Link Analysis

Tobias Walker, Student-ID: 990443

Master of Data Science and Economics

Algorithms for Massive Datasets, July 2023

1. Introduction

The aim of this paper is to utilize the PageRank algorithm to rank Amazon products. The PageRank algorithm works on networks by assigning a numerical weight to each node based on the quantity and quality of its incoming connections. This weight can be interpreted as the likelihood a random surfer would land on a node when he would only follow outgoing connections. In our application, the products serve as the nodes, and the links between products represent the edges. The links are constructed using a dataset comprising of customer reviews and are defined such that a link exists between two products if at least one customer has reviewed both products.

Two significant properties can be deduced from this link definition. Firstly, each link is bidirectional, eliminating the problems of dead-ends and spider traps, because if an incoming connection exists, also an outgoing exists. Products without any link are removed from the dataset. Secondly, irrespective of the number of customers who reviewed both products, only one link exists between them.

However, this link definition does not guarantee a fully connected network. In certain cases, there may be two products in the dataset with only one review each, and both reviews written by the same person. Consequently, these two products become connected to each other, and therefore stay in the dataset, but remain disconnected from the rest of the product network. The initially given PageRank values are captured on these “islands”, potentially inflating the rankings of the affected products. They may appear therefore relatively more important compared to products connected to the larger network. To address this issue, one potential solution is random teleportation, which involves redistributing a portion of the captured PageRank randomly. However, due to its potential to introduce distortions to all product rankings, and its limited ability to redistribute the captured PageRank, random teleportation is not employed in this specific use-case. Another possible, and probably more appropriate solution is an extensive network analysis to identify and flag products on isolated islands. Due to memory limitations and a focus of the paper on the application of the PageRank algorithm, a network analysis is not performed and therefore products on islands remain unidentified inside the dataset.

In the subsequent sections, this paper will delve into the details of the implementation of the PageRank algorithm. It will explore the methodology, discuss the challenges encountered, present the results, and lastly discuss the limitations of this ranking approach in the given use-case.

2. The implementation of the PageRank algorithm

The dataset of Amazon reviews is divided into different product categories. For this project, only the dataset labelled 'automotive' was used due to memory limitations. These memory limitations posed the most significant challenge in implementing the project. The selected dataset alone consists of over 2.1 million reviews of more than 550,000 products. In theory, if every product were connected to every other product, the number of links would exceed 250 billion. With around 15.8 million this number of links is much smaller in practice. Nonetheless storing the PageRank matrices completely in memory wasn’t feasible

To address the memory limitations, the chosen solution leveraged the capabilities of PySpark data frames. The dataset was imported directly into a PySpark environment, and all necessary transformations and computations were performed within the data frame framework. This approach proved to be memory-efficient, as it eliminated the need to collect the entire dataset at once. However, using data frames also brought some disadvantages with it. The primary drawback is that data frames are a higher-level storage form, which means that explicit computation methods specifically optimized for PageRank, such as MapReduce, could not be directly employed. Instead, the computation was optimized by PySpark in the backend, which may result in slower computation speeds.

Regarding the PageRank algorithm itself, as said it works based on the concept of a random surfer and it involves two key matrices. The first matrix describes the transition probabilities between each node in the network. Meaning that it represents the likelihood that the random surfer, starting from one node, will end up at another node, after taking only one step. To save memory, a sparse matrix is constructed, containing information in the format $\left[\left(ID_i, ID_j, transitionProbability\right), \ldots\right]$, with $transitionProbability \in (0,1]$ and where ID_i and ID_j denote all existing links. The second matrix consists of the initial PageRank values for each node. Initially, all products have the same PageRank value, represented as $PR_0 = \left[\left(ID_i, \frac{1}{n}\right), \left(ID_j, \frac{1}{n}\right), \ldots\right]$, where n represents the total number of unique products with existing links. Because the counting of unique products was implemented via an approximation, the PageRank values might not add up to exactly to 1, but it should be close to it.

The computation of PageRank occurs within a loop, every iteration representing another step the random surfer makes. This loop continues until either a maximum number of iterations is reached (in this paper, 50 iterations) or until the PageRank values converge to a stable state. The distance measure used in this paper is the sum of the squared differences between PageRank values

$$\Sigma_{j=1}^{n}\left(PR_{k,j} - PR_{k-1,j}\right)^2$$

here after the k-th iteration. If this value falls below $1.97 * 10^{-7}$, then the loop is terminated. Within each iteration of the loop, the PageRank values are updated based on the transition probabilities and the PageRank matrix. The formula for the update step can be expressed as:

$$PR_{k,i} = \Sigma_{j=1}^{n} m_{ij} PR_{k-1,j},$$

where $PR_{k,i}$ represents the updated PageRank value for node i in the k-th iteration, $PR_{k-1,j}$ represents the PageRank of node j in iteration $k-1$, and m_{ij} represents the transition matrix containing the transition probabilities. This formula enables the iterative computation of the PageRank values, gradually refining the rankings of the nodes based on the incoming link structure.

Overall, the implementation of the PageRank in this paper considers the constraints of the dataset, leverages PySpark data frames for memory efficiency, and follows the fundamental steps of the PageRank algorithm to determine the relative importance of nodes in the network.

