

ABSTRACT

Search engines are one of the day-to-day daily attributes that are used for daily activities. It is used in such a way to find the particular contents on web as well as it can rank the fetched pages in a useful manner. The search engine fetches web pages that are stored from the database when a user searches for a specific content. Most of the search engines crawl the web pages and they build an inverted index by listing all the words or other strings found in crawled page. When a user searches for a query, the terms that are used in the query are used to search in the inverted index and all the web pages that contains the terms are returned. After fetching the web pages Trust Rank is calculated to avoid spammed pages and it is matched with the Page ranking value. The overall rank of a web page and the fetched pages are sorted in descending order and displayed according to their ranks. Hence Page-ranking plays an important role in case of a search engine. To process huge dataset for page ranking map-reduce model is used.

The main objective of the project “**Implementation of Topic Sensitive Page Rank Using Map-Reduce**” is to implement topic sensitive page rank using the map-reduce model with the help of spark in a standalone computer. We aim to accomplish for the underlying implementation by:

1. Crawl sastra.edu website using Scrapy package in python
2. Pre-process the data generated in form of edge list
3. Use the edge list to:
 - a. Visualize the graph
 - b. Calculate page rank and Topic Sensitive page rank and list them

The project was completed with:

1. Extracting hyperlinks from particular domain for the dataset.
2. Pre-processing of the dataset to form edge list.
3. Added Visualization of graph using Gephi tool.

4. Output of page rank and topic sensitive page rank of top ten pages from the graph using map-reduce model with the help of pyspark for sample and actual dataset.

Specific Contribution

- Creating crawler using pyspark package and generating dataset.
- Applying topic sensitive page rank and page rank algorithm using python for pre-processed dataset using python.

Specific Learning

- Algorithm and implementation of page rank and topic specific PageRank using MapReduce.
- Installation and usage of spark.
- Create crawler to crawl hyperlinks.

CHAPTER 1

INTRODUCTION

The technique used to crawl a part of the web and allocate actual values to web pages in the Internet to rank them in order is PageRank. It uses a simple logic as to allot a higher PageRank to the most dominant and main web page. There exist two estimations in computing for every page obtained from the database of a search engine which are Trust Rank and PageRank. Properties of a web such as the number of hits, the location of the search terms and the way in which search query is related and paired with the contents of the search engine are used for determining the Trust Rank of a web page. Once the web pages are fetched and Trust Rank is calculated, it is compared with the PageRank value. According to their ranks, the inclusive and overall rank of a web page and the fetched pages are categorized and displayed. In an abstract, a Trust Rank verifies that the page is not a spam and PageRank informs us about how important the website is on the web.

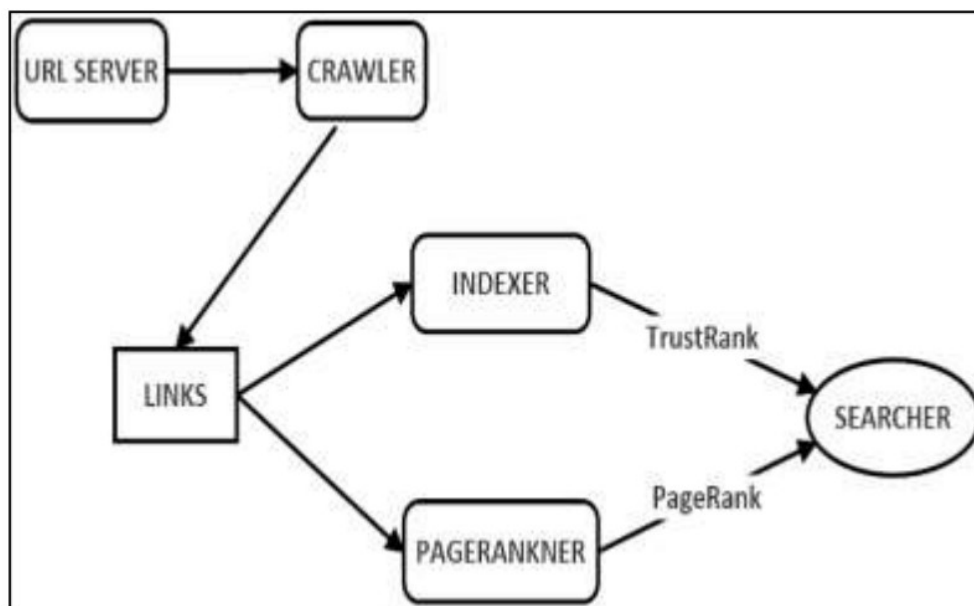


Fig. 1.1: Basic design structure of Search Engine

1.1 WHAT IS TOPIC SENSITIVE PAGE RANK?

Topic sensitive page rank also known as personalized page rank is a type of page rank which gives a biased output of the page rank for web pages that have same topics but different meaning. In other words, two or more different topics can have same query to browse in the search engine. One such example of a searching a query is “Bat”. It can mean both sports bat or also mean animal bat. To avoid these confusions topic sensitive page ranking is introduced. It is done by giving priority to some set of pages based on topics. If I wanted to see sports bat more when I search the query “Bat” then pages that are containing those topics are given priority.

1.2 WHY MAP-REDUCE IMPLEMENTATION?

Search engines such as google process huge amounts of data every day to bring desired outputs for user. To process such huge amount of data, map-reduce algorithm was used. Map-Reduce is a procedure and a program model for distributed computing dependent on java. The Map-Reduce calculation contains two significant errands, to be specific Map and Reduce. Map takes a set of data and converts it into another set of data, where singular components are separated into tuples (key/value sets).

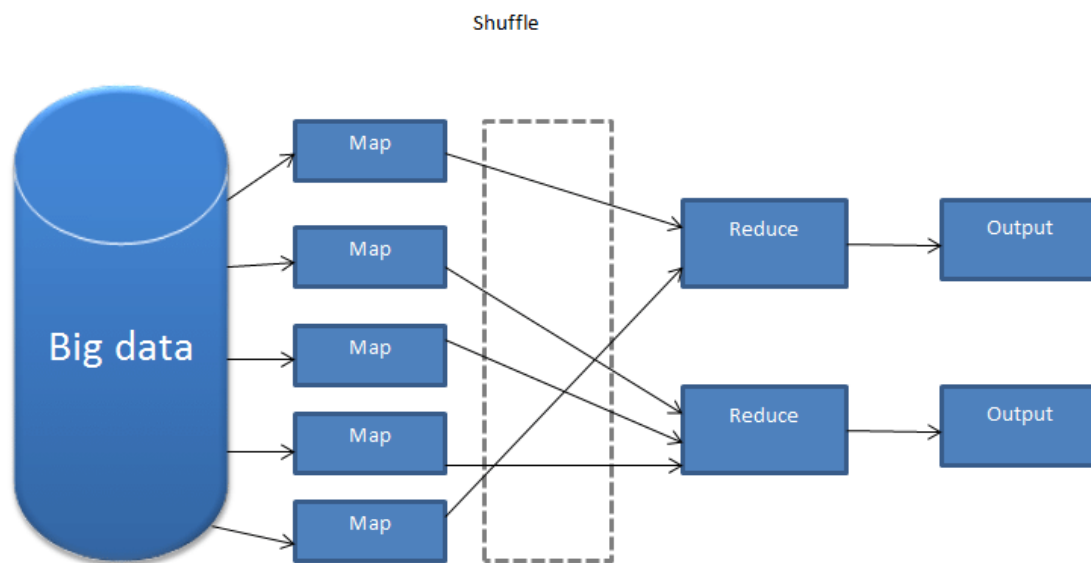


FIGURE 1.2 MAP-REDUCE ARCHITECTURE

The pros of using map reduce model is that it is easy to scale huge data processing over multiple nodes. Under the MapReduce model, the information handling primitives are called mappers and reducers. Breaking down an information preparing application into mappers and reducers is in some cases nontrivial. However, when we compose an application in the MapReduce structure, scaling the application to run more than hundreds, thousands, or even huge number of machines in a cluster is only a configuration change. This simple scalability is the thing that has pulled in numerous software engineers to utilize the MapReduce model. This is done with the help of Apache Spark application with prebuilt for Hadoop.

1.3 SOFTWARE SPECIFICATION

Following are the software used:

1. **Python:** Python has been the go-to language for the developers who does data analytics. It is easy to utilize, straightforward and it has various statistical and numerical libraries. It is also easy to visualize the processed data. Python version 3.7.4 is used.
2. **MS Office Excel:** Excel is the mostly used spreadsheet software ever since it is developed by the Microsoft. It stores the data in rows and columns and is used to perform data manipulation using mathematical functions.
3. **Gephi:** Gephi is an open-source visualization software that uses NetBeans platform to analyse and visualize networks and graphs. It is written in java. Gephi version 0.9.2 is used.
4. **Apache Spark:** Spark is used to implement Map-Reduce and comes with prebuilt for Hadoop 2.7. It provides an interface for programming clusters with fault tolerance and parallel computing. Spark version 3.0.1 is used.
5. **Visual Studio Code:** It is a free open-source editor made by Microsoft for multiple OS. It is easy to debug various types of codes in it. It got extensions that support different file formats.
6. **Anaconda3:** Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment.

Python packages used:

1. **Re:** Regular expression package in python for data pre-processing. It provides regular expression matching operations.
2. **Sys:** The python sys module provides functions and variables which are used to manipulate different parts of python run time environment.
3. **Operator:** The operator module exports a set of efficient functions corresponding to the intrinsic operators of python.
4. **Pyspark:** This is a python API written in support of Spark which is in Scala. This package helps to script spark application.

5. **Scrapy:** It is a free open-source package in python to crawl websites and extract various data in it. It was originally created for web scrapping.
6. **Pandas:** Pandas is an open-source python library used for data analysis and manipulation. It is used to read data sets and convert them into data frames.
7. **Numpy:** Numpy is a library for the Python, adding support for large, multi-dimensional arrays and matrices, besides that it has a large collection of high-level mathematical functions such as mean, median etc., to operate on these arrays.
8. **Random:** Python offers random module that can generate random numbers. These are pseudo-random number as the sequence of number generated depends on the seed.

1.4 HARDWARE SPECIFICATIONS

OS	:	Windows 10 Home (x64 bit)
Processor	:	Intel Core i7-9750H CPU @ 2.60GHz 2.59 GHz
RAM	:	16 GB

CHAPTER 2

OBJECTIVE

To develop a Topic sensitive application with the help of Map-Reduce. To do that we first require a dataset of web pages in form of network graph. Also, we need a set of pages that are based on a topic to give more priority to the pages.

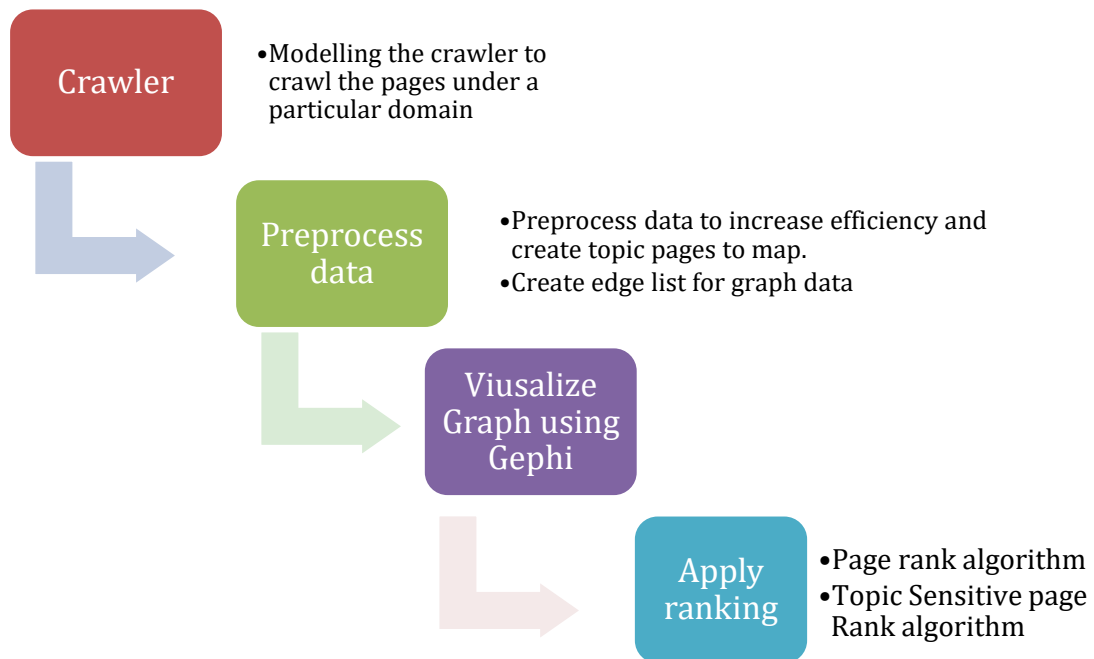


Figure 2.1 Work flow of the project

As mentioned in the figure 2.1 above, the basic ideology is to create a dataset for the project using SCRAPY crawler and process the data in such a way to form edge list. Using the edge list generated it is used to visualize the graph. Following these steps would give the solution for the main objective which is applying the ranking algorithms to get the results using the Map-Reduce model.

CHAPTER 3

METHODOLOGY

Handling enormous dataset is a challenge. A standalone system can't deal with the registering necessity for that sort of calculation. So, Google thought of a unique yet simple programming model called Map-Reduce and that can be adequately utilized for handling of a gigantic dataset. Map-Reduce doesn't depends on the idea of scaling up rather it depends on scaling down. Improving the process of a hardware system is much expensive than add multiple hardware systems that are cheaper.

3.1 SCRAPY CRAWLER

Crawling a web is a necessary process for page ranking. A web graph can be created by crawling URL that contains references to multiple URLs. A base URL consists of part of a domain. This process is recursive until all the URLs are fetched. A base URL is the consistent part of your web address. By fetching the base URL of the selected page and now joining the base URL to the relative URL and to the absolute path that is formed. Web server contains paths of pages that are called relative URLs. Web server is called to find them. So, during scraping we need to find the absolute path from the relative URL and then only the page is added to the queue. Every URL has many parts.

Most of the URLs that are fetched to implement the page ranking algorithm are useless and hence we need to process only the meaningful part of the URLs only. Scrapy takes of the task based on the rules that we set in the link extractor to crawl the data. Here the domain that is used to crawl the data is "sastra.edu". The crawled data is then stored in a csv file format in which the parent node maps multiple web pages as in one-to-many format as in the webpage that contains multiple hyperlinks that jump to different pages. An example is shown below in Table 3.1. The HTTP response status of the parent pages are also returned so that it validates only pages that are available (i.e., with HTTP status code 200). The parent node is named URL and the child nodes are named linked URLs that contains an array of child notes. The child node refers to the multiple webpages in parent URL. There are also nodes or cells that are empty which doesn't map to any other pages. Pages are pre-processed later.

Table 3.1 Format of crawled data stored in CSV

HTTP STATUS	LINKEDURLS	URLS
200	['LINK1', 'LINK2', 'LINK3', ...]	PARENT_LINK1

Assume 2 pages are there in www.example.com. For example, www.example.com/a.php and www.example.com/b.php. Presently we perform 2 search queries in these pages and passed with a GET inquiry and move to a part of next page. So, assume the 2 queries are like

```
-www.example.comna:php?q=query1#k
-www.example.comnb:php?q=query2#s
```

These two queries refer to two different websites but actually pointing to same page. This costs time and such instances must be avoided by processing these hyperlinks in such a way that only www.example.com is used for the page rank algorithm. In our crawler code we have applied not to take links that have JavaScript reference. So that the JavaScript related files can be avoided. The command to crawl using scrapy for the domain is “*scrapy crawl graphspider*” where graphspider is the name of the crawler.

3.2 PRE-PROCESSING DATA

The most important form of data to create a network graph is the edge list. To form the edge list a python program named converter.py is done. It creates 3 different types of files from the data produced by the crawler. The files are:

1. Edge List
2. Node Data
3. Topics (for topic sensitive page ranking)

3.2.1 EDGELIST

The edge list file is saved in both text(.txt) and excel file(.csv) format. Ranking algorithms uses only the text file to read and rank the pages. This file is stored in the format of source to destination as shown in sample table 3.2 below.

Table 3.2: Sample Format of edge list

SOURCE	TARGET
0	1
0	2
0	3
1	0
1	3
2	4
3	1
3	2

The node values are also modified in such a way that each URL present in the crawled data of both URL and linked URLs are given a unique integer number as a reference. Thus,

helping to reduce the size of edge list file. As mentioned previously, URLs that are mapping to empty nodes are also removed so the data is now cleaned.

3.2.2 NODEDATA

This file is saved in excel file format and is used to hold the unique integer node number of the URLs in an ordered fashion. An example of format in which it is stored is shown below in sample Table 3.2. This file is used later after calculating the page rank to return the top 10 webpage URLs not the node numbers.

Table 3.3: Sample Format of NODE DATA

URL	INTEGER
Example1.com	0
Example2.com	1
Example3.com	2
Example4.com	3
Example5.com	4

3.2.3 TOPICS

The Topics file is saved only in text format where each line in the topics file contains the node value of particular pages that are to be biased and given more priority. Here the topics file that is generated is obtained randomly using the random method for the huge dataset but generally in search engines those page URLs are stored in databases that are stored for each topic. In such a way the topics gets biased. For the sample format lets take nodes 1 and 2 are stored in topics file.

3.3 VISUALIZATION USING GEPHI

As mentioned before Gephi is a data visualization tool for network graphs that is built on NetBeans platform in Java language. Using this a simple graph can be visualized in multiple ways. For our objective this tool can be used to visualize a huge data. One disadvantage of using this tool is that it requires more memory to allocate so that it eats up RAM space. The more the number of nodes and edges the more it gets slower to visualize the graph. As an add on to this tool, it also contains in built page rank and HITS calculation algorithm that can calculate based on the graph we provide. Since it is a network graph of crawled URLs the graph must be directed. If the graph isn't directed then the page rank algorithm in both the Map-Reduce implementation and Gephi tool implementation will consider the edges directed to both the nodes. We can also view distribution graphs for larger datasets. Figure 3.1 shows the sample directed graph. Table 3.4 gives the output of page ranks, hubs and authority values of different pages. HITS is a link analysis algorithm that rates web pages.

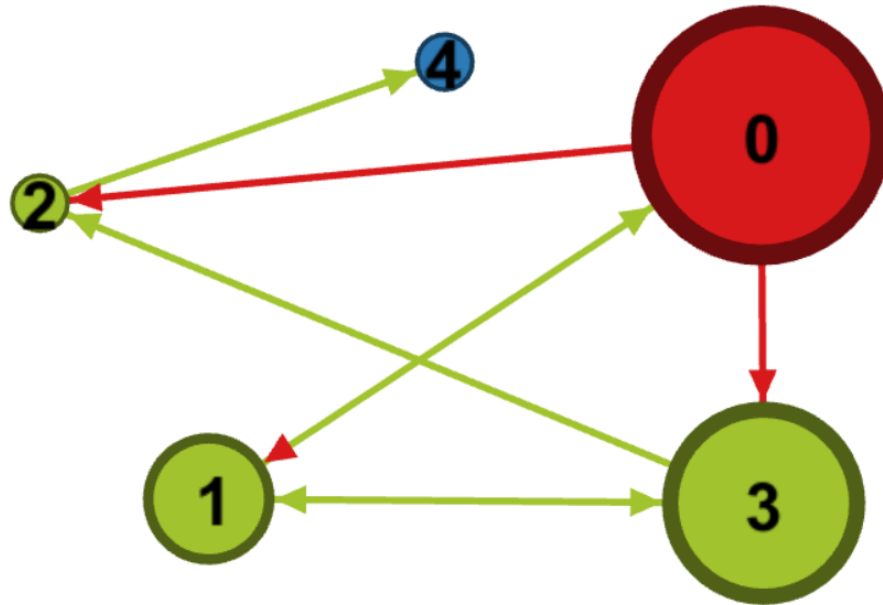


Figure 3.1 Sample directed web graph visualized in Gephi

Table 3.4 sample output of PageRank, authorities and hubs

Node	Authority	Hub	PageRank
0	0.127737	0.780454	0.156362
1	0.612025	0.279604	0.200665
2	0.612025	0	0.200665
3	0.484288	0.559207	0.200665
4	0	0	0.241644

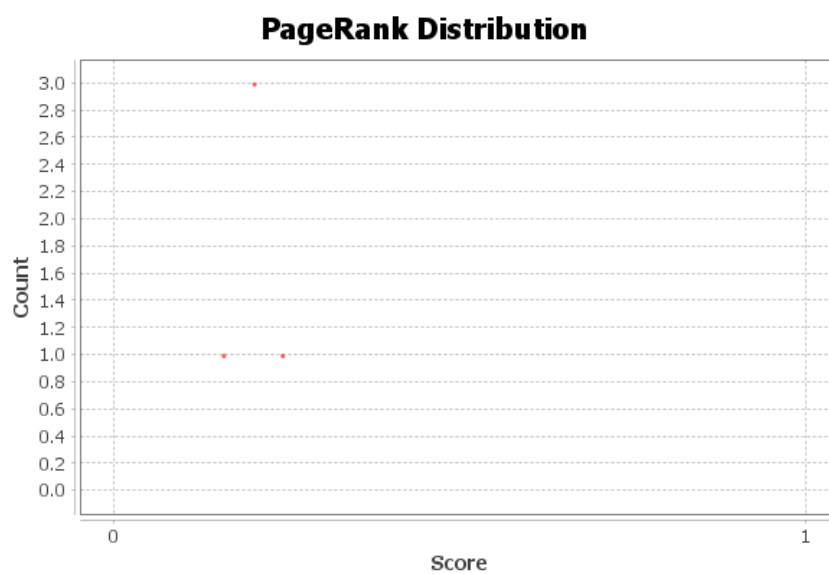


Figure 3.2 Sample PageRank distribution graph

3.4 TOPIC SENSITIVE PAGE RANK USING MAP-REDUCE

To deeply understand about topic sensitive page rank first we have to understand how page rank algorithm works. PageRank is a function that assigns a real number to each page in the Web (or at least to that portion of the Web that has been crawled and its links discovered). The intent is that the higher the PageRank of a page, the more “important” it is. A method for rating the importance of web pages objectively and mechanically using the link structure of the web.

3.4.1 PAGERANK

The links between pages are represented by a graph. A node represents a webpage and an arrow from page A to page B means that there is a link from page A to page B. Which means, the number of out-going links is an important parameter. We use the notation “out-degree of a node” which means number of outgoing links from a page. The graph is usually referred to as the web graph. Each node in the graph is identified with a page.

The simplified PageRank algorithm is: Initialize x to an $N \times 1$ column vector with non-negative components, and then repeatedly replace x by the product $R(v)$ until it converges.

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

Figure 3.3 Simplified page rank formula

- u : A web page
- B_u : The set of u 's backlinks (in links)
- $R(v)$: Page rank of V
- N_v : The number of forward links (out links) of page v
- C : a constant

Here the column vector is none other than the page rank vector which changes for every iteration. The page rank can be stopped once all the rank values stays constant after a particular iteration. Let us take another sample example shown in Figure 3.4.

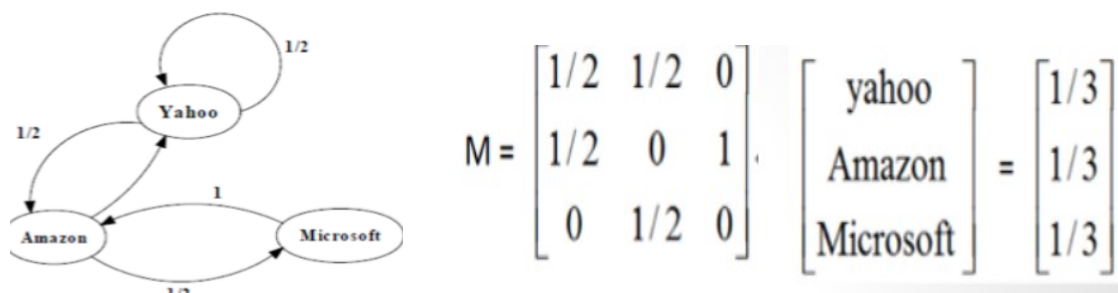


Figure 3.4 Simplified page rank example

In the given figure 3.4 matrix M is the notation of the graph where the values are represented in 1/(Number of out links of the edge). If there is no edge then it is represented as 0. Multiplying matrix M with the vector in the right will give the page Rank. After repeating the same process i.e., after some iteration's till the point of convergence.

$$\begin{bmatrix} 5/12 \\ 1/3 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix} \quad \begin{bmatrix} 5/12 \\ 1/3 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix}$$

(a)

(b)

$$\begin{bmatrix} 3/8 \\ 11/24 \\ 1/6 \end{bmatrix} \quad \begin{bmatrix} 5/12 \\ 17/48 \\ 11/48 \end{bmatrix} \quad \dots \quad \begin{bmatrix} 2/5 \\ 2/5 \\ 1/5 \end{bmatrix}$$

(c)

FIGURE 3.5 (A) PageRank after 1 iteration (B) PageRank after 2 iteration (C) PageRank after some iterations until convergence

The problem with the simplified page rank is that they fail to work during a certain case called Spider trap. During each iteration the loop accumulates rank but never distributes rank to other pages. There are a group of pages that all have out links but they never link to any other pages and because of this after certain iterations at convergence the values become 0 using which can't be ranked further. Another issue comes up with the Dangling node. A node is called a dangling node if it does not contain any out-going link, i.e., if the out-degree is zero. At this case also the value becomes zero after some iterations at convergence. There are two possible ways to overcome these issues.

- 1) Removing dead ends - Recursively drop dead ends from the graph along with their incoming arcs. Eventually, we will be having a strongly connected component with no dead ends.
- 2) Taxation – Overcoming the problem of dead end and spider traps. we add another term to the basic formula which consider the probability of moving from one page to another without following any link.

Considering the above points a modified page rank formula was introduced as shown in fig. 3.6

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

Figure 3.6: Improvised page rank Formula

Here d denotes damping factor which means the probability that the surfer quits the current page and teleports to a new one. The more the damping factor the more the time taken to calculate the PageRank value. Each page has 1/N probability to be chosen since every page

can be teleported. Hence using this formula in figure 3.1 Page rank can be calculated for every page. The Map-Reduce pseudo code for the Page rank is shown below in figure 3.7 using which top 10 webpages with best ranks is found.

```
map( key : [url, pagerank], value : outlink_list )
  for each outlink in outlink_list :
    emit( key : outlink, value : pagerank / sizeof(outlink_list) )
  emit( key : url, value : outlink_list )

reducer( key : url, value : list_pr_or_urls )
  outlink = [ ]
  pagerank = 0
  for each pr_or_urls in list_pr_or_urls :
    if is_list(pr_or_urls)
      outlink = pr_or_urls
    else
      pagerank += pr_or_urls
  pagerank = 1 - DAMPING_FACTOR + ( DAMPING_FACTOR * pagerank )
  emit( key : [url, pagerank], value : outlink_list )
```

Figure 3.7 Pseudo code for page rank using Map-Reduce

3.4.2 TOPIC SENSITIVE PAGE RANK

As mentioned back in the introduction part, Topic Sensitive page rank or personal or context sensitive page rank is based on queries that have same topics where different people have different interests. Users want only relevant webpages to what they search in such case topic sensitive page rank is used. Here pages that satisfy the given topic are given more preference which means those particular pages are biased and are given more priority. In such a way the users can get their desired output. This is an extension of pagerank and only a small modification is required. To implement this teleport is used. Here topics file act as the teleport set where we assume that the selected random pages are biased to specific topic. Topic sensitive page work based on the following formula in fig 3.8.

$$A_{ij} = \begin{cases} \beta M_{ij} + (1 - \beta)/|S| & \text{if } i \in S \\ \beta M_{ij} & \text{otherwise} \end{cases}$$

Figure 3.8: Topic sensitive page rank formula

S denotes the *teleport set* here which has the biased pages that help improvise their page rank. B denotes the damping factor again in which the pages are jumped randomly. So, based on the above formula it clearly states if the page belongs to the teleport set then an extra value is added giving it more biased hence resulting in that biased move to the top. The fig. 3.9 shows the pseudo code of topic sensitive page rank using MapReduce model. Hence with the help of these the ranking values can be obtained.

```

map( key : [url, pagerank], value : [outlink_list, teleport_list] )
  for each outlink in outlink_list :
    emit( key : outlink, value : pagerank / sizeof(outlink_list) )
  emit( key : url, value : [outlink_list, teleport_list] )

reducer( key : url, value : list_pr_or_urls )
  outlink = [ ]
  teleport = [ ]
  pagerank = 0
  for each pr_or_urls in list_pr_or_urls :
    if is_list(pr_or_urls)
      outlink = pr_or_urls[0]
      teleport = pr_or_urls[1]
    else
      pagerank += pr_or_urls
  if url ∈ teleport
    pagerank = 1 - DAMPING_FACTOR
  pagerank += DAMPING_FACTOR * pagerank
  emit( key : [url, pagerank], value : outlink_list )

```

Figure 3.9 Pseudo code for topic sensitive page rank using MapReduce

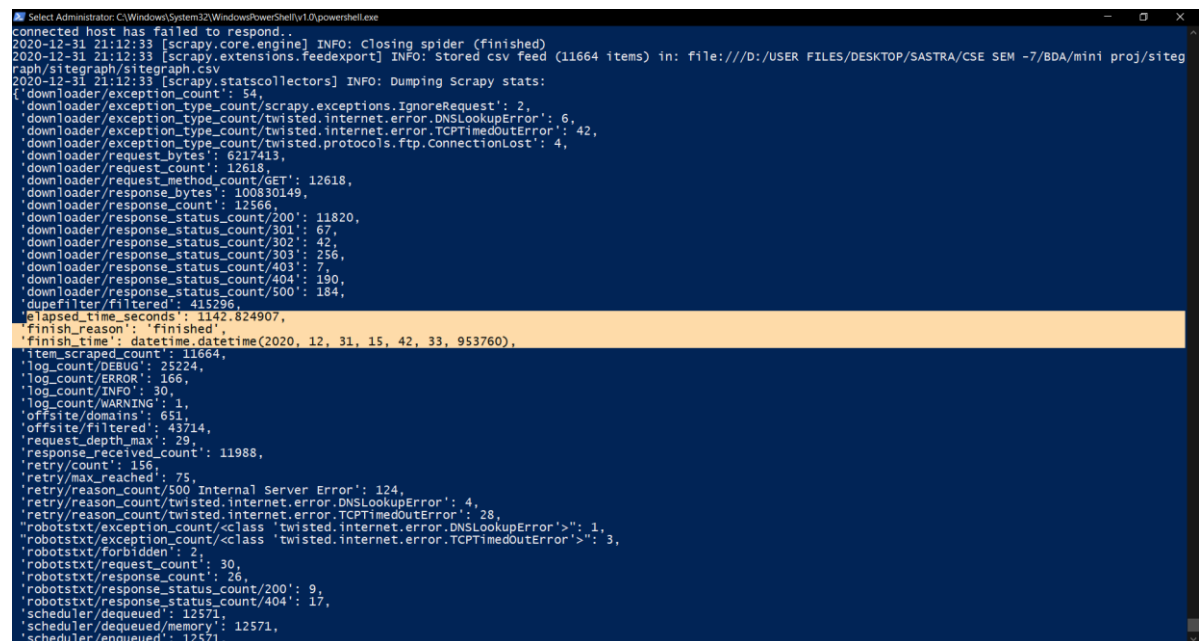
3.4.3 Map-Reduce

MapReduce is a framework for executing highly parallelizable and distributable algorithms across huge datasets using a large number of commodity computers. MapReduce model originates from the map and reduce combinators concept in functional programming languages. MapReduce is based on divide and conquer principles. The input datasets are split into independent chunks, which are processed by the mapper in parallel. Execution of the maps is typically co-located with the data. Then the framework sorts the outputs of the maps, and uses them as an input to the reducers.

CHAPTER 4

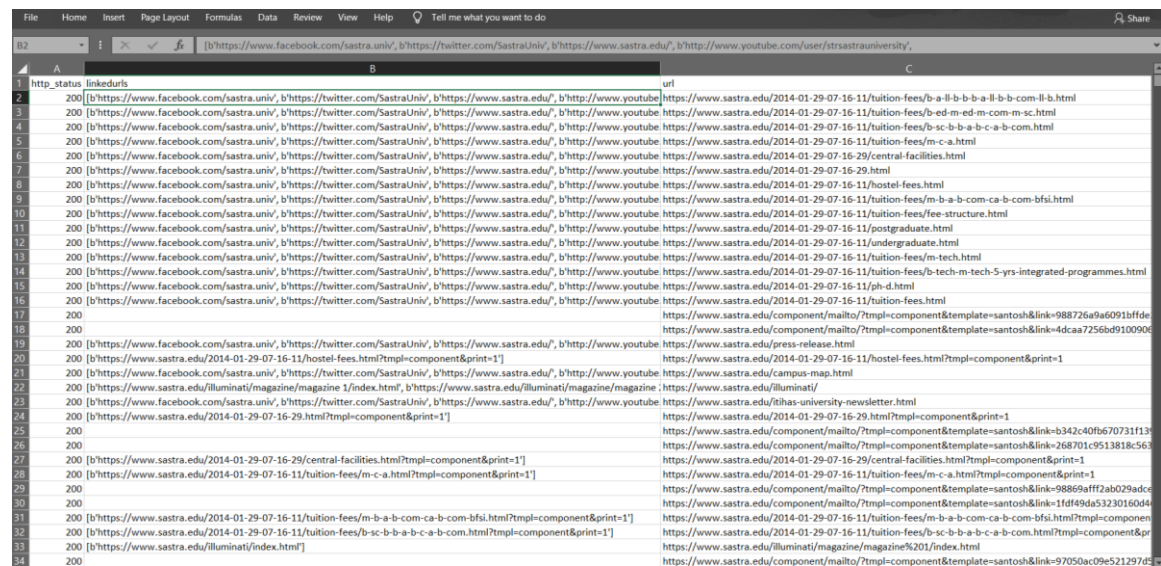
RESULTS

Various snapshot of images are obtained from different stages on implementation. The first two snapshots fig 4.1 and 4.2 deals with the output of the crawler that mapped the domain “sastra.edu”. Fig 4.1 shows the statistics by the *Scrapy* crawler after crawling the data. The highlighted part in it specifically shows the time taken to crawl the whole data which took exactly 1142.824 seconds (i.e., ~19.05 minutes). The image also shows the location of file stored and other statistics such as count of various HTTP status received in the power shell.



```
Select Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
connected host has failed to respond..
2020-12-31 21:12:33 [scrapy.core.engine] INFO: Closing spider (finished)
2020-12-31 21:12:33 [scrapy.extensions.feedexport] INFO: Stored csv feed (11664 items) in: file:///D:/USER FILES/DESKTOP/SASTRA/CSE SEM -7/BDA/mini proj/sitegraph/sitegraph/sitegraph.csv
2020-12-31 21:12:33 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/exception_count': 54,
 'downloader/exception_type_count/scrapy.exceptions.IgnoreRequest': 2,
 'downloader/exception_type_count/twisted.internet.error.DNSLookupError': 6,
 'downloader/exception_type_count/twisted.internet.error.TCPTimedOutError': 42,
 'downloader/exception_type_count/twisted.protocols.ftp.ConnectionLost': 4,
 'downloader/request_bytes': 6217413,
 'downloader/request_count': 12618,
 'downloader/request_method_count/GET': 12618,
 'downloader/response_bytes': 100830149,
 'downloader/response_count': 12566,
 'downloader/response_status_count/200': 11820,
 'downloader/response_status_count/301': 67,
 'downloader/response_status_count/302': 42,
 'downloader/response_status_count/303': 256,
 'downloader/response_status_count/403': 7,
 'downloader/response_status_count/404': 190,
 'downloader/response_status_count/500': 184,
 'dupefilter/filtered': 415296,
 'elapsed_time_seconds': 1142.824907,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2020, 12, 31, 15, 42, 33, 953760),
 'item_scraped_count': 11664,
 'log_count/DEBUG': 25224,
 'log_count/ERROR': 166,
 'log_count/INFO': 30,
 'log_count/WARNING': 1,
 'offsite/domains': 651,
 'offsite/filtered': 43714,
 'request_depth_max': 29,
 'response_received_count': 11988,
 'retry/count': 156,
 'retry/max_reached': 75,
 'retry/reason_count/500 Internal Server Error': 124,
 'retry/reason_count/twisted.internet.error.DNSLookupError': 4,
 'retry/reason_count/twisted.internet.error.TCPTimedOutError': 28,
 'robotstxt/exception_count/ciclass 'twisted.internet.error.DNSLookupError': 1,
 'robotstxt/exception_count/ciclass 'twisted.internet.error.TCPTimedOutError': 3,
 'robotstxt/Forbidden': 2,
 'robotstxt/request_count': 30,
 'robotstxt/response_count': 26,
 'robotstxt/response_status_count/200': 9,
 'robotstxt/response_status_count/404': 17,
 'scheduler/dequeued': 12571,
 'scheduler/dequeued/memory': 12571,
 'scheduler/enqueued': 12571,
```

Figure 4.1 Scrapy crawler output statistics



	A	B	C
1	http_status	linkedurls	url
2	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/b-a-b-b-a-b-b-com-b-b.html
3	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/b-ed-m-ed-m-com-m-sc.html
4	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/m-c-a-b-com.html
5	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/m-c-a.html
6	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/central-facilities.html
7	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/central-facilities.html
8	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/hostel-fees.html
9	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/m-b-a-b-com-ca-b-com-bfsi.html
10	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/fee-structure.html
11	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/postgraduate.html
12	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/undergraduate.html
13	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/m-tech.html
14	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/b-tech-m-tech-5-yr-integrated-programmes.html
15	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/ph-d.html
16	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees.html
17	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/component/mailto/?tmpl=component&template=santosh&link=988726a9a6091bf1fde
18	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/component/mailto/?tmpl=component&template=santosh&link=988726a9a6091bf1fde
19	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/press-release.html
20	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/hostel-fees.html?tmpl=component&print=1
21	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/campus-map.html
22	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/illuminati/magazine/1/index.html
23	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/illuminati/magazine/1/index.html
24	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/illuminati/magazine/1/index.html
25	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/illuminati/magazine/1/index.html
26	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/illuminati/magazine/1/index.html
27	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/central-facilities.html?tmpl=component&print=1
28	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/m-c-a.html?tmpl=component&print=1
29	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/m-c-a.html?tmpl=component&print=1
30	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/m-b-a-b-com-ca-b-com-bfsi.html?tmpl=component&print=1
31	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/m-b-a-b-com-ca-b-com-bfsi.html?tmpