Università
Ca'Foscari
Venezia

INFORMATION RETRIEVAL AND WEB SEARCH
CM 0473

PAGE RANK/ HITS COMPUTATION

Students:

Luca Bizzotto 875814

Giosuè Zannini 873810

Academic year 2021/2022

Contents

Chapter 1

Introduction

1.1 Problem

In this project we have to perform two different algorithms to evaluate the prestige of each web page:

1. Page Rank.
2. HITS.

For each of them we had to write a sequential code in C++ and measure the execution times for some Network Dataset. Moreover for each graph we have to rank nodes by: HITS, Page Rank and In Degree and computing for them the top k nodes ($k = 10, 20, 30, \dots$) and show the results using Jaccard coefficient.
We have chose four different dataset:

- Berkeley-Stanford web graph: it is composed by 685230 nodes and 7600595 edges.
- Google web graph: it is composed by 916428 nodes and 5105039 edges.
- Notre Dame web graph: it is composed by 325729 nodes and 1497134 edges.
- Stanford web graph: it is composed by 281903 nodes and 2312497 edges.

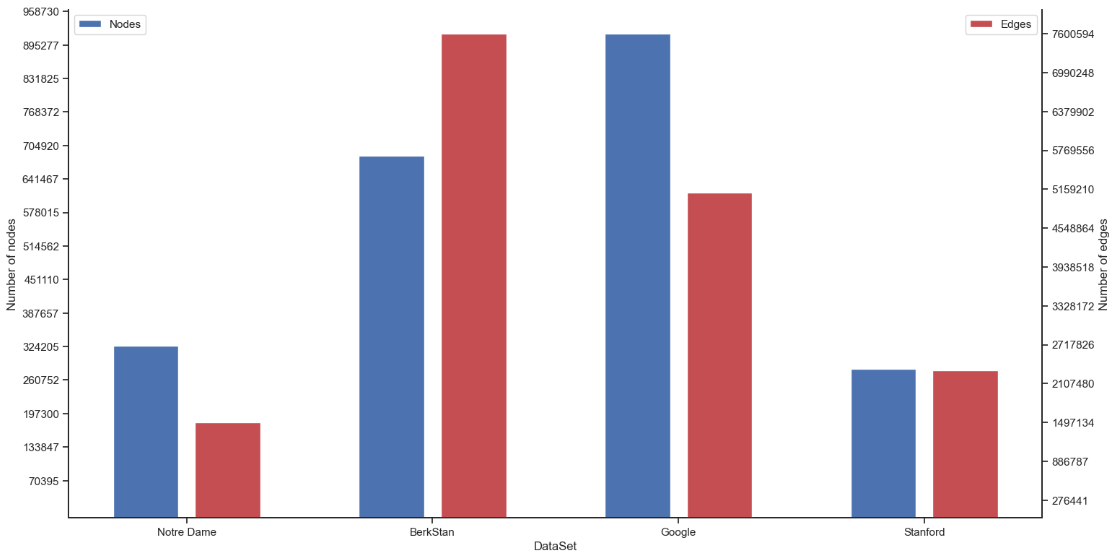

FIGURE 1.1: Summary of datasets composition

1.2 Approach

In this section we just specify a couple of items used in the final solution, more implementation details are explained in the code. In our implementations we had to create a library that contains:

- Hits.h : contains the class to perform the HITS algorithm.
- PageRank.h : contains the class to perform the Page Rank algorithm.
- InDegree.h : contains the class to perform the InDegree algorithm.
- main.cpp : contains the main function to run code.

Each file.h rappresents a class, the reason why we created them is to simplify certain types of operations and to use programming object-oriented approach. Initially we did a brief analysis on the datasets that we decided to use during our analysis. We realized that some datasets did not have the ID of the minimum node equal to 0. For this reason we decided to carry out a normalization of the graphs. We performed it by inserting a line in the dataset which indicates the ID of the smallest node and the ID of the largest node. This precomputation

phase was performed with Python for faster implementation.
Once we have performed this initial phase we started to read each file, we realized that if we represent the dense adjacency matrix e.g. Notre Dame graph using a double per entry, we need about ~ 800 GB. So for this reason we decided to use the CSR or CRS compressed technique to rappresent the matrix.
Even using this compressed technique the amount of data to be stored exceeds the amount of RAM available, so to overcome this problem we used mmap function that allows us to store data in the disk. To improve the performance we thought which variables keep in main memory, to do this we did an analysis focused on memory based on the greatest number of edges and nodes present in the datasets treated by us. This analysis brought us to the following results:

- Page Rank: ~ 25 MB.
- HITS: ~ 30 MB.

So they can easily fit into RAM.

Chapter 2

Page Rank

2.1 Theory in a nutshell

Page Rank is a link analysis algorithm, named after Brin & Page, and used by the Google internet search engine, which assigns a numerical weighting to each element of hyperlinked set of documents, such as the World Wide Web, with the purpose of measuring its relative importance within the set.
Page Rank relies on the democratic nature of the web. Uses the vast link structure as an indicator of an individual page's value or quality. The score of the page i is defined by:

$$P(i) = \sum_{(j,i)\in E} \frac{P(j)}{O_j} \tag{2.1}$$

To achieve the above formula is used the Markov chain, this framework models web surfing as a stochastic process.
We have three assumptions:

- The back button is not used.
- The surfer does not type in an URL.
- The surfer clicks the hyperlinks in the page uniformly at random.

By the Ergodic Theorem of Markov chain "A stochastic matrix $\mathbf{A}$ has a unique stationary probability distribution if A is irreducible and aperiodic".

So:

$$\lim_{k \to \infty} P_k = \pi \qquad \pi = \mathbf{A}^{\mathrm{T}} \pi \tag{2.2}$$

Where π is the principal eigenvector of $\mathbf{A}^{\mathrm{T}}$ with eigenvalue of 1.
To satisfy the stochastic property we decided to add a complete set of outgoing links from each page i to all the other pages on the web.
To satisfy both irreducible and aperiodic properties we used the teleportation technique which adds a link from each page to every page and gives each link a small transition probability controlled by a hyper parameter d, $0 < d < 1$.
So after this augmentation, at a page, the random surfer has two options:

- With probability d he/she randomly chosen an out link.
- With probability $1 - d$ he/she jumps to a random page.

Now we can use a well known mathematical technique called power iteration method to find P. The formula that we used is the following:

$$P_{k+1} = (1-d)\frac{e}{n} + d\mathbf{A}^{\mathrm{T}} P_k \tag{2.3}$$

2.2 Pseudo code

Below the pseudo code of the algorithm just described:

```
1: PageRank_Iterate(G)
2: $P_o \leftarrow \frac{e}{n}$
3: k $\leftarrow$ 0
4: do
5:     $P_{k+1} \leftarrow (1-d)\frac{e}{n} + d\mathbf{A}^{\mathrm{T}} P_k$
6:     $k \leftarrow k + 1$
7: while $||P_{k+1} - P_k||_1$
8: return $P_{k+1}$
```

Chapter 3

Hyperlink-Induced Topic Search (HITS)

3.1 Theory in a nutshell

It is a link analysis algorithm that rates web pages. Unlike PageRank which is a static ranking algorithm, HITS is query dependent.
When an user issues a search query HITS first expand the list of relevant pages returned by a search engine, and then produce two rankings of the expanded set of pages, authority ranking and hub ranking. Authority and hub values are defined in terms of one another in a mutual recursion. An authority value is computed as the sum of the scaled hub values that point to that page.
A hub value is the sum of the scaled authority values of the pages it points to. Roughly, we can say that an authority is a page with many in-links, the idea it is that the page may have good or authoritative content on some topic thus many people trust it and link to it.
An hub otherwise is a page with many out-links, the pages serves as an organizer of the information on a particular topic and points to many good authority pages on the topic. The key idea is so that a good hub points to many good authorities and a good authority is pointed to by many good hubs.
HITS as we said before work with the expanded sets of pages returned by the search engine and assign to each page an authority score and a hub score.

