Algorithms for Massive Data
Project: Link Analysis

by Enrico Darra, 21571A

Università degli Studi di Milano

Data Science for Economics

September 13th, 2024

ABSTRACT

In this project, two variations of the PageRank algorithm, unweighted and weighted edges, were applied to a dataset of paintings from the Prado Museum. The goal was to rank the paintings based on their relationships, where connections were defined by shared tags between artworks. Each painting was considered as a node in a graph, and edges between these were formed based on the shared tags. The unweighted PageRank algorithm treated all connections equally, while the weighted PageRank variation assigned more importance to links with more common tags, providing a nuanced ranking system. The implementation utilized PySpark's distributed computing capabilities to efficiently handle the large dataset and create scalable solutions.

The results showed that both algorithms ranked the same painting *"Retrato mortuorio del periodista Pedro Avial Taracena" by* José Nin y Tudó at the top, although the remaining rankings varied between the two approaches; interestingly, the weighted PageRank algorithm favored several paintings by Francisco de Goya.

I/We declare that this material, which I/We now submit for assessment, is entirely my/our 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/our work, and including any code produced using generative AI systems. I/We 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/us or any other person for assessment on this or any other course of study.

CONTENTS

1. INTRODUCTION

The PageRank algorithm, originally developed by Google, is a link analysis algorithm designed to rank web pages based on their relative importance, where the importance is represented by the number of other web pages linking to a specific one. In essence, PageRank works by measuring the probability that a random web surfer would visit a specific page, assuming they either click links from one page to another or randomly "teleport" to any page in the network. The algorithm keeps assigning iteratively a rank to web page and the PageRank values converge over a certain number of iterations. PageRank has since been extended to various sectors and has proven itself to be an effective tool for ranking vertices in large-scale networks.

In this project, the PageRank algorithm was applied to the dataset of paintings from the Prado Museum, in Madrid. The Prado Museum is one of the most famous and visited art museums in the world, with over 3 million visitors per year: its collections ranges from ancient to contemporary art, with an extensive collection of European art pieces[1].

The goal of this project was to rank these paintings based on their connections with each other, where a connection is represented by common tags between the two paintings. In practice, each painting can be seen as a node in a graph, with relationships between paintings forming the edges. Two approaches have been implemented: the traditional PageRank and a weighted PageRank. The weighted PageRank variation assigns more importance to certain edges based on the amounts of tags that the two paintings share, providing a more nuanced ranking system rather than assigning equal probability to the random surfer of following the out-links.

The implementation was carried out using the PySpark framework, which enabled efficient processing of this large dataset by leveraging its map-reduce functions and the Resilient Distributed Dataframe.

[1] https://www.museodelprado.es/en/museum

2. DATASET

The Prado Museum dataset, available on Kaggle[2], has been presumably constructed through web scraping of each art piece information on the Prado Museum website. The dataset overall composed of 13 487 rows, each representing a painting, and 30 columns. The dataset contains several information, such as the painting title, the author, a description of the painting, techniques of realizations, materials, dimensions, a link to the painting web page, and a tags list.

This tags list consists of tags, or labels, that are used to classify the paintings and can be in regards of the art style, the subjects depicted, the materials or way of realization, the period during which the painting has been made and so on. Here follows an example of tags for the first painting in the dataset:

Author - Title	**Tags**
Bayeu y Subías, Francisco – "*Cabeza de Gigante*"	• Serie de dibujos para el fresco la Caída de los Gigantes en el Palacio Real de Madrid • Lápiz negro • Papel verdoso • Estudio de cabeza • Mitología • 1764 • +

These tags are of primary focus in this report because these allow us to create connections between the paintings and, effectively, consider this data as a graph with nodes and edges.

In our dataset, we have 3548 unique tags, with the most frequent tags among paintings being: '+' for 12 993 paintings; '*Óleo'* for 4 177; '*Lienzo'* for 3 225; and '*Papel verjuardo*' for 1 477. Not accounting for the plus sign tag, it makes sense that the most used tags are about the popular painting technique of oil (*Óleo*) on canvas (*Lienzo*).

