Improvised PageRank Algorithm using Dangling Nodes and Spider Traps Taxation

Sankara Subramanian Venkatraman
School of Computing
National College of Ireland
Dublin, Ireland
x18179541@student.ncirl.ie

***Abstract*—The main objective of this research is to understand the architecture of a distributed system and implementation of PageRank and Improved PageRank algorithms. Also, the MapReduce framework is studied to work in a Distributed File System (DFS). Hadoop an open-source platform which has distributed file System (HDFS) by default is used in this research for storage and processing of files. In addition to this Hadoop architecture of Master Node, Data Node, Resource Manager and Node Managers are also considered. The key challenge in Web Mining is to understand the user and produce quality search results. The data on the web is huge, so a part of data is taken from the common crawl website and analyzed for improving the search results. Web structure is not as simple as the relational database because of the nature of data flow, datatype and frequency of the data. To understand the structure of the Web, Link Graph is constructed. Finally, the quality of Web search is improvised by analyzing the scores for valid webpages.**

***Index Terms*—Link Analysis; PageRank Algorithm; Dangling Nodes; Hadoop MapReduce**

I. Introduction

In general, PageRank (PR) is an unbiased algorithm to rank the web pages on the search engine when a user searches for a particular query. The top results displayed for the user is based on the topic of their interest. Understanding of users behaviour has a practical application and comes under the classification of Web Structure Mining (WSM). Discovery of web content is called Web Mining. Web Mining has three sections Web Content Mining (WCM), Web Usage Mining and of Web Structure Mining (WSM) [1].

This research work focuses on two challenges faced in the area of WSM. The first one is to improvise the existing PageRank algorithm developed by Google in the year 1998, and the other one is the efficiency of computing in mining web content. A lot of research has been carried out in improving PageRank algorithm.

An improved algorithm by [2] has used anchor text and title tag to distribute the value of PageRank to outgoing links based on the keyword match. This increases both credibility and relevance of the web search. In research by [3] has established an associated PageRank algorithm that uses most frequent term sets (MFST) to calculate the similarities of outgoing links of a web page. It overcomes the complexity of long querying time in the topic-sensitive algorithm. It also improves the relevance of web search and decreases the spam link effects.

In this research, the improved PageRank algorithm is implemented using link analysis and Hadoop MapReduce algorithms. The difficulties of dangling nodes and spider-trap or dead-end chains are addressed. It also compares the original PageRank with improved algorithms after several iterations. Initially, the Hadoop environment is set up in a distributed cluster with 1 master and 3 worker nodes, depending on the size of data the worker nodes are scaled up. Also, problems encountered during cluster setup, implementation and automation for cluster setup are discussed. Finally, the scores of various PageRank algorithms against different web size is evaluated.

The second section critically compares various data streaming techniques in works of literature. Link analysis such as dead-end traps, spider traps and taxation, topic-sensitive PageRank and spam farm are studied. Also, the implementation of PageRank algorithms in parallel and the distributed environment such as Hadoop is explained in detail. In the following sections DataMap, methodology, implementation, architecture, results and conclusion are discussed.

II. Background on Scalability and Parallel Algorithms

This section explains a brief history of parallel computing using Hadoop MapReduce and how the architecture of distributed file system work. Also, it describes how the PR algorithm has evolved with minor improvements.

A. Architecture of Distributed File System and MapReduce

In the era of big data, a huge amount of data being generated rapidly by almost every industry. To process these large volumes of data distributed architecture was preferred. This architecture is widely used in the area of social media web mining applications such as link graph analysis. A new storage stack called a distributed file system (DFS) which stores data in blocks of disks was developed to store big data. DFS is extensively used in Google File System (GFS) by Google, Hadoop Distributed File System (HDFS) an open-source platform and Kosmix's CloudStore. DFS stores data in chunks with 64MB size. The major advantage of this file system is data replication which avoids the major drawback of node failure. The default replication factor is 3.

The data available in DFS is processed using a high-level language called MapReduce (MR), it handles the issue of fault

tolerance in the storage layer. In the case of datanode failure or unavailability, MR job won't fail instead it will check for availability of replicated datanode. MR computation uses a single processor, in-memory and cache. It divides computation into tasks, and tasks are independent of each other. The upcoming section will explain how the MapReduce algorithm implemented in Hadoop architecture with HDFS as primary storage.

MapReduce framework is popular because of its large-scale parallel and distributed computing of data. Generally, MapReduce is split into Map task and Reduce task. In Map function, the data is read from DFS and split into chunks of a key-value pair. In the next step, the grouping of values based on a common key. Assume k is the key and v_1, v_2 and v_n are values. The key-value pair (k, v_1), (k, v_2), ..., (k, v_n) grouped to (k, [v_1, v_2, ..., v_n]). Before key-value pair passed into Reduce task, grouping and aggregation are required. The worker nodes can either process map or reducer worker at a time [7, pp. 19–22].

Hadoop can either configured in single-node or multi-node clusters. It is made of 3 daemons namely master node, data nodes and client nodes. Using MapReduce, the master node manages the parallel computation of data. It assigns works to worker nodes using JobTracker and handles the data storage. The worker nodes with a minimum replication factor of 3 stores data in different racks to avoid rack failure. It also executes the tasks assigned by the master node and communicates using TaskTracker. Finally, the client nodes maintain the cluster settings, submit MR jobs, store and retrieve data into the clusters [4].

The architecture of Hadoop MapReduce discussed by [5] has utilized two daemons name node and data node. The graph data stored in a distributed cluster processed using Map and Reduce tasks have high availability, reliability and scalability. It has used the default configuration of 1 master and 3 worker nodes. An analysis of graph and matrix computation uses Hadoop MapReduce for High performance (HPC) and High Throughput (HTC) Computing.

The proposed architecture consists of two modules namely HDFS and MapReduce. The former provides new information storage of Network Overlapped Compression (NOC) and Compression Aware Storage (CAS). Compressed Task divider and block reader are the modules provides by MapReduce. HDFS module increases the storage performance by 33% using constant decompression of data. The job processing time was decreased by 66% which allocates uncompressed information using MapReduce module [6].

Hadoop Yet Another Resource Negotiator (YARN), is a resource manager that handles resource allocation and utilization at a global level. The yarn has 3 components Resource Manager, Node Manager and Application master. The resource manager is set up in the master node and the node manager that manages the resources at the machine level is installed in worker nodes. Also, the application master monitors the application Id and manages the containers.

B. Applications of MapReduce & Datastreams

Mathematical applications such as relational algebra, matrix addition and multiplication can be implemented using MR parallel algorithm. The degree of replication can be optimized by implementing Lagrangean multipliers using a single MapReduce job. In optimization problems if a function has equality constraints, Lagrangean multipliers strategy applied to find the local maxima and minima. Similarly, fact-dimension table joins in query language can be joined by Multiway using the MR algorithm. In graph analysis, nodes that represent incoming and outgoing links are resolved using the MR algorithm [7, pp. 64–66].

Streaming data is entirely different from processing data in the traditional Database Management System (DBMS). Unlike DBMS, it has no fixed datatype, frequency of data and data size. Generally, streaming data is processed using two-step storage. The first one is archival storage and the other is working storage it can either be disk or in-memory. Retrieval of data from archival storage is a time-consuming process and mostly not preferable for stream processing. So, the data stored in limited working storage produce the results based on ad-hoc queries to the user for streaming data. Applications such as the Global Positioning System (GPS), Image data from satellites, Weather stations and Internet traffic data use streaming data sources.

The major limitation of stream processing is memory and size of the data. If sufficient memory or disk not available to process the real-time data, then the data will be moved to archival storage which is hard to process. So, when working with streaming data, there is a tradeoff between accuracy and processing time. There are two generalizations for streaming data by [7, pp. 125—127]. First, an approximate solution is much efficient than an exact solution. The second one is the usage of the hashing function that introduces randomness to the existing algorithm improves the approximability.

C. Web structure & Improved PR Algorithms on MapReduce

Before going to Link analysis it is necessary to understand the structure of the Web. Web structure consists of 3 main components

1) Strongly Connected Component (SCC)
2) In-Component
3) Out-Component

In real-time, these components are not interconnected. The In-component can reach SCC, whereas SCC cannot reach in-component. In Contrast, out-component are reachable from SCC but the reverse is not possible.

The 3 minor components are

1) Tendrils
2) Tubes
3) Disconnected components

Tendrils are of 2 types tendrils in and tendrils out. Tendrils in are reachable from in-component and Tendrils out can reach out-component respectively. Tubes are the pages that reach both in component and out component except SCC. Finally,

disconnected or isolated components are unreachable from any of the above 5 mentioned pages. [7, pp. 159—161].

Web Mining is a vast field that deals with numerous applications and algorithms. Out of various algorithms, Google PageRank (PR) which was developed by Lawrance Page and Sergey Brin is the popular algorithm even today. It uses a web link graph to improve the quality of the search engine queries. In research [8] has adopted the anchor text analysis that describes the accuracy of a web page developed by the World Wide Web Worm (WWWW). The first-page rank algorithm has considered the random surfer and number of citations a website acquired. It is calculated using the below formula. d represents the damping factor and default value is 0.85.

$$PR(A) = (1-d) + d\sum(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$$

Similarly, weighted PageRank by [1] have used the popularity of the webpage to calculate the score. The popularity of the website depends on the number of links pointed to that webpage. Instead of evenly distributing the score to all links, it assigns a larger score to popular (important) links. The popular links pointing to other pages get the score proportionally. The weighted page rank is formulated using

$$PR(A) = (1-d) + d \sum_{v \in A(u)} (PR(A) W^{in}_{(v,u)} W^{out}_{(v,u)})$$

$W^{in}_{(v,u)}$ and $W^{out}_{(v,u)}$ represents the popularity of inlinks and outlinks.

Research work by [9] to improve basic PR algorithm based on the web page similarity, webpage update rate and topic relevance. It is denoted by the formula

$$PR(A) = (1-d) + d \sum_{L \in B_A} (PR(L).W(L,A) + \alpha.Topic(Q,A) + \beta.T(A)$$

B_A represents a collection of outlinks from webpage A. W(L, A) denotes the similarity of webpages L and A. The similarity is calculated using (Tf-IDF) which is the Term Frequency Inverse Document Frequency technique. The topic of relevance T(Q, A) is calculated using a BM25F probabilistic model for information retrieval. BM stands for Best Model. It calculates the correlation of headline, text and anchors text of webpages L and A. Finally, the factor of page update rate is calculated using Poisson distribution assuming the fact that webpages are not frequently updated. The proposed PR algorithm improves precision and search results.

III. Data Map

Datasets are sourced from Common Crawl website [10]. February 2020 data is used in this research. Initially, Common Crawl website crawls data in archive format (ARC). Currently, it uses 3 different formats Web Archive Format (WARC), Web Assembly Text Format (WAT) and WinSQL Export Template (WET) format. The raw data crawled is stored in WARC files and metadata of WARC format is stored in WAT files. WET files stores plain text data of WARC format.

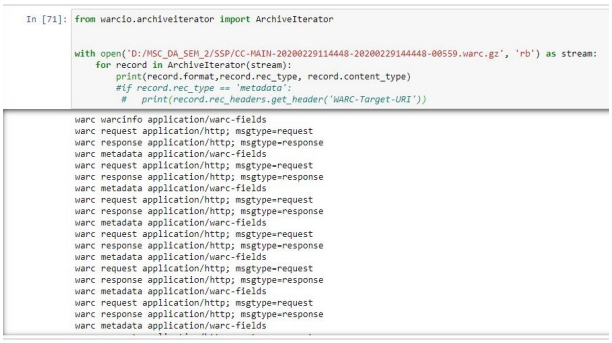

Fig. 1. WARC File Formats

WAT file format that contains metadata of the WARC files has three types of records (request, response and metadata). For implementing PR algorithm WAT files are analyzed by processing the metadata of **Request** record type. Fig.1. and Fig.2. represent WARC and WAT file record types from a sample respectively.

