# PageRank

Cloud Computing final project - Prof. Nicola Tonellotto

*Leonardo Turchetti*
*Lorenzo Tonelli*
*Ludovica Cocchella*
*Rambod Rahmani*

Msc. in Artificial Intelligence and Data Engineering

May 25, 2021

## Contents

## 1 Introduction

The importance of a web page is an inherently subjective matter, which depends on the readers interests, knowledge and attitudes. PageRank can be defined as a method for rating web pages objectively and mechanically, effectively measuring the human interest and attention devoted to them. In order to measure the relative importance of web pages, PageRank was proposed as a method for computing a ranking for every web page based on the graph of the web.

The project focused on designing a MapReduce algorithm (using pseudocode) to implement the PageRank (using both Hadoop and Spark). Initially, a pseudocode implementation and the design assumptions are presented. The successive sections focus on the implementation details using both Hadoop and Spark. Finally, the validation results obtained using both a realistic and a synthetic dataset are provided.

The entire codebase is available at https://github.com/lorytony/CloudComputing.

## 2 PageRank

PageRank[1] is a measure of web page quality based on the structure of the hyperlink graph. Although it is only one of thousands of features that is taken into account in Google's search algorithm, it is perhaps one of the best known and most studied.

The reason why PageRank is so interesting is that there are many cases where simple citation counting does not correspond to our common sense notion of importance. For example, if a web page has a link off the Yahoo home page, it may be just one link but it is a very important one. This page should be ranked higher than many pages with more links but from obscure places. PageRank is an attempt to see how good an approximation to "importance" can be obtained just from the link structure.

Every page has some number of forward links (out-edges) and backlinks (in-edges). We can never know whether we have found all the backlinks of a particular page but if we have downloaded it, we know all of its forward links at that time.
The previous example models the so called "Propagation of Ranking Through Links": a page has high rank if the sum of the ranks of its backlinks is high. This covers both the case when a page has many backlinks and when a page has a few highly ranked backlinks.

Formally, given

- a page $p_i$ among the total $N$ nodes (pages) in the graph;
- the set of pages $L(p_i)$ that link to $p_i$;
- and the out-degree $C(p_j)$ of node $p_j$;
- the random jump factor $\alpha$;

the PageRank $PR$ of a page $p_i$ is defined as follows:

$$PR(p_i) = \alpha \frac{1}{N} + (1-\alpha) \sum_{p_j \in L(p_i)} \frac{PR(p_j)}{C(p_j)}$$

The definition of PageRank above has another intuitive basis in random walks on graphs. The simplified version corresponds to the standing probability distribution of a random walk on the graph of the Web. Intuitively, this can be thought of as modeling the behaviour of a "random surfer". The "random surfer" simply keeps clicking on successive links at random. However, if a real Web surfer ever gets into a small loop of web pages, it is unlikely that the surfer will continue in the loop forever. Instead, the surfer will jump to some other page. The additional factor $\alpha$ can be viewed as a way of modeling this behaviour: the surfer periodically "gets bored" and jumps to a random page. This residual probability, $\alpha$, is usually set to 0.15, estimated from the frequency that an average

[1]The PageRank Citation Ranking: Bringing Order to the Web - January 29, 1998 - http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

surfer uses his or her browser's bookmark feature. Alternatively, $1 - \alpha$ is referred to as the "damping" factor.

## 2.1 Computation

PageRank can be computed either iteratively or algebraically. Using the iterative method, at $t = 0$, an initial probability distribution is assumed

$$PR(p_i, 0) = \frac{1}{N}$$

where $N$ is the total number of pages, and $p_i$;0 is page $i$ at time 0. At each time step, the computation, as detailed above, yields

$$PR(p_i, t+1) = \alpha \frac{1}{N} + (1 - \alpha) \sum_{p_j \in L(p_i)} \frac{PR(p_j, t)}{C(p_j)}$$

## 2.2 Implementation Assumptions

In the presented implementation, some assumptions were made.

Firstly, Nowadays, a colossal 4.2 billion pages exist on the Web, spread across 8.2 million web servers. No crawling operations were taken into account. In what follows, the assumption was made that the the inputs to the program are pages from the Simple English Wikipedia. We will be using a pre-processed version of the Simple Wikipedia corpus in which the pages are stored in an XML format. Each page of Wikipedia is represented in XML as follows:

```
<title>web page name</title>
...
<revision optionalVal="xxx">
    ...
    <text optionalVal="yyy">page content</text>
    ...
</revision>
```

The pages have been "flattened" to be represented on a single line. The body text of the page also has all newlines converted to spaces to ensure it stays on one line in this representation. This makes it easy to use the default `InputFormat`, which performs one `map()` call per line of each file it reads. Links to other Wikipedia articles are of the form `[[page name]]`. Starting from this input file, first the hyperlink graph is constructed and then the PageRank is computed for each node.