3. Results

With the PageRank successfully constructed, now a look at the results of it can be taken. At first, I have a look at the properties of the update loop. The loop terminated after 24 iterations, triggered by the change in the PageRank values dropping below the threshold. Figure 1 visually demonstrates that the marginal rate of change converged to a value close to the threshold within a few iterations. This convergence graph indicates that significant fluctuations in the PageRank values no longer occurred, justifying the early termination of the loop.

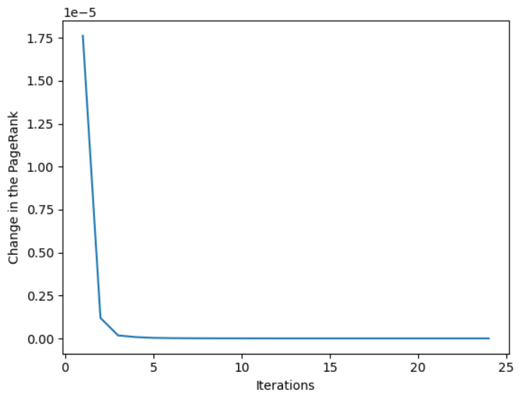

Figure 1: Change in the PageRank over the iterations

With the confidence in the appropriate termination of the loop, attention can now be directed towards the resulting PageRank values. Figure 2 shows the products with the five highest PageRanks. The product with the ID $B000CITK8S$ received the highest PageRank, approximately $1.2 * 10^{-3}$, while the fifth most important product, $B00068XCQU$, was assigned a PageRank value of around $0.6 * 10^{-3}$. It is evident that these top-ranked products accumulated a substantial amount of the PageRank. Taking the initially given PageRank value $1.8 * 10^{-6}$, the highest rank product was able to gather the PageRank of over 650 products onto himself. Because the PageRank is a zero-sum game, the increase in some product's PageRank comes at the expense of some others. Therefore, the product with the smallest PageRank, $B004XH6PQ4$ has a PageRank of $4.7 * 10^{-8}$, which is below the initially given one.

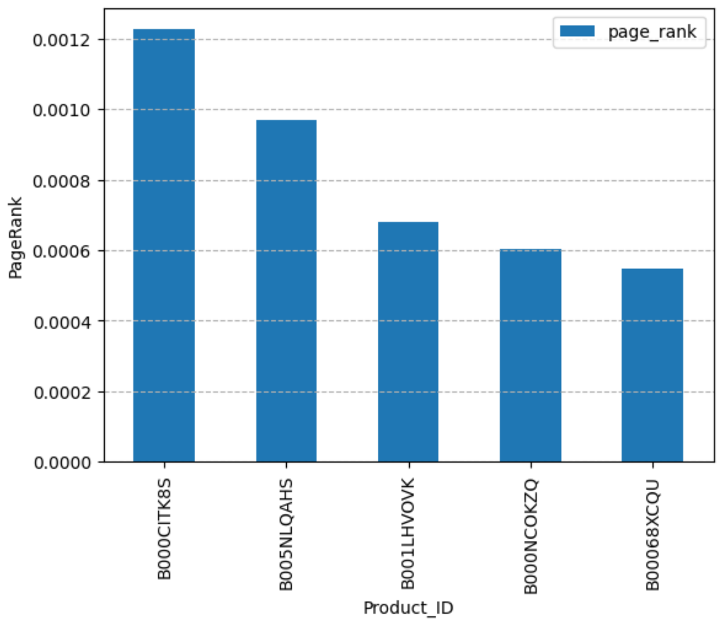

Figure 2: The Products with the 5 highest PageRanks

4. Discussion

The resulting PageRank estimate gives us now the opportunity to rank the items in the category 'automotive'. But not only in the general category, but also in every sub-category, be it a product sub-category (e.g., headlights, phone holder) or a sub-category created by filters (e.g., price, average rating). Furthermore, because the PageRank is constructed on a network, also a ranking in the form of people who bought this product also bought these ones can be built from the PageRank, by taking all products that are directly linked to the looked at product and order them by their PageRank.

Nonetheless, this approach has also some major disadvantages, that can be split into two categories. The first category of disadvantages is concerned with the way the PageRank was implemented in this case. For one, the PageRank method doesn't consider the rating of the review in its estimate. Therefore, a negatively rated product could be ranked high as long as it has many high-quality incoming links. A partial redistribution of the PageRank based on the average review rating could mitigate this problem. Secondly this method is prone to manipulation. If a firm wants to increase the rank of its product, it simply has to review the highest ranked products and its own product with one account. This problem can only be mitigated by a very effective fake review recognition.

The second category are general limitations of the PageRank approach. Here especially the individual customer independence must be named. The PageRank is built upon the aggregated network generated through the data of everyone. A customer might have reviewed a specific brand negatively multiple times, but still gets a product recommendation from this firm, if the PageRank of it is high. The PageRank isn't a sophisticated recommendation system, it's a simple but effective network node ranking method. For this use-case the PageRank might only be acceptable in the usage for new users. For customers that already expressed their personal preferences in their search and purchasing history, a ranking based on PageRank might not be appropriate.