PageRank for Small World Datasets

Social Network Analysis for Computer Scientists — Course paper

Diego Barreiro
s3254933@vuw.leidenuniv.nl
Leiden University
Leiden, Netherlands

Kenneth Fargose
s3281353@vuw.leidenuniv.nl
Leiden University
Leiden, Netherlands

ABSTRACT

In this paper, we attempt to assess how various implementations affect the PageRank algorithm results.

We discovered an opportunity to apply PageRank algorithms to professors at Leiden University's Department of Computer Science. Based on the computation times, we compare the obtained results to the expected results and attempt to devise a faster implementation or approximated approach.

For classifying nodes according to importance, we used a variety of random walk-based methods, and we attempted to evaluate the performance of these methods on small world networks such as those in the LIACS data-set.

By categorizing researchers according to their significance, we can assist future PhD candidates or researchers in finding a suitable collaborator with whom to collaborate on their projects.

KEYWORDS

PageRank, random walks, link analysis, social network analysis, network science, power iteration, monte carlo method, graph theory

ACM Reference Format:
Diego Barreiro and Kenneth Fargose. 2022. PageRank for Small World Datasets: Social Network Analysis for Computer Scientists — Course paper. In *Proceedings of Social Network Analysis for Computer Scientists 2021 (SNACS '21).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Prior to PageRank, most search engines solely relied on content and metadata to determine whether or not a webpage was relevant to a particular search. Such an approach was readily controlled, and the ultimate effect was bad search results with a lot of keywords crammed into the text of the top rated pages. Developed in 1998 by Google's co-founders, Larry Page and Sergey Brin, during their university years at Stanford, PageRank is a system for ranking web pages The first PageRank paper [2] covers the original algorithm used by Google to determine the importance of a web page and rank it. This is the premise upon which Google was founded. PageRank, according to Google, works by counting the number and quality of links to a page to determine a rough estimate of the website's importance. The underlying assumption is that more important websites are more likely to be linked to by other websites. Google has surpassed Yahoo as the most used search engine on the planet in the last few years. Apart from its excellent performance and ease of use, the greater quality of its search results over other search engines was a deciding factor. PageRank, plays a large role in the quality of these search results.

The higher a link's PageRank, the more authoritative and significant it is on a network.

PageRank, in a nutshell, tells us which webpage is the most important and critical in a given network, this can be very interesting if we only need to access a few valuable nodes in a network and obtain the connections to other less important nodes and how they impact the network.

The application of PageRank in most networks exposes nodes than with other algorithms would go unnoticed, because most implementations take solely into account centrality metrics for the measure of importance. This is not the case with PageRank, where lower degree nodes can appear with higher score than high degree nodes, this is due to the fact that PageRank revolves around link importance, it values the quality of links over the quantity.

The overall idea of the paper is to evaluate the performance of different PageRank implementations. We start of by explaining some essential concepts necessary to comprehend the steps taken in the experiments. Next we explain with very detailed information the data extraction process followed to create the LIACS dataset used for the performance evaluation task, so it can be easily reproduced. As a final step we comment on the results and we propose future work ideas that can be based on this paper.

SNACS '21, Master CS, Fall 2021, Leiden, the Netherlands

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 PROBLEM STATEMENT

The primary goal of this experiment is to implement and test various PageRank approaches for efficiently and effectively calculating the PageRank of links in a small world dataset, particularly the dataset we created. The objective of this experiment is to use various PageRank algorithm methods to rank the various professors and researchers who have collaborated on a research project and thus find the most influential person for a subject.

We do not know whether the current PageRank algorithms are effective for small datasets at this point in time. In practice, the current PageRank algorithm is extremely fast and efficient when dealing with large and massive amounts of data; however, we were interested in seeing if this was still true when dealing with much smaller datasets.

We aim to find a good approximation of the original PageRank algorithm, to reduce computing times and resources.