## 2.3 Pseudocode Implementation

# 3 PageRank Implementation using Hadoop

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam aliquam justo eget nunc condimentum, eget iaculis justo venenatis. Aenean id tellus at velit vestibulum tempor nec eu mi. Maecenas lobortis eu mi quis condimentum. Maecenas mi ipsum, semper at felis in, consequat lobortis lorem. Ut imperdiet, ante mattis interdum tristique, orci leo feugiat lorem, facilisis volutpat diam tortor ac quam. Integer faucibus, odio sed consequat sagittis, nunc mi aliquet turpis, ut laoreet velit dolor non nisi. Vivamus vitae libero a est feugiat iaculis ac id mauris.

### 3.1 Custom `WritableComparable` object

## 4 PageRank Implementation using Spark

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam aliquam justo eget nunc condimentum, eget iaculis justo venenatis. Aenean id tellus at velit vestibulum tempor nec eu mi. Maecenas lobortis eu mi quis condimentum. Maecenas mi ipsum, semper at felis in, consequat lobortis lorem. Ut imperdiet, ante mattis interdum tristique, orci leo feugiat lorem, facilisis volutpat diam tortor ac quam. Integer faucibus, odio sed consequat sagittis, nunc mi aliquet turpis, ut laoreet velit dolor non nisi. Vivamus vitae libero a est feugiat iaculis ac id mauris.

### 4.1 Python

### 4.2 Java

## 5 Validation

```
hadoop@namenode:¬/PageRank/hadoop/pagerank$ hadoop fs —cat output1/part*
2021—05—21 10:12:43,606 INFO sasl.SaslDataTransferClient: SASL encryption ...
    trust check: localHostTrusted = false, remoteHostTrusted = false
N       5
hadoop@namenode:¬/PageRank/hadoop/pagerank$ hadoop fs —cat output2/part*
2021—05—21 10:12:57,424 INFO sasl.SaslDataTransferClient: SASL encryption ...
    trust check: localHostTrusted = false, remoteHostTrusted = false
n1      0.2     n4]]n2
n2      0.2     n5]]n3
n3      0.2     n4
n4      0.2     n5
n5      0.2     n3]]n2]]n1
hadoop@namenode:¬/PageRank/hadoop/pagerank$ hadoop fs —cat output3/part*
2021—05—21 10:13:00,755 INFO sasl.SaslDataTransferClient: SASL encryption ...
    trust check: localHostTrusted = false, remoteHostTrusted = false
n1      0.06666666666666667     n4]]n2
n2      0.16666666666666669     n5]]n3
n3      0.16666666666666669     n4
n4      0.30000000000000004     n5
n5      0.30000000000000004     n3]]n2]]n1
hadoop@namenode:¬/PageRank/hadoop/pagerank$ hadoop fs —cat output4/part*
2021—05—21 10:13:09,420 INFO sasl.SaslDataTransferClient: SASL encryption ...
    trust check: localHostTrusted = false, remoteHostTrusted = false
n5      0.30000000000000004
n4      0.30000000000000004
n3      0.16666666666666669
n2      0.16666666666666669
n1      0.06666666666666667
hadoop@namenode:¬/PageRank/hadoop/pagerank$
```

```
hadoop@namenode:¬/PageRank/hadoop/pagerank$ hadoop fs —cat output1/part*
2021—05—21 14:43:14,966 INFO sasl.SaslDataTransferClient: SASL encryption ...
    trust check: localHostTrusted = false, remoteHostTrusted = false
N       5
hadoop@namenode:¬/PageRank/hadoop/pagerank$ hadoop fs —cat output2/part*
2021—05—21 14:43:18,513 INFO sasl.SaslDataTransferClient: SASL encryption ...
    trust check: localHostTrusted = false, remoteHostTrusted = false
n1      0.2     n4]]n2
n2      0.2     n5]]n3
```

```
n3      0.2     n4
n4      0.2     n5
n5      0.2     n3]]n2]]n1
hadoop@namenode:~/PageRank/hadoop/pagerank$ hadoop fs -cat output3/part*
2021-05-21 14:43:21,890 INFO sasl.SaslDataTransferClient: SASL encryption ...
   trust check: localHostTrusted = false, remoteHostTrusted = false
n1      0.06666666666666667     n4]]n2
n2      0.16666666666666669     n5]]n3
n3      0.16666666666666669     n4
n4      0.30000000000000004     n5
n5      0.30000000000000004     n3]]n2]]n1
hadoop@namenode:~/PageRank/hadoop/pagerank$ hadoop fs -cat output4/part*
2021-05-21 14:43:25,608 INFO sasl.SaslDataTransferClient: SASL encryption ...
   trust check: localHostTrusted = false, remoteHostTrusted = false
n1      0.10000000000000002     n4]]n2
n2      0.13333333333333336     n5]]n3
n3      0.18333333333333335     n4
n4      0.2     n5
n5      0.3833333333333334      n3]]n2]]n1
hadoop@namenode:~/PageRank/hadoop/pagerank$ hadoop fs -cat output5/part*
2021-05-21 14:43:29,478 INFO sasl.SaslDataTransferClient: SASL encryption ...
   trust check: localHostTrusted = false, remoteHostTrusted = false
n5      0.3833333333333334
n4      0.2
n3      0.18333333333333335
n2      0.13333333333333336
n1      0.10000000000000002
```