Link Analysis Project

Algorithms for Massive Datasets

Student

Name: Francesco
Surname: Ferrulli

1 Introduction

The project involves a link analysis of the Amazon Books Reviews dataset to find relationships between entities modeled in a graph.

The dataset is divided into two files:

- `books_rating`: contains 3 million reviews from users on 212,404 unique books.
- `books_data`: contains information about the 212,404 unique books.

The dataset is available on Kaggle. The version of the dataset is version 1.0, updated 3 years ago.

2 Dataset Preprocessing

For the purpose of this project and due to computational limits on Google Colab , a sampled version of the original dataset was used. In particular, a subset of 300,000 rows was extracted from the books_rating dataset in order to reduce computational costs.

After downloading the dataset, some few steps of preprocessing were made:

1. A sample of 300,000 rows was extracted from the original dataset in order to reduce computational complexity.
2. For the reviews dataset, only the columns `Id`, `Title`, and `User_id` were retained, discarding all other attributes.
3. All rows containing null values in any of the selected columns (`Id`, `User_id`, `Title`) were removed.
4. For the books dataset, only the columns `Title` and `categories` were selected.
5. Rows containing null values in either `Title` or `categories` were removed.

Note: for the topic-sensitive pagerank, in order to get the category for each book, a join on the attribute `Title` was made, since the books dataset doesn't have an `Id` or `ISBN` attribute. Since the `Title` attribute is not as reliable as an unique identifier, before joining, the `Title` field was normalized. In particular, unnecessary whitespace characters were removed and all text was converted to lowercase

3 Generating the graph

The ranking task is performed on a graph whose entities are books. Formally, we define an undirected weighted graph $G = (V, E, W)$, where:

- V is the set of books;
- $E \subseteq V \times V$ is the set of edges connecting books;
- W is the set of edge weights.

Two books are connected by an edge if they have been reviewed by at least two different users. Therefore, an edge $(i, j) \in E$ exists if there are at least two users who reviewed both book i and book j.

The weight associated with an edge represents the number of users who reviewed both books. Only pairs with $w_{ij} \geq 2$ are retained.

To generate the graph, the first operation was to generate book pairs by performing a self-join on the reviews dataset using the `User_id` as key. This operation creates all pairs of distinct books reviewed by the same user. After generating the pairs, a grouping operation was performed, considering as key the pair itself, in order to count the number of users who reviewed the same pair of books. This defines the weight of the edge.

Pairs with weight lower than two were discarded. To allow an efficient computation, each book was assigned a unique integer identifier.

The graph represents a co-review network and can be seen as a undirected graph, since the relationship between two books is symmetric. However, for implementation purposes, edges are seen as ordered pairs, which allow the application of algorithms.

The generated graph comprises 5641 nodes and 148800 edges.

4 Ranking Algorithms

The proposed ranking algorithms are PageRank, Topic-sensitive PageRank and HITS. Before discussing these algorithms, first let's discuss how data are organized.

4.1 Adjacency Matrix

Let $G = (V, E, W)$ be the weighted graph defined in the previous section, with $|V| = n$ books.

The graph is represented through its weighted adjacency matrix $A \in \mathbb{R}^{n \times n}$, defined as:

$$A_{ij} = \begin{cases} w_{ij} & \text{if there exists an edge from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

where w_{ij} is the number of users who reviewed both books i and j.

Since the graph is sparse, the matrix is stored using a sparse representation. Each row corresponds to a source node and contains only the non-zero weighted edges toward its neighbors.

To apply ranking algorithms, the adjacency matrix is transformed into a row-stochastic matrix P, obtained by normalizing each row:

$$P_{ij} = \frac{A_{ij}}{\sum_k A_{ik}}$$

This ensures that:

$$\sum_j P_{ij} = 1$$

for every node i with at least one outgoing edge.

Finally, since the importance of a node depends on the nodes pointing to it, the transpose P^T is used in the ranking computations.

4.2 PageRank

PageRank assigns an importance score to each node based on the structure of incoming links. Let $r \in \mathbb{R}^n$ be the ranking vector.

The iterative PageRank is based on the following formula:

$$r^{(t+1)} = \beta P^T r^{(t)} + (1 - \beta)\frac{1}{n}\mathbf{1}$$

where:

- P^T is transposed adjacency matrix;
- $\beta \in (0, 1)$ is the damping factor;
- $\mathbf{1}$ is the vector of all ones;
- n is the number of nodes.

The algorithm iteratively updates the ranking vector until convergence (or until the maximum number of iterations has been reached).

In Spark, the ranking vector is distributed across workers via the `broadcast` method, and each iteration consists of a distributed multiplication between the transposed stochastic matrix and the current rank vector. Each worker computes partial contributions to the rank of destination nodes based on its assigned matrix partitions. These partial results are then aggregated through shuffle operations.

4.3 Topic-sensitive PageRank

In Topic-sensitive PageRank, the teleportation vector v replaces the uniform distribution with a topic-dependent probability vector.

The implementation of Topic-Sensitive pagerank is the same as the normal pagerank; it only affects the described vector in the formula, while the matrix-vector multiplication remains unchanged.

For the project, the topic used for Topic-Sensitive PageRank was the book category. The information was obtained by linking the reviews dataset with the books dataset, where the `categories` attribute is available.

An alternative topic could have been the helpfulness of reviews, since the reviews dataset also includes an attribute measuring the helpfulness rating of the review.

In particular, the selected topic was "Religion" books. Since the `Title` attribute is not a reliable unique identifier (e.g due to formatting inconsistencies), a normalization step was applied before joinning the datasets: titles were converted to lowercase and leading/trailing whitespace were removed. The reviews dataset was then joined with the book dataset on the normalized title field in order to associate each book with its corresponding category.

Books belonging to the selected topic were identified by filtering the `categories` attribute using a case-sensitive matching rule. The personalization vector $v \in \mathbb{R}^n$ was constructed as:

$$v_i = \begin{cases} 1 & \text{if book } i \text{ belongs to the selected topic} \\ 0 & \text{otherwise} \end{cases}$$

Finally, the vector was normalized to ensure that:

$$\sum_{i=1}^{n} v_i = 1.$$

This normalized vector was used in the Topic-Sensitive PageRank update rule:

$$r^{(t+1)} = \beta P^T r^{(t)} + (1 - \beta) v$$

4.4 HITS

The HITS algorithm assigns two distinct scores to each node:

- an authority score a_i;
- a hub score h_i.

The idea behind HITS is that a good authority is a node pointed to by good hubs, and a good hub is a node that points to good authorities. Therefore, authority and hub scores reinforce each other through an iterative process.

Let A be the adjacency matrix. The iterative update rules are:

$$a^{(t+1)} = A^T h^{(t)}$$

$$h^{(t+1)} = A a^{(t+1)}$$

In matrix form, this corresponds to:

$$a = A^T A a$$

$$h = A A^T h$$

Thus, authority scores correspond to the dominant eigenvector of $A^T A$, while hub scores correspond to the dominant eigenvector of AA^T.

In the context of the book co-review network, an authority book is one that is strongly connected to highly connected books. A book receives a high authority score if it is co-reviewed with books that are themselves important hubs. A book has a high hub score if it is connected to many authoritative books.

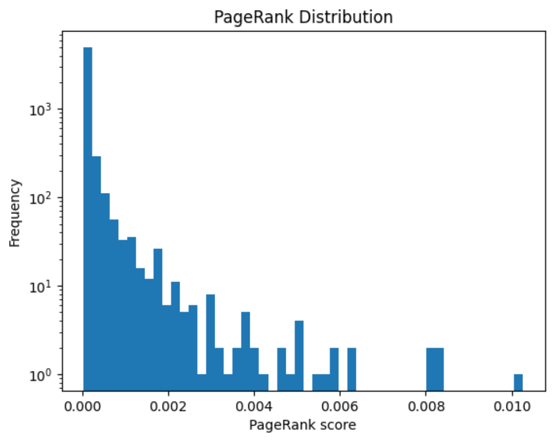
The HITS algorithm is implemented as an iterative procedure, like PageRank, based on matrix-vector multiplications. Each iteration requires two operations:

$$a^{(t+1)} = A^T h^{(t)}, \quad h^{(t+1)} = A a^{(t+1)}.$$

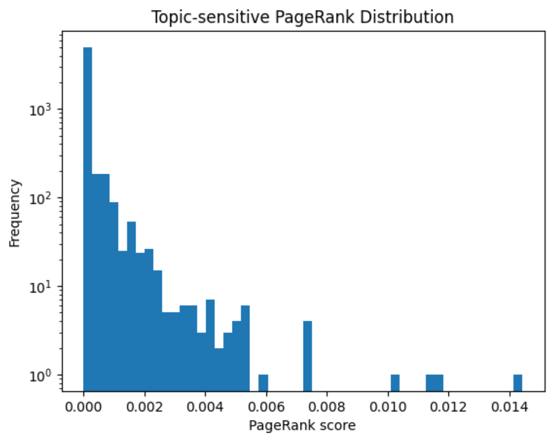
The algorithm iteratively updates and normalizes both vectors until convergence or the maximum number of iterations has been reached. Both vectors are normalized to prevent numerical instability.

5 Results

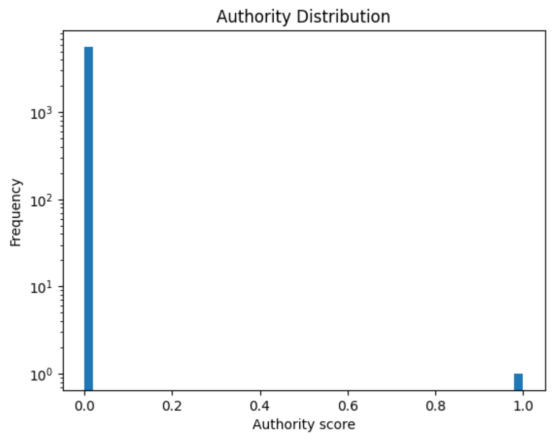
These are the corresponding distributions for each ranking algorithm.



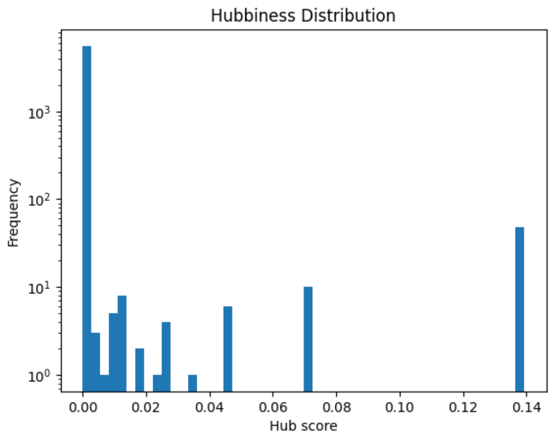
PageRank



Topic-Sensitive PageRank



Authority Scores



Hub Scores

The execution time for each algorithm is shown in the following table:

Algorithm	Execution Time
PageRank	2min 56s
Topic-Sensitive PageRank	2min 35s
HITS	8min 55s

Table 1: Execution time comparison of ranking algorithms.

The distributions of the ranking algorithms differ from each other, especially PageRank and HITS.

In PageRank there is a smoother distribution of scores compared to HITS. This is likely the effect of teleportation mechanism, which prevents excessive concentration on a node or a set of nodes. Similarly, the Topic-Sensitive PageRank presents a smooth distribution with minor changes due to the bias of the ranking toward the selected topic. It can be seen that the books with the highest value change in the topic-sensitive pagerank, being in the specified category..

In contrast, HITS (particularly authority scores) shows a more extreme concentration of values. The top authority book "Anima Gospel" receives a score approximately 0.9998, while all other nodes receive much smaller values . This is because HITS can amplify dense substructures in the graph (for the absence of a teleportation method) and the dominant eigenvector concentrates almost entirely on a single dominant node.

The Spearman's rank correlation coefficient was used to compare the different ranking strategies. This coefficient measures the monotonic agreement between two ranked variables, and it operates on node positions rather than raw scores.

The correlation between PageRank and Topic-Sensitive PageRank is moderate ($p \approx 0.4722$), meaning that while the topic bias modifies the ranking, a substantial part of the global structure is preserved.

The correlation between PageRank and HITS Authority is lower ($p \approx 0.3370$), reflecting the differences between the two algorithms. PageRanka incorporates a teleportation mechanism for random walks, while HITS doesn't.

The correlation between Authority and Hubbiness relatively high ($p \approx 0.5238$). Even if the co-review network is conceptually undirected, the normalization introduces asymmetry, preventing Authority and Hub score from being identical.