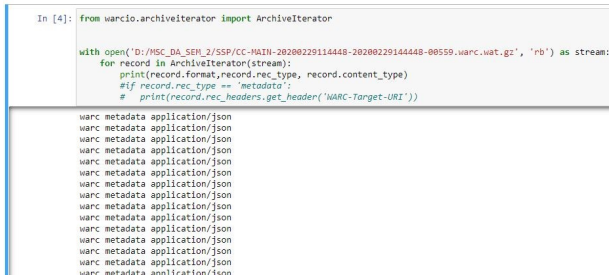

Fig. 2. WAT File Formats

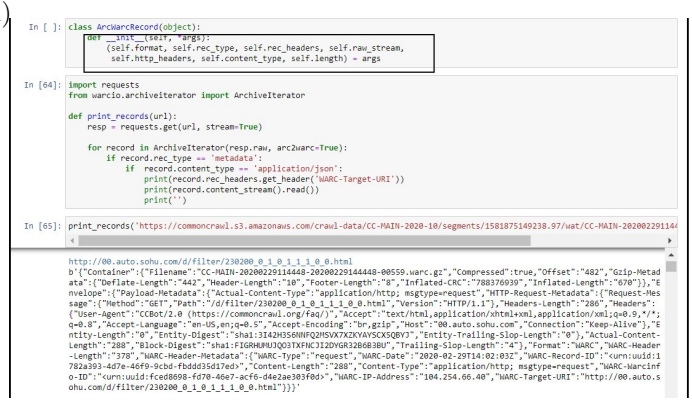

Fig. 3. WAT JSON Sample Record

The record type of WAT file is metadata and the contents are stored in JSON format. A sample record of WAT file is shown in Fig.3. ArcWarcRecord class of warcio can read the information of the following in WAT record.

- rec_format
- rec_type
- rec_headers
- raw_stream

- http_headers
- content_type
- length

The records in the archive format of WAT files are structured using warcio module available in python3 shown in Fig.4. It helps to read data from common crawl AWS S3 buckets without using boto libraries. Also, the usage of the disk space in HDFS is not required and process the data in-memory. page_url selects webpage and links represents outgoing links from the webpage.

Fig. 4. WAT Inlink and Outlinks

Fig. 5. Feb 2020 WAT Files

IV. Methodology

As discussed earlier, WAT format is the best choice for implementing PR algorithm in Hadoop MapReduce. The idea of implementing PR algorithm using Hadoop MR job is inspired from [11]. This research analyse and improves the existing PR algoritm with 2 different solutions. Initial PR score is calculated by default formula.

$$PR(A) = \frac{(1-d)}{N} + d \sum_{B \in A} \frac{PR(B)}{out(B)}$$

A. Dangling Node

As discussed in the Web structure, dangling nodes or sink nodes are webpages with no outgoing links. It is called as Isolated or disconnected webpages. The web consists of a lot of disconnected webpages. The existing PR algorithm gives scores equally to all webpages irrespective of its nature. But in the proposed algorithm, the scores allocated to sink nodes are distributed to webpages with outgoing links implicitly.

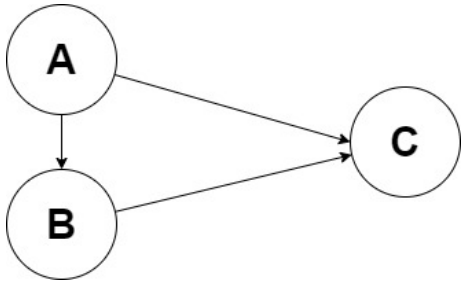

Fig. 6. Dangling Nodes

As mentioned by Vectore in [12], assuming that sink nodes are connected to whole web. The probability that a surfer goes to page A (or) B from sink node C is 1/N. The improved Dangling node PR algorithm is formulated by

$$PR(A) = \frac{(1-d)}{N} + d \sum_{B \in A} \frac{PR(B)}{out(B)} + d \sum_{C \in \varnothing} \frac{PR(C)}{N}$$

B. Spider Traps and Taxation

Web pages with no dead-end or chains of dead-end are termed as Spider Traps. Fig.7. shows spider trap structure. These web structures are formed intentionally or unintentionally, but it maintains the score of PR within the trap. In general, this issue is addressed by taxation technique.

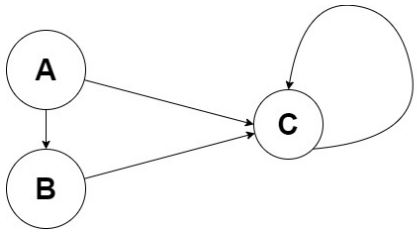

Fig. 7. Spider Trap Structure

In this research, spider traps are modified by checking the outgoing link of a website based on 2 conditions. The first one is to check whether the existing webpage has only 1 outgoing link. The second condition is to examine whether the webpage is pointing to itself. If the above 2 conditions are satisfied, then the PR score of 0 assigned to that webpage. The PR score is calculated by

$$PR(A) = \frac{(1-d)}{N} + d \sum_{B \in A} \frac{PR(B)}{out(B)}$$

C. Ethical Implications

WSM uses the content of the webpage, title, anchor text and HTTP response for analyzing the webpage reputation. The data available in the common crawl S3 bucket is a trusted open-source repository and ethically sourced for computation. Also, there is no personal information mining carried out in the

analysis. The proposed PR algorithm doesn't apply any kind of bias to calculate the scores. The scores are calculated based on the title of the webpage and corresponding outgoing links. The datasets are processed and analyzed using python and Hadoop packages which are open-source platforms recommended for scalable datasets.

V. Implementation Architecture

1. Implementation

1) The key challenge of this research is link-graph creation. In this pre-processing step MR job of mrjob.job split the tasks into map and reduce. Two mapper and reducer tasks are applied, one for link graph creation and the other one is to normalize the graph for each webpage.
2) In Mapper of linkgraph, try-catch exceptions are implemented to remove invalid links. The .netloc function of urlparse library iterates within a for loop (key, value) pair of (website, list(outgoing links)) is generated.
3) In the reducer phase of linkgraph, for each webpage; id, state and outgoing links in JSON format are formed. The structure of Reduce output looks like (webpage,webpage, state, outgoing links).
4) The normalize function iterates through the output of reducer from linkgraph. Lambda function is applied to filter webpage without webpage, state, outgoing links value.
5) In the next stage, the output of linkgraph passed to calculate the web size. Simultaneously, the output of linkgraph is then passed to the initial PR calculation, where all the nodes receive equal scores of 1/N.
6) As suggested by [13], the basic PR algorithm converges after 15 iterations are initiated for 10 warc.wat.gz files. The PR algorithm ran for 15 iterations, and their corresponding PR score in each iteration is tabulated. Also, the size of the web and the time taken to run each job is determined. The scalability is examined by increasing the size of the input files by a unit of 10.
7) The fourth step is to calculate the dangling node PR score. If the outgoing link length is 0 then its state value is summed. In the improved dangling PR algorithm, graph size and dangling score are passed as arguments and new PR score is calculated.
8) Finally, the spider trap PR score is calculated and compared with the actual PR score and dangling node PR scores.

2. Architecture

Initially, Hadoop-3.1.3 is set up in distributed clusters with 1 master node and 3 worker nodes configuration. Based on the size of the web, slaves nodes are scaled horizontally. The architecture diagram in Fig.7. explains how Hadoop components are configured with HDFS layer, MapReduce Layer and YARN Layer. The configuration of Python3 across the nodes with required packages are installed using pip install. Master and data nodes are computed with below specifications. The size of the data for 10 files is almost 1.1 million, so 2 more datanodes are added to the existing architecture.

