears in the top 5 for the "Fiction" and "Education" topics) is "Midwest Book Review", which is a fairly well-known organization focused on book reviews. Therefore, at least in this example, the outcome of the PageRank algorithm appears to be quite reasonable.

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. 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. No generative AI tool has been used to write the code or the report content.

References

[1] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford infolab, 1999.