[2] https://www.kaggle.com/datasets/maparla/prado-museum-pictures

3. DATA PRE-PROCESSING & EDA

In this part, we see the data pre-processing, the creation of the edge list needed to transform the Prado Museum dataset into a graph structure suitable for the PageRank algorithm and an overview of the graph nodes degrees. Pre-processing involved cleaning the data, extracting the relevant columns, and building an edge list that represents relationships between paintings based on shared tags. From the Jupyter notebook in my Github repository[3], it is possible to see the preprocessing steps that have been taken.

3.1. Tags analysis

Firstly, I uploaded the dataset as a pySpark Dataframe to perform the initial steps. As a first step, the column containing the tags needed to be converted to an actual array of tags, from the initial string containing all the tags.

Checking the most frequently used tags, described in the section above, it is possible to see that the most common tag is the plus sign '+'. Investigating the Prado Museum website, I could establish that this tag is not really a label used to categorize art. It is in fact a button that the webpage user can click on to display more tags; hence I presumed that '+' has made its way into the dataset because of the way the web pages have been scraped for information by the creator of the dataset. Since this tag has no relation to the type of art, I decided to drop this particular tag from the dataframe, leaving us 3 547 unique tags.

Specifically, the average tags frequency is 22.86. Moreover, looking at the Boxplot in Figure 1, we can see that besides a few outliers, the majority of tags appear in just a few paintings.

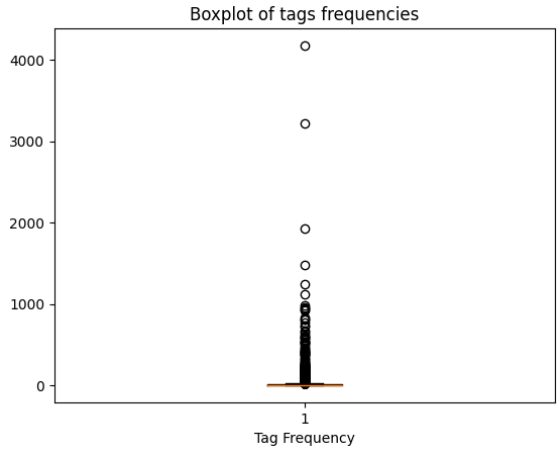

Figure 1: *Boxplot of tags frequencies (the tag + has already been removed).*

3.2.Graph creation

[3] https://github.com/Enri98/pageRank-prado

The PageRank algorithm is typically applied to directed network. Regarding our data, the way by which we define the network does not contain any information about edge direction: our network is undirected. In fact, two paintings are linked if they share one or more tags, without one of these paintings having precedence over the other. For this reason, I represented the graph, undirected by nature, as a directed graph with two edges of opposite directions for each link between the nodes.

For the purpose of this project, it is not necessary to represent the network of paintings using the specific libraries for storing such data structures. Hence, I represented the network simply as an edge list.

Firstly, I introduced a proxy ID for each node as a numerical integer, for the reason of having a more immediate representation of the node's connection. After that, the only columns required to build the edge lists are the node ID and the tags list.

To identify the connections between paintings sharing a tag, an immediate approach could be exploding the tags list column, having multiple rows for a painting with a different tag each, then join the exploded dataframe on itself on the tags columns: by selecting the columns of node IDs from the two dataframes (itself and a copy of it), we would have our start and end node. However, seeing the dimensions of the dataframe, a join operation can require quite a lot computing effort.

The actual approach used for gathering the edge information has been, after exploding the dataframe by the tags list column, performing a group by on the tags and computing the combinations of all two paintings that are contained in a specific tag's group. In practice, an edge dataframe is generated with a column for start and end nodes where each linked nodes pair appears twice to represent an edge in both directions. Moreover, the duplicates rows mean that a particular pair of paintings shares more than one tag; I dropped the duplicates, but saving the information of multiple tags shared to use it later in the weighted PageRank approach. The final number of edges is 36 145 728.

3.3.Nodes Degree

To better understand the structure of the graph, the degree distribution of the nodes was calculated, depicted in Figure 2 as the total node's degree given by the sum of in-degree and out-degree.

The node with the lowest degree has 26 connections, or an in-degree and out-degree of 13. Hence, considering this together with the fact that, by design, every node must have both in-links and out-links, the presence of possible spider traps and dead-end nodes can be

excluded. In fact, these two situations could have presented some difficulties to the PageRank algorithm.

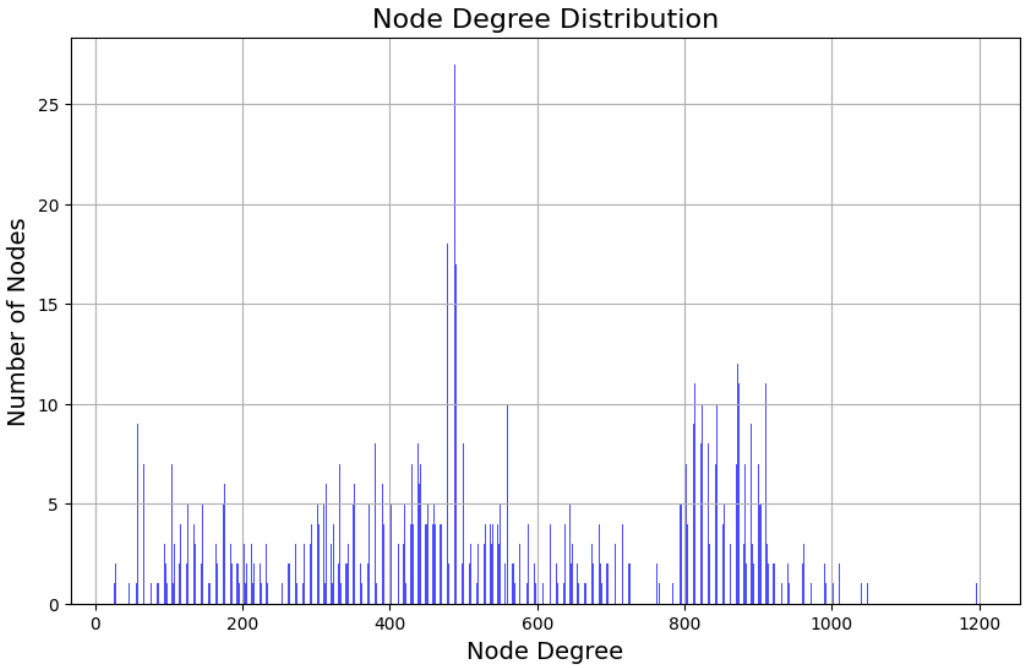

Figure 2: Nodes degree distribution

4. PAGERANK

In the following section, the theoretical aspects of the algorithms are presented; then there is an overview of the ways in which PageRank has actually been implemented in the context of this project.

4.1. PageRank Algorithm

As mentioned in the introduction, PageRank is an algorithm first developed by the Google founders to evaluate the importance of web pages. After that, it has been imported in several other domains of different nature as a measure of "prestige" of nodes.

The creation of such algorithm was the answer to the problem of the so-called term spam, i.e. spammers were exploiting the way searching engines worked: by listing the terms inside web pages and providing the web page top inverted rank to a search query containing the terms. The way PageRank works is by simulating where web surfers would group after beginning on a random web page, moving through linked pages randomly. The importance of

the page is defined by how many surfers they would encounter. Another key aspect of PageRank, that allowed to give a challenging time to spammers, was in fact that queried terms were given importance also when present in neighboring web pages. In this way, the importance of pages is passed through its out-links to other pages, hence, spammers could not just create helper pages because these would be without in-links.

PageRank is substantially a function that assigns an "importance" value to each node (web page), the higher the value the more important the node in the network. The importance of a node is passed along to its neighbor nodes following the direction of edges. In fact, the graph is depicted as a directed graph. From the network structure, a Transition Matrix is created, that is a square matrix where the element m_{ij} for row i and column j has value $1/k$ where k is the number of out-links of node j. Essentially, the columns of the Transition Matrix represent the probability distribution that a random surfer finds themselves at a certain node.

If two conditions are met, graph is strongly connected and absence of dead-end nodes, the property of PageRank ensure a convergence of the distribution vector: $\boldsymbol{v} = M\boldsymbol{v}$ where M is the Transition matrix and $\boldsymbol{v}$ the distribution vector. Moreover, since M is row-stochastic, $\boldsymbol{v}$ is the principal eigenvector of M.

To overcome the problem of dead-end nodes and spider traps (group of nodes with only in-links and no out-links to the rest of the graph), the so-called taxation parameter is introduced when computing the PageRank values. Basically, we introduce a probability that the surfer teleports to a random node instead of following the out-links.