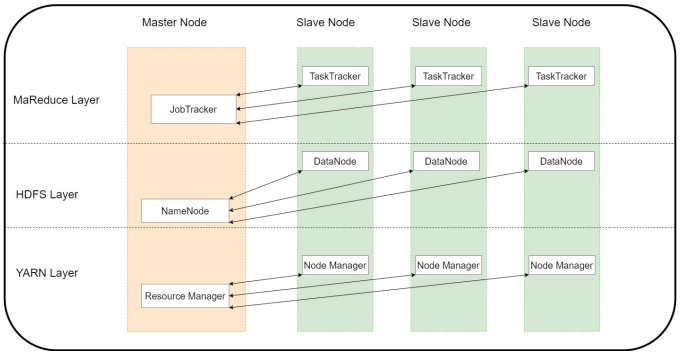

Fig. 8. Hadoop Architecture

Machine Specification:

- Machine_Type - Ubuntu-Bionic 18.04.3 LTS
- RAM - 8 GB
- Virtual CPUs - 4
- Hard disk - 80 GB

VI. Evaluation and Results

A. Cluster Configuration

In terms of architecture, distributed cluster setup of hdfs-site.xml, core-site-xml and Hadoop-env.sh files are studied in detail. Automation script is prepared to configure Hadoop in new datanode from existing ones to reduce cluster setup timing. In terms of configuring Hadoop in AWS cloud stack, few challenges were faced in firewall settings. Log management of Hadoop is analyzed to understand the configuration and application logs.

The configuration logs explain the setup of Hadoop in namenode, secondary namenode, a resource manager in master. Datanode and node manager in worker nodes. The application in user logs folder describes the MapReduce Job-related logs and standard error (stderr) shows error related to the job submitted from the master node.

B. Dataset and Output from PR algorithm

At first, the PageRank algorithm ran for 1 iteration from end to end. The total time taken is 13 minutes 49 seconds for 10 files. The size of the web is 1183423. The initial PR-score is 8.45006392473359E-07. A random 5 samples are taken without any bias and its 15 iterations are analyzed.

Fig.9.shows how the webpage scores converge for 15 iterations. After the 2nd iteration, the convergence rate decreases slowly. Website 0.ssbet077.cn converges after 9^{th} iteration, codex.core77.com at 11^{th} iteration and dailyvoice.com at 15^{th} iteration. Fig.10. represents dangling PR-score distribution.

For Improved dangling node algorithm, the PR scores of dangling nodes are distributed equally to other nodes, so the convergence rate is slow after the 1st iteration. Dangling PR-score value is 0.79317. Website 0.ssbet077.cn converges after 8^{th} iteration, codex.core77.com at 11^{th} iteration and

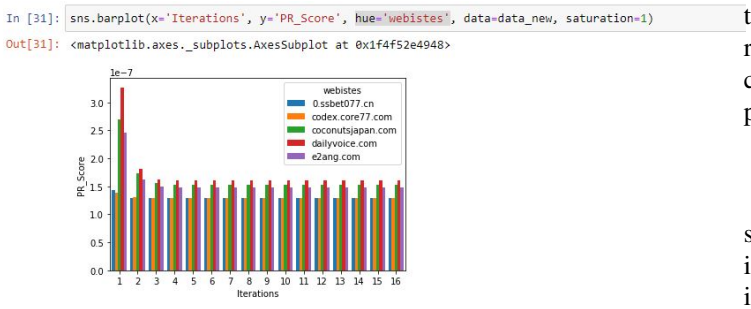

Fig. 9. PR-score Distribution

dailyvoice.com at 14^{th} iteration. Out of 5 webpages chosen, codex.core77.com has no outgoing links, which is a dangling node.

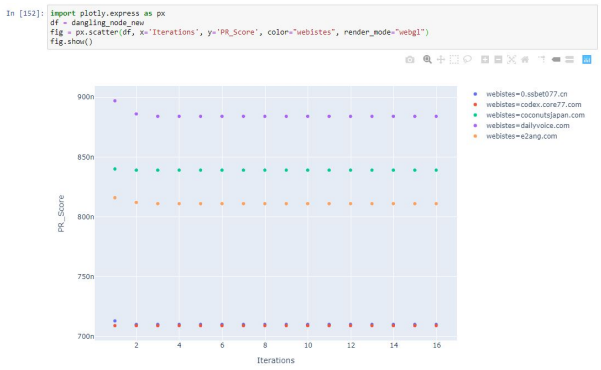

Fig. 10. Dangling Node Convergence

Dangling PR score for 20 files is 0.80379 and the web size is 1783442.The box plot in Fig.11. explains how much time it took to run each job with 1 iteration, 15 iterations and overall. We have compared the configuration of datanodes against the time taken to run jobs with different algorithms.