When dealing with large data-sets with billions of nodes, all PageRank algorithms perform similiar, but when dealing with small data-sets there is room for improvement.

With a small world data set, which contains information about researchers in the LIACS, we attempt to apply the PageRank 'classification' algorithm to it. Based on the approaches described above, we attempt to create a PageRank to rank the researchers.

In this study, we want to see how different PageRank implementations behave in a small world data set, such as our LIACS research data set, and compare the efficacy of each implementation.

3 BACKGROUND

Prior to the introduction of page rank, search engines relied on RankDex, which used link analysis. The popularity of a website was determined by how many other websites linked to it. PageRank refers to this as well as the idea that information on the web can be ordered in a hierarchy based on link popularity.

The PageRank Algorithm uses the probability distribution as an output to address the possibility that a person will arrive at a specific page after clicking on random pages. .

The probability is always between 0 and 1. A probability of 0.5 indicates that an event has a 50% chance of occurring. As a result, a page with a PageRank of 0.5 indicates that there is a 50% chance that this person will arrive at a desired page even after clicking on random pages.

3.1 Google matrix

A stochastic matrix used by Google's PageRank algorithm is referred to as A Google matrix.

The matrix represents a graph with edges representing links between pages. This stochastic matrix is then used to generate the PageRank of each using the Power Method.

In order to use this power method the matrix should be irreducible and aperiodic meaning, that associated directed graph of this matrix is strongly connected. The PageRank of each page can then be generated iteratively from the Google matrix using the power method. However, in order for the power method to converge, the matrix must be stochastic, irreducible and aperiodic.

Google matrix G can be expressed as:

$$G_{ij} = d * S_{ij} + (1 - d)\frac{1}{N} \qquad (1)$$

where S is a Sparse Matrix and the damping factor is d

The click-through probability dampening factor d is introduced to avoid sinks (i.e. pages with no outbound links) from consuming the PageRanks of pages linked to the sinks. If d=1, the person clicking will continue to click indefinitely, always ending up in a sink. In this scenario, the first phrase is discarded.

If the click-through probability is d=0, all clicks are random restarts that are evenly dispersed As a result, a damping factor of 0<d<1 is a weighted average of the two extremes.

4 METHODS

4.1 Random Walk Method

Algorithm 1 Random Walk

1: We set a value of zero to all the nodes in the data set.
2: Next we select a node out of all, randomly.
3: We find and keep the neighbors of the selected node
4: Then, we get another node from this list and we increment its value.
5: **if** Outdegree(SelectedNode)==0; **then**
6: select another node.
7: **else if** Outdegree(SelectedNode<0 **then**
8: increment its value
9: **end if**
10: Repeat the previous step till we reach convergence

The Random Walk Method is an algorithm for defining random paths in a graph. A random walk is similar to how a drunk person will roam around town. As the name implies, it begins with a node and chooses to traverse a neighboring node at random or based on the probability distribution provided, then repeats the process for this node and continues to list the resulting path.

4.2 Power method

The Power Method is simply the Power iteration for determining the eigenvector corresponding to the transition matrix's largest eigenvalue. Consider the following network

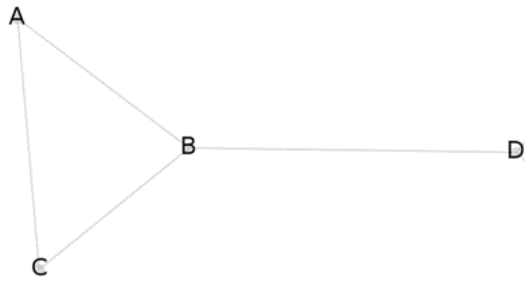

Figure 1: graph with 4 nodes ABCD

M – n x n Transition Matrix Fixed for a given network P_i-PageRanks at iteration

$P_{i+1} = M.[P_i]$

The transition matrix of the above network is

$$M = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Transpose of this transition matrix

$$M^{-1} = \begin{bmatrix} 0 & 1/3 & 1/2 & 0 \\ 1/2 & 0 & 1/2 & 1 \\ 1/2 & 1/3 & 0 & 0 \\ 0 & 1/3 & 0 & 0 \end{bmatrix}$$

The PageRank at this iteration is given by

$$P_i = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

After the matrix multiplication of P_i and M^{-1}, the result is the PageRank of the network at the iteration

$$P_{i+1} = \begin{bmatrix} 5/24 \\ 1/2 \\ 5/24 \\ 1/2 \end{bmatrix}$$

4.3 Our approach

We try and implement a PageRank 'classification' algorithm to a small world data set, a data set with information about researchers in the LIACS. We try to build a PageRank to rank the researchers based on the above mentioned approaches.

Our goal is to examine how different PageRank implementations behave in small world data set like our LIACS research data set, and compare the efficiency of each approach.

4.4 Dataset

As mentioned above the experiments carried out will be done on the LIACS researchers data-set that we have built. The information on this data set has been obtained scraping and processing the data as follows.

Next the steps followed to retrieve the information contained in this data-set will be explained:

(1) First we have to set up the packages we are going to use, we used `selenium` for the scraping process as it offers very simple commands to access different parts of a website. Alongside `selenium` we also need to download the chromedriver for our current chrome version (each browser has its own driver). Other packages used are `numpy`, `pandas` for the manipulation of the data and `matplotlib`, `seaborn` for the visualizations.
(2) Once we have the environment set up , we can initialize the chromedriver to navigate to the desired web-page, in this case we go into the Leiden University's official page, inside we go through the faculty of science and into the computer science section, in this last section we find the staff sub-page which will be where we obtain all our data. We do a manual inspection of the data first to know which `HTML` tags have the information we want. We need for every user its name and all the papers that they have published to their name. One small inconvenience encountered was finding that the publications for every person had a tricky access via `HTML` tags, the solution we came up with was to loop first through all users and extracting the link of their publications sub-folder (if any) and add such link to a list for later use.
(3) Next we loop through the list containing the link to the publications of a person, we scrape the name and all the paper's titles like mentioned above. Since there is a lot from the title we don't need we split it until we find the date, this will return all the contributors to the paper, we start a creating a first adjacency list making connections of the person whose publications sub-folder we were in with every other contributor, disregarding the connection made with himself (as his name also appears in the list of contributors), we obtain the date of the paper to add it as a timestamp to the adjacency list. The name of the person does not always appear in the same format, for this reason we had to stick to one, in this case the chosen format was the full surname followed by the initial of the name.
(4) After we have repeated this process for every person who has publications we start 'cleaning' the adjacency list, removing duplicate links and taking into account the number of times a links repeats itself to add it as weight attribute for the link, in duplicate links with different dates, we preserve the latest date as we estimate a later published paper adds more meaning to future searches.

(5) As a final step we change the data structure of the adjacency list to a `pandas.DataFrame` with four columns: 'Source', 'Target', 'Date' and 'Weight'.

	Source	Target	Date	Weight
0	Aivaloglou E.	Hermans F.	2019	10
1	Aivaloglou E.	Swidan A.	2018	2
2	Aivaloglou E.	Chen G.	2018	1
3	Aivaloglou E.	Davis D.	2018	1
4	Aivaloglou E.	Krause M.	2018	1
5	Aivaloglou E.	Hauff C.	2018	1
6	Aivaloglou E.	Houben G.	2018	1
7	Aivaloglou E.	Smit M.	2018	1
8	Aivaloglou E.	Moreno-Leon J.	2017	1
9	Aivaloglou E.	Robles G.	2017	2
10	Aivaloglou E.	Moreone-Léon J.	2017	1

Figure 2: Preview of the data set.

In the webpage we find a total of 134 researchers with publications uploaded under their name, and after parsing all of them we collected a total of 8808 collaborations between all papers, counting the links made from these initial 134 researchers with every single other person that made a contribution to their papers, we could argue that the collaborators inside a paper could be linked among themselves, but we wanted to focus our aim only in researchers inside LIACS. If any of the collaborators inside a researcher is also in LIACS then their links will also be available under their name. With this approach researchers from LIACS will have a higher degree and most likely a higher PageRank value than outside researchers.

It is also interesting to show the links with the largest weight, researchers that collaborated together more often than others, to compare if any of them are also in the most important researchers after implementing PageRank.

	Source	Target	Date	Weight
0	Kleijn J.	Koutny M.	2018	76
1	Koster B.	Koster AJ.	2013	75
2	Bäck T.	Emmerich M.	2021	50
3	Rozenberg G.	Ehrenfeucht A.	2017	44
4	Bäck T.	Michael T.	2016	42
5	Bonsangue M.	Rutten J.	2017	38
6	Bonsangue M.	Boer F.	2015	38
7	Bäck T.	Wang H.	2021	32
8	Lew M.	Bakker E.	2021	30
9	Stefanov T.	Nikolov H.	2017	29
10	Lucas P.	Hommersom A.	2018	29

Figure 3: Top links ordered by weight.

5 EXPERIMENTS, ANALYSIS AND RESULTS

Once we have the final data set we can create a graph object using the `networkx` python package. This package allows us to easily manipulate and obtain meaningful metrics from a given graph.

We compute centrality measures, such as the degree centrality, and plot the degree, indegree and outdegree distribution of the graph to get an understanding on how data is distributed. We also obtain the number of connected components (strongly and weakly), and we plot the the distance distributions for both the giant component of the graph as well as on the entire data set.

In the table 1 and figure 4 below we show some stats about the graph of the data set.

Statistic	Amount
Nodes	2889
Links	3968
Density	9.085×10^{-4}
Clustering coefficient	0.1539
Connected components	
strongly	2794
weakly	6
largest strongly	87
largest weakly	2868

Table 1: Main statistics of the graph.

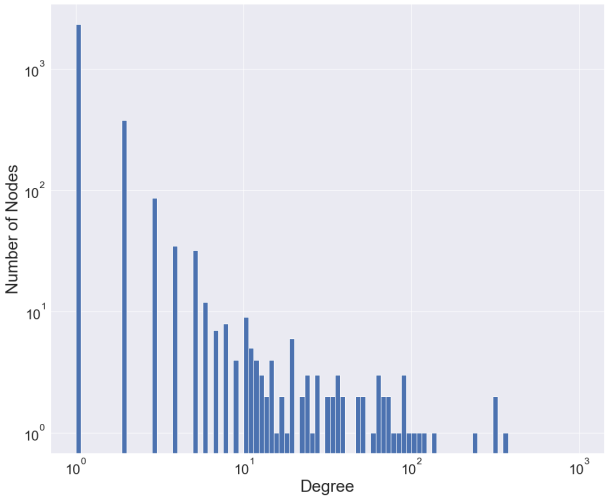

Figure 4: Degree distribution of the data set.

We can also export the data to a csv file from the previously `DataFrame` object, this way we can import this data to `Gephi` (an application with strong visualization tools) to picture the data-set. `Gephi` also allows to change the size/color of a node and link based on different measures, we apply a color palette to differentiate the first ten most populated communities and we change the size of the nodes based on the degree value.

The labels on the nodes are also proportional to the total degree of the node, so more important nodes will have more readable label. We could hypothesize that each community will differ from the rest in the sense that papers in the same community are more likely to talk about similar topics, hence it will be useful for future PHD candidates to find a suitable researcher to collaborate with in the desired field.

First we calculate the ranking of nodes using the PageRank algorithm in the data set, we can find this algorithm inside the `networkx` package, we call the function on the data-set and it returns a dictionary containing the name of the node as a key and their importance as the value.

5.1 Random Walk method Implementation

We can also implement the PageRank algorithm using a method based on random walks onto the data set, to obtain an ordered list of the highest ranking nodes in the graph. The steps followed in the implementation are as follows:

(1) We set a value of zero to all the nodes in the data set.
(2) Next we select a node out of all, randomly.
(3) We find and keep the neighbors of the selected node, then we get another node from this list and we increment its value.
(4) In the case that the node has an outdegree of 0 we need to select another node from the list. If the outdegree is greater than 0 we increment its value and repeat the previous step until convergence.

This implementation is faster since we are not looping the entire set of nodes, we are selecting a few random nodes and computing the PageRank of the randomly selected nodes and their neighbors.

We experiment with different values of iterations for the random walks implementation to see how it performs on the data set compared to the nomal PageRank implementation. The way in which we compare both methods is by comparing the amount of similar researchers there have in the top 100 most important nodes, which are the most interesting to study.

Not only we check the top 100, but also the top 50 and the top 10 to have a more meaningful insight in the trend of the amount of similarities between both ranks.

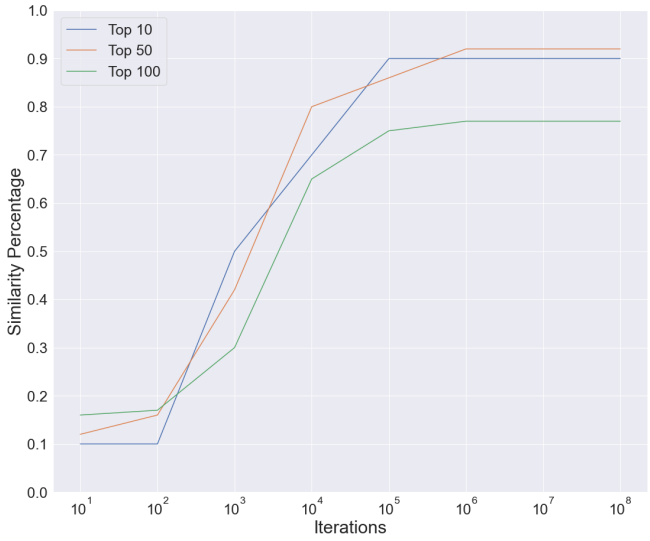

Figure 6: Percertange of similarities between both implementations.

In the plot of the Figure 6, where the 'y' axis represents the Similarity percentage between the two ranks and the 'x' axis the amount of iterations done in logarithmic scale, we can appreciate three lines, the blue represents the amount of similar items in the top 10 most important researchers, similarly the orange line for the top 50 and the green line for the top 100. We can observe the plot has an elbow shape, and the values of similarity stabilised after a specific amount of iterations, this value of iterations where the lines flatten is around $10e5$ iterations, right in the elbow of the plot. We can safely say it is not worth doing more iterations in hopes of getting a better approximation, since the computation time increases vastly with more iterations, as we can see in the plot of the Figure 7.

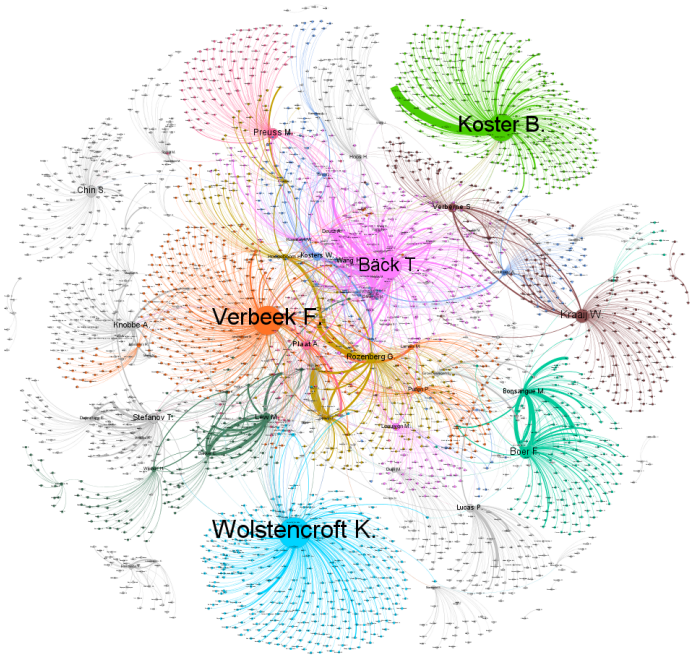

Figure 5: Visualization of the data set via GEPHI.

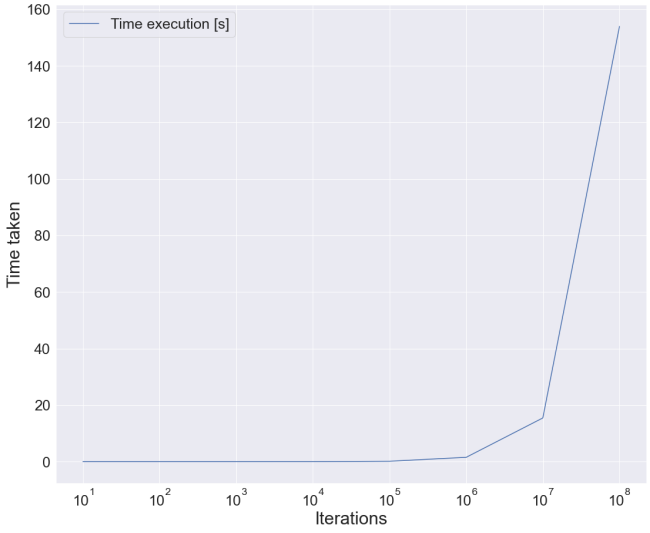

Figure 7: Time taken for every amount of total iterations.

We also made a visualization of the percentage of similarities between the exact order of both ranks, expecting a very low percentage, since one implementation can rank the same first 10 researchers in a slightly different manner and taht would result in a very poor exact order similarity value, but the results were given around a 40% similarity score, meaning 40 % of the researchers were in the exact same position in both ranks.

In the plot of the Figure 8 we appreciate a similar trend as seen in the previous Figure, where the similarity score flattens at a specific point, this being at around $10e5$ iterations, (with a peak of almost 45% at $10e6$ iterations).

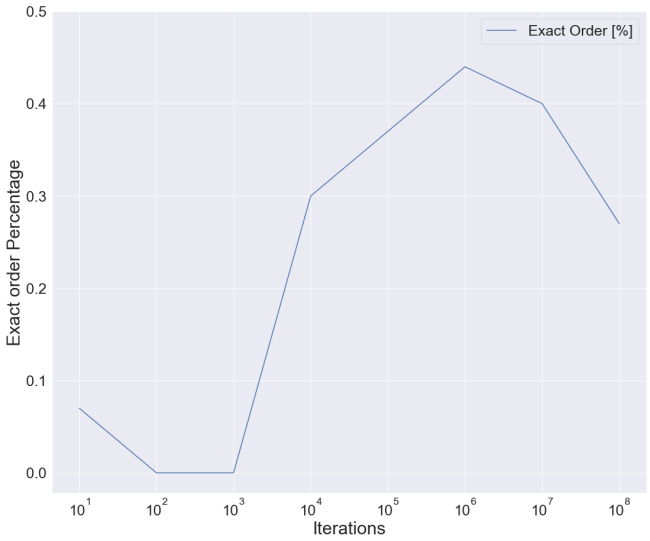

Figure 8: Percertange of similarities in the exact order of both ranks.

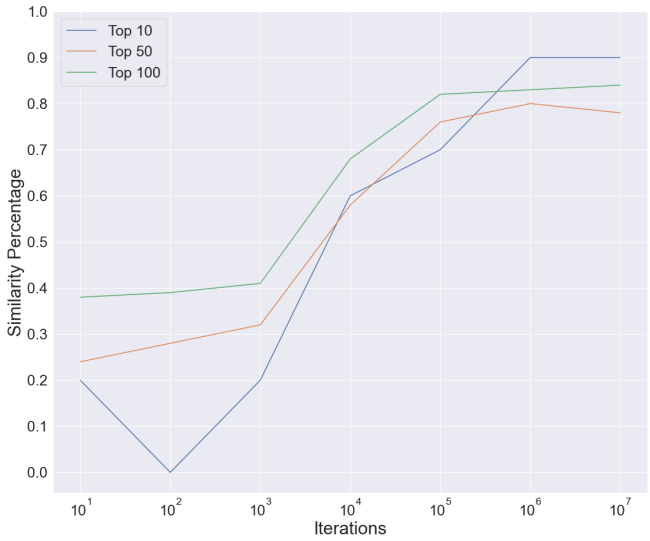

Figure 9: Percertange of similarities in the KONECT dataset.

Ultimately to corroborate our findings we experimented with another well know dataset of researchers available in KONECT, the dataset contains information about researchers in the area of social networks, and has a total amount of 1461 nodes and 2742 links, so very similar to our LIACS dataset in terms of size, both considered small world datasets. Applying the random walks aproximation and comparing it to the normal PageRank we find at first that the results are very poor, and not what we expected. Further analyzing the KONECT dataset we find out that the graph is very disperse, it has a lot of small connected components so the random walks can very often get stuck in one of these components (since we loop through the neighbors of the previously selected node, also in the connected component). For this reason we obtain the giant component of the graph and experiment on it, the results achieved now confirmed what we thought to be true because now we get a plot similar to the Figure 6 explained above, where both ranks have about a 85% similarity percentage in the top 100 researchers after 10e5 iterations

5.2 Power method Implementation

We experimented lastly with the power method approach of the PageRank algorithm, we found that the similarity percentage obtained for the first 100 ranked researchers was much higher, in the order of 98% both for the LIACS dataset and the KONECT dataset with around a 70% exact order similarity percentage for the entire data set.
Next we select a node out of all, randomly.

Overall the power method approach seems to be much more effective and efficiant compared to the random walks implementation, because it needs less time to achieve results closer to those obtained through normal PageRank.

6 CONCLUSION

In this paper, we used several random walk-based methods to classify nodes based on importance, and we tried to evaluate the performance of such methods on small world networks like the LIACS data-set.

By categorizing researchers based on their importance, we can help future PHD candidates or researchers find a suitable collaborator to work with.

Future work line ideas could include incorporating a small data set containing words related to specific computer science fields, making it as simple as typing a topic and receiving the top contributors for that topic, rather than necessarily the top contributors overall. This would necessitate parsing information from the title of each publication and storing keywords and the number of occurrences of such words for each person. The internet is getting bigger and bigger by

every click on a search engine. Every click is stored and processed by the search engines. It is not possible to perform analysis and computations on these huge data-sets on normal machines. This is where Map-Reduce [1] comes into play, Map-Reduce is a computation model for processing huge amounts of data in parallel manner, using large number of machines.

REFERENCES

[1] Adhitya Bhawiyuga and Annisa Puspa Kirana. 2016. Implementation of page rank algorithm in Hadoop MapReduce framework. *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)* (2016), 231–236.

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web. *Stanford InfoLab* (1999).

7 ACKNOWLEDGEMENT

We would like to express our gratitude to our colleague Srishankar for his assistance and constructive feedback on our initial draft of the paper as well as our implementation.

Our implementation would not have been possible without the assistance of our Professor Frank Takes.