Finally, the steps to compute the PageRank of node v_i can be summarized by the formula, where d is the taxation parameter:

$$PR(v_i) = d \cdot \sum_{v_j \in in-Neighbors(v_i)} \frac{PR(v_j)}{Count(out-Neighbors(v_j))} + \frac{1-d}{N}$$

4.2. PageRank Implementation

The Transition Matrix for our dataset is quite big: $13\,487 \times 13\,487$ and, being the Transition Matrix sparse, it is not efficient to store all the zero values when there are no links between the nodes. For this reason, a better approach to store the linkage information is the adjacency list.

To compute the PageRank, we developed a function that takes as input the edge dataframe in the form of RDD together with a parameter for maximum iterations,

convergence distance and taxation parameter, leveraging the map-reduce methods of pySpark, computes the PageRank values.

Firstly, the function condenses the edge information from the edge RDD into an adjacency list. Then, the initial values of the PageRank vector are set to be uniform, i.e. 1 divided by the number of nodes. Then a loop is started. Inside the latter, the contributions of the PageRank values from the in-Neighboring nodes are computed using the adjacency list and the previous iteration PageRank values. Finally, the PR value of each node is computed by applying the aforementioned formula.

During each loop, the Euclidean distance between the current PageRank vector and the one computed in the previous iteration is computed and checked against a tolerance parameter. When this distance becomes smaller than the tolerance, I consider the PageRank values to have sufficiently converged, since the PR vector has not changed much from the previous iteration.

The output of the function is the PageRank vector with the values of the last iteration. The results of the algorithm implementations are discussed in the Results section.

5. WEIGHTED EDGES PAGERANK

In this section, another approach at the PageRank algorithm is presented: PageRank with weighted edges.

As previously mentioned, in our dataset there are paintings that have more than one tag in common. In this sense, we could interpret this as the pair of paintings having a stronger link. I have decided to take this information into account when computing the PageRank values by using the number of shared tags as weight. Then, when the contributions given by a certain node to its out-link neighbors are computed, these contributions will no longer be uniform across its neighboring nodes but proportional to the edge weights.

5.1. PageRank - weighted edges implementation

Our new function that computes the PageRank with weights on edges is similar to the previous one with one main difference. The edge RDD taken as input, besides the two columns with source and destination nodes, presents a third column with the weights. These weights have been previously normalized such that the sum of all weights of the out-links of a node is equal to 1. Like the previous PageRank function, from the edge RDD, the adjacency

list is computed but this time it contains also the weights of each link. When computing the contributions of the PageRank values given to a certain node by its neighbors, we no longer divide the neighbors PR values by the number of out-links but multiply instead the PR values by their respective edge weight. This process is summarized in the following formula, where w_{ji} represents the weight on the edge from node j to node i:

$$PR(v_i) = d \cdot \sum_{v_j \in in-Neighbors(v_i)} PR(v_j) \cdot w_{ji} + \frac{1-d}{N}$$

Similarly to the previous function, a convergence check is performed by confronting the Euclidean distance between current and previous iterations PageRank vectors and the output is, in fact, the last PageRank values computed.

6. RESULTS

In this section, the results of the two PageRank implementations are discussed, as well as parameters set for the two executions.

6.1. PageRank

The unweighted PageRank function has been executed with a maximum number of iterations of 50, but it showed to converge after just 22 iterations. The tolerance given as input to approximate the convergence was $1e-6$ and the Euclidean distance to measure the convergence was L2-norm. The taxation parameter was 0.85, since commonly values in the range of 0.8-0.9 are used. In the following table, the top 10 paintings with the highest PageRank values are presented.

Author – Title of the painting	**PageRank value**
Nin y Tudó, José - "*Retrato mortuorio del periodista Pedro Avial Taracena*"	0.00015103
Anónimo - "*San Joaquín, Santa Ana y la Virgen Niña*"	0.00015027
Sánchez Perrier, Emilio - "*Berlina tirada por un caballo en Sevilla*"	0.00014948
Reynolds, Sir Joshua - "*Retrato de James Bourdieu*"	0.00014937
Anónimo (Obra copiada de: El Greco) - "*San Francisco en oración*"	0.00014747
Creti, Donato - "*Estudio de guerrero romano*"	0.00014429
Hernández Nájera, Miguel - "*Víspera del Dos de Mayo*"	0.00014112
Jover y Casanova, Francisco - "*Tratado de Cambray*"	0.00013979
Oliva y Rodrigo, Eugenio - "*Cervantes, en sus últimos días, escribe la*	0.00013889

dedicatoria del Quijote al conde de Lemos"	
Goya y Lucientes, Francisco de - "*Cabeza de asno de perfil, relinchando*"	0.00013832