Donating the adjacency matrix of the graph $\mathbf{L}$ we represent authority score and hub score as follows:

$$\begin{aligned} a(i) &= \sum_{(j,i)\in E} h(j) \\ h(i) &= \sum_{(i,j)\in E} a(j) \end{aligned} \tag{3.1}$$

If we use a as the column vector with all the authority scores $a = (a(1), ..., a(n))^{\mathrm{T}}$ and h denote the column vector with all the hub scores $h = (h(1), ..., h(n))^{\mathrm{T}}$ then we can rewrite the formula 3.1 in

$$\begin{aligned} a &= \mathbf{L}^{\mathrm{T}} h \\ h &= \mathbf{L} a \end{aligned} \tag{3.2}$$

Then the computation of authority scores and hub scores is based in the power iteration method used even for the computation of the Page Rank.

3.2 Pseudo code

Below the pseudo code of the algorithm just described:

```
1:  Hits_Iterate(G)
2:  $a_o \leftarrow (1, ..., 1)$
3:  $h_o \leftarrow (1, ..., 1)$
4:  k ← 0
5:  do
6:      $a_k \leftarrow \mathbf{L}^{\mathrm{T}} h_{k-1}$
7:      $h_k \leftarrow \mathbf{L} a_{k-1}$
8:      $a_k \leftarrow$ normalized $a_k$
9:      $h_k \leftarrow$ normalized $h_k$
10: while $a_k$ and $h_k$ not change significantly
11: return $a_k$ and $h_k$
```

Chapter 4

Results and Comparison

In this section we will make a comparison between the various results obtained in the datasets used with respect to time, convergence and finally Jaccard score. About Page Rank algorithm we chose as teleportation probability with value equal to 0.8. As performance measures we used the computational time and steps to converge for both algorithms.

4.1 Berkeley-Stanford web graph

The Berkeley-Stanford web graph is characterize by:

- Min node ID: 1.
- Max node ID: 685230.
- Number nodes: 685230.
- Number edges: 7600595.

As we can see from the chart below we can say that the HITS algorithm uses less time and converge steps than Page Rank even if the difference is minimal.

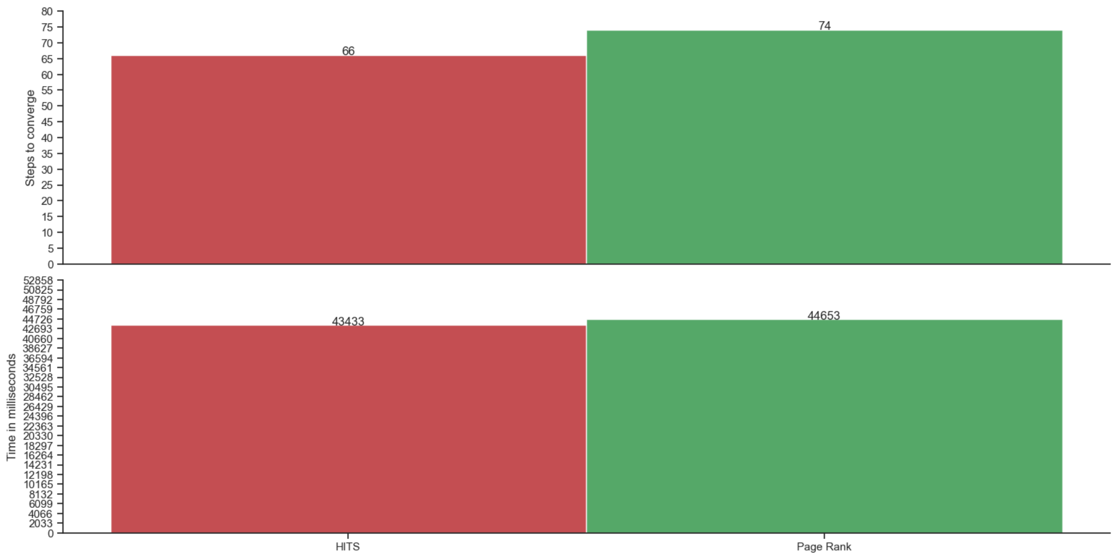

FIGURE 4.1: Difference between the two algorithms in time and number of steps

From the plot below we can say that the similarity among various methods used, is marked as far as the couple HITS autority and InDegree especially for k=32. For the other couples it is almost zero.

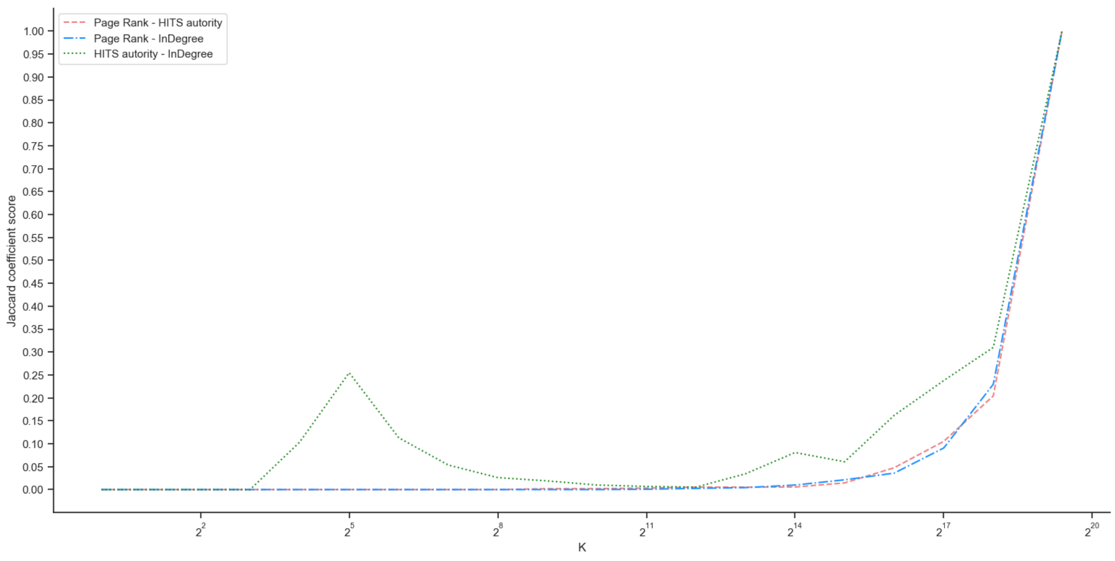

FIGURE 4.2: Plotting Jaccard coefficient score for different values of k

4.2 Google web graph

Google web graph is characterize by:

- Min node ID: 0.
- Max node ID: 916427.
- Number nodes: 916428.
- Number edges: 5105039.

As we can see from the chart below we can say that Page Rank algorithm takes much less time and converge steps than HITS.

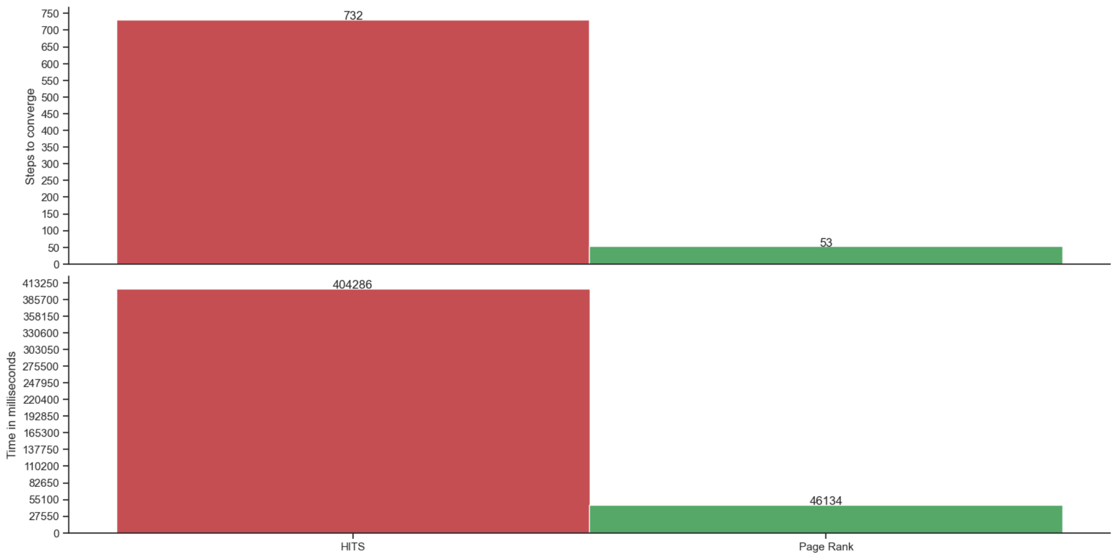

FIGURE 4.3: Difference between the two algorithms in time and number of steps

From the plot below we can say that the similarity among various methods used, is marked as far as the couple HITS autority and InDegree in all the curve. For the other couples it is almost zero.

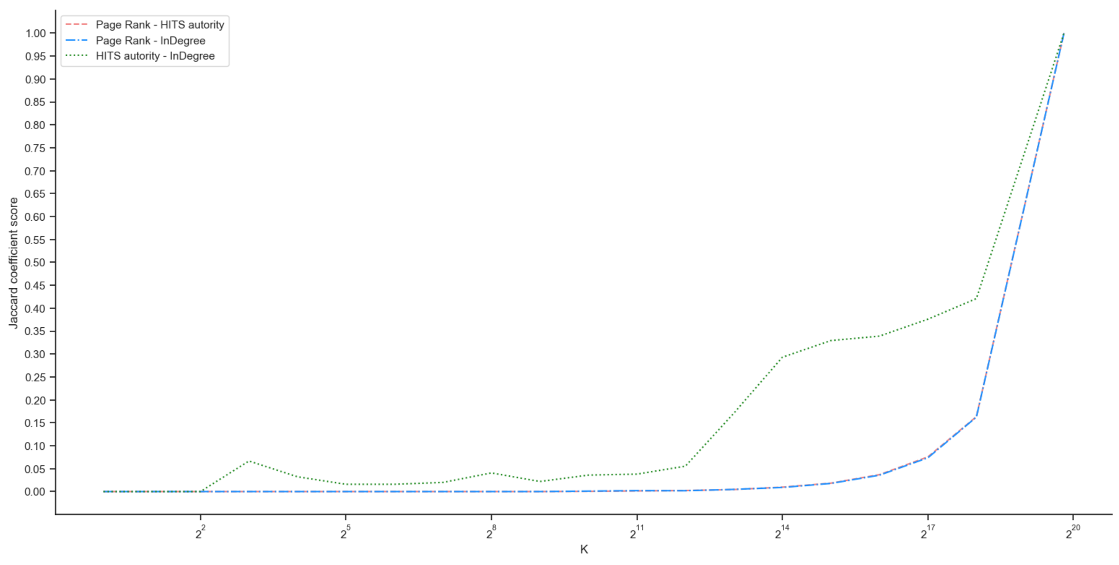

FIGURE 4.4: Plotting Jaccard coefficient score for different values of k

4.3 Notre Dame web graph

Notre Dame web graph is characterize by:

- Min node ID: 0.
- Max node ID: 325728.
- Number nodes: 325729.
- Number edges: 1497134.

As we can see from the chart below we can say that Page Rank algorithm uses less time and converge steps than HITS especially the gap between steps to converge is marked.

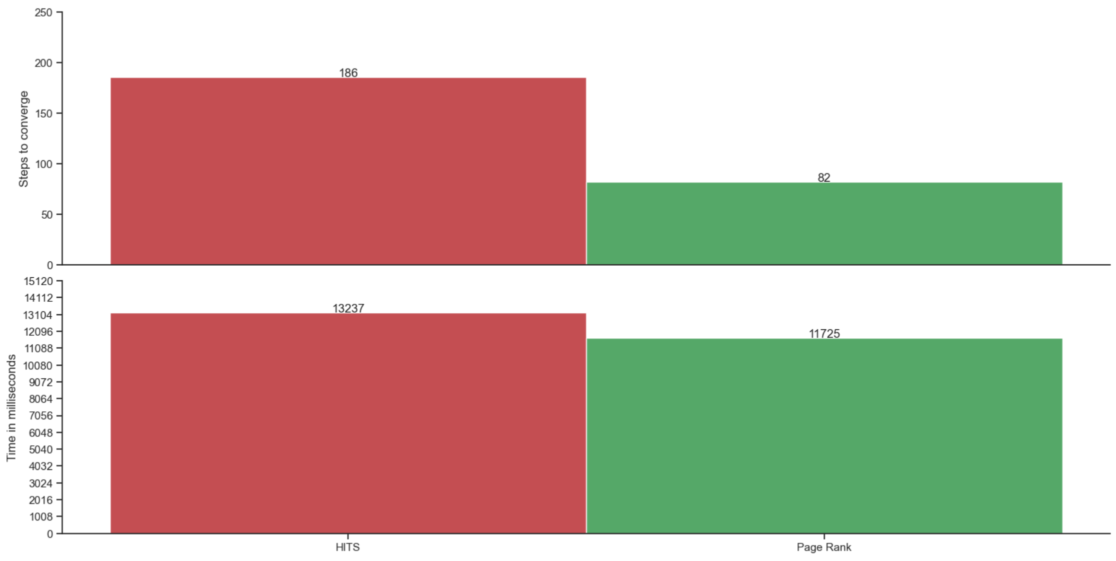

FIGURE 4.5: Difference between the two algorithms in time and number of steps

From the plot below we can say that the similarity among various methods used, is marked for all curves except the couple Page Rank and HITS autority.

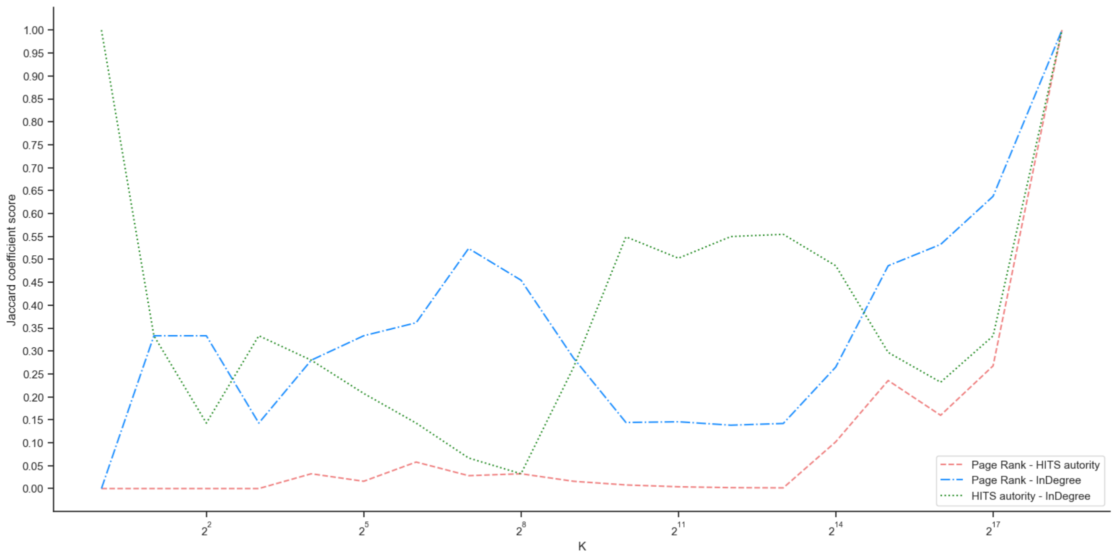

FIGURE 4.6: Plotting Jaccard coefficient score for different values of k

4.4 Stanford web graph

- Min node ID: 1.
- Max node ID: 281903.
- Number nodes: 281903.
- Number edges: 2312497.

As we can see from the chart below we can say that Page Rank algorithm uses more time and converge steps than HITS especially the gap between time is slightly more marked.

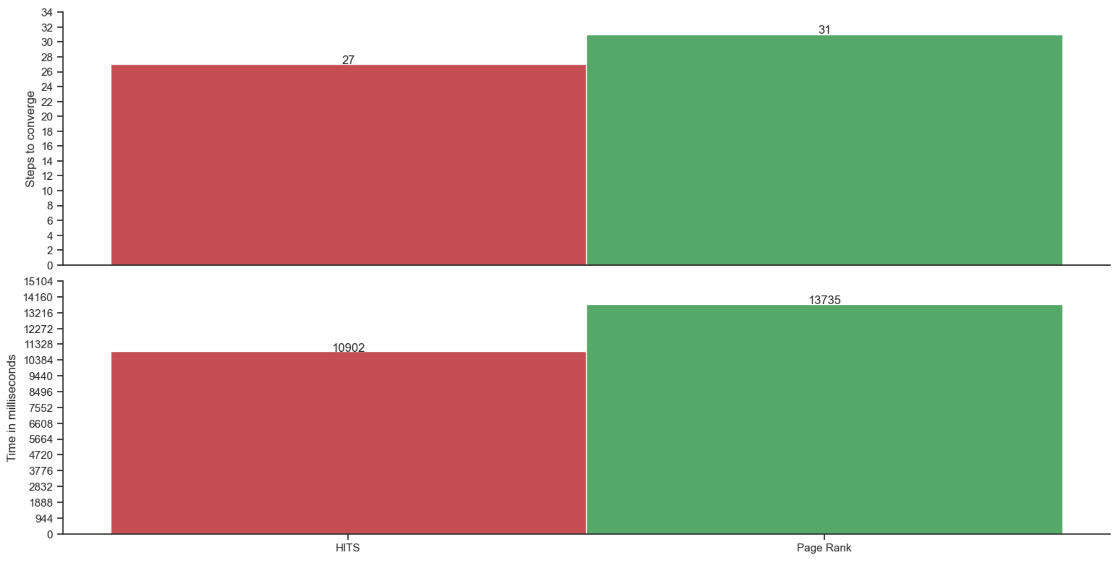

FIGURE 4.7: Difference between the two algorithms in time and number of steps

From the plot below we can say that the similarity among various methods used, is marked as far as the couple HITS autority and InDegree in fact for smaller values of k the similarity is roughly 1. For the other couples it is almost zero.

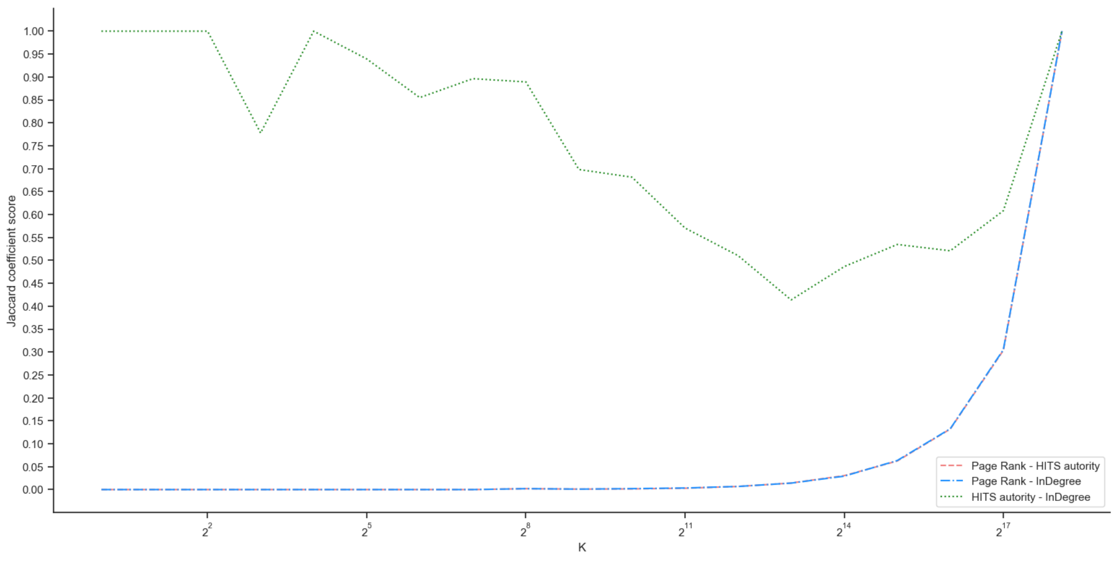

FIGURE 4.8: Plotting Jaccard coefficient score for different values of k

Chapter 5

Conclusions

In conclusion from the comparison among all the previous plots 4.2, 4.4, 4.6 and 4.8 this analysis showed us that from a general point of view the Jaccard coefficient it's low for the couples: Page Rank - InDegree, Page Rank - HITS autority, otherwise for the remaining couple HITS autority - InDegree it's quite high because both are based on the number of in degree of each node. Moreover for large number of top k nodes, the similarity among all couples converge to 1. The motivation is due to the fact that when we increase k we are increasing the number of documents that are retrieved, so the cardinality of intersection improves until reach the cardinality of the union. As last observation speaking about performance the plot below shows us that in mean the Page Rank algorithm uses less time and steps to converge than HITS.

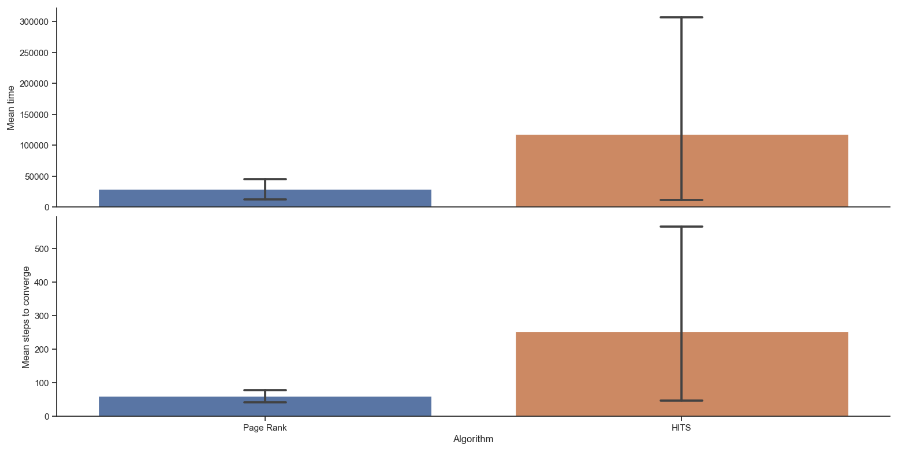

FIGURE 5.1: Comparison between the metrics for each algorithm using mean