Fig. 11. Bench-marking of DataNodes

Initially, the configuration of 1 master and 3 datanodes didn't work for 1.1 million data due to java heap memory. So the configuration is increased to 5 datanodes. As the size of the web increases to 1.7 million, 6^{th} node is added, but there is no decrease in processing time. There are several factors to be considered while running an MR job. If a mapper or reducer fails in any datanode, the tasks start again with a new container in the same node or new datanode. It increases the processing time.

VII. Conclusion & Future work

In this research, various data mining process used in large scalable systems is studied. Different algorithms and processing of stream data from the web are discussed. Python MR implementation in the calculation of the PR algorithm helped us to create linkgraph and analyze the PR score. The average convergence of default PR algorithm is 2.25E-07 and dangling nodes is 7.95E-07 from initial PR score 8.45E-07 respectively after 1^{st} iteration for sample webpages. Even though the dangling PR algorithm convergence rate is slower, the problem of dangling nodes is resolved by distributing equal scores to valid webpages.

The architecture of the distributed system and its complex structure are explored. From this implementation of PR and improved dangling node algorithm, the web pages can be sorted according. Spider trap and biased random walk algorithms are studied, but the implementation of these algorithms considered for the future scope of this research. Also, scalability of the existing algorithms will be tested with more files and vertical scaling of datanode to reduce the storage disk.

References

[1] W. Xing and A. Ghorbani, "Weighted PageRank algorithm," Proc. - Second Annu. Conf. Commun. Networks Serv. Res., pp. 305–314, 2004.
[2] Z. Hao, P. Qiumei, Z. Hong, and S. Zhihao, "An improved pagerank algorithm based on web content," Proc. - 14th Int. Symp. Distrib. Comput. Appl. Business, Eng. Sci. DCABES 2015, pp. 284–287, 2016.
[3] C. C. Yen and J. S. Hsu, "Pagerank algorithm improvement by page relevance measurement," J. Converg. Inf. Technol., vol. 5, no. 8, p. 17, 2010.
[4] https://pubs.vmware.com/datadirector-25/index.jsp?topic=%2Fcom.vmware.datadirector.hadoop.doc%2FGUID-A9288BC4-AB74-405B-BC06-402FA3B9E1F4.html
[5] P. P. Talan and K. U. Sharma, "An overview and an Approach for Graph Data Processing using Hadoop MapReduce," Proc. 2nd Int. Conf. Comput. Methodol. Commun. ICCMC 2018, no. Iccmc, pp. 59–63, 2018.
[6] N. Ramakrishnaiah and S. K. Reddy, "Performance analysis of matrix and graph computations using data compression techniques in mpi and hadoop mapreduce in big data framework," 2017 IEEE Int. Conf. Smart Technol. Manag. Comput. Commun. Control. Energy Mater. ICSTM 2017 - Proc., no. August, pp. 54–62, 2017.
[7] J. Leskovec, A. Rajaraman, and J. D. Ullman, "Mining of Massive Datasets," Min. Massive Datasets, 2014.
[8] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine BT - Computer Networks and ISDN Systems," Comput. Networks ISDN Syst., vol. 30, no. 1–7, pp. 107–117, 1998.
[9] W. Yang and P. Zheng, "An improved Pagerank algorithm based on time feedback and topic similarity," Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS, vol. 0, pp. 534–537, 2016
[10] https://commoncrawl.org/2020/03/february-2020-crawl-archive-now-available/
[11] https://eliteinformatiker.de/2016/05/01/analyzing-the-commoncrawl-using-mapreduce
[12] https://www.youtube.com/watch?v=_Wc9OkMKS3g&list=PLBv09BD7ez_6b7342XmYBMzPXwJ2YAWRN&index=7
[13] S. Kamvar, T. Haveliwala, and G. Golub, "Adaptive methods for the computation of PageRank," Linear Algebra Appl., vol. 386, no. 1-3 SUPPL., pp. 51–65, 2004.