6.2. PageRank – weighted edges

To keep a consistent process, and compare results, also the weighted PageRank function as been given the same input of 0.85 for the taxation parameter, $1e-6$ for the tolerance value, and 50 as the maximum number of iterations. Specifically, the algorithm converged after 24 iterations. Below, a table with the top 10 ranking paintings is presented.

Author – Title of the painting	**PageRank value**
Nin y Tudó, José - "*Retrato mortuorio del periodista Pedro Avial Taracena*"	0.00014251
Reynolds, Sir Joshua - "*Retrato de James Bourdieu*"	0.00014012
Anónimo (Obra copiada de: Velázquez, Diego Rodríguez de Silva y) - "*Francisco de Ocáriz y Ochoa*"	0.00013815
Ribera y Fieve, Carlos Luis de - "*Busto de Rodrigo Calderón*"	0.00013707
Goya y Lucientes, Francisco de - "*Anotación sobre la boda del artista. Referencia a cantidades pecuniarias. Santa Bárbara, contradibujo*"	0.00013595
Goya y Lucientes, Francisco de - "*El sueño de san José o La muerte de san Francisco Javier (estudio preparatorio). Anotación de pago a Tomás Goya. Apuntes de motivos vegetales*"	0.00013595
Goya y Lucientes, Francisco de - "*Virgen con el Niño sentado en su regazo, enmarcada en un óvalo*"	0.00013595
Goya y Lucientes, Francisco de - "*Y no hai remedio*"	0.00013531
Goya y Lucientes, Francisco de - "*Estragos de la guerra*"	0.00013531
Anónimo - "*Proyecto de decoración arquitectónica para un palacio del rey Carlos II de Inglaterra*"	0.00013529

7. SCALABILITY

In regards of scalability, the proposed algorithms implementations should be able to handle bigger datasets.

The use of adjacency list as a way to represent edges, instead of the Transition Matrix, provides that only the relevant links are stored in memory and avoid unnecessary computational overhead. In essence, the complexity of the calculations is reduced from $O(N^2)$ when using the full Transition Matrix, to $O(E)$ where E is the number of edges.

Moreover, the functions take as input the Spark RDDs of edges and distribute computations across a cluster of machines. The use of RDDs ensures that the adjacency list, contributions and PageRank computations are distributed and performed in parallel. The

map-reduce functions of PySpark, such as map, `flatMap`, `groupByKey`, `reduceByKey`, are designed to work on distributed data and each machine in the cluster can process a part of the data independently. Specifically, by using `flatMap` and `reduceByKey`, we compute the values of contributions incrementally, without storing useless intermediate results.

8. CONCLUSION

In this project, I implemented and compared two versions of the PageRank algorithm, one using unweighted edges and the other using weighted edges, on the Prado Museum dataset. The goal was ranking the paintings based on common tags. The developed solutions should be able to handle larger dataframes thanks to the use of parallelized computation allowed by PySpark RDD, its map-reduce functions and a more efficient representation of nodes' links than the Transition Matrix.

Regarding the results of the two PageRank approaches, we can see that the same painting from the artist José Nin y Tudó is ranked at the top for both the algorithms, with a PageRank value slightly smaller in the weighted edges algorithm. However, the other top ranked paintings varied between the weighted and unweighted algorithm, with the exception of "*Retrato de James Bourdieu"* from Sir Joshua Reynolds, that is ranked second in the weighted edges approach and fourth in the unweighted edges algorithm. It is interesting to notice that 5 of the top paintings ranked by the weighted edges algorithm are from the artist Francisco de Goya y Lucientes, with very small differences in the PageRank values, showing that these works shared highly frequent occurring tags.

In summary, the unweighted PageRank put higher emphasis on the overall connectivity of the nodes, while the weighted PageRank gave priority also to the significance of the common tags.

9. BIBLIOGRAPHY

- Leskovec J, Rajaraman A, Ullman J. *Mining of Massive Datasets.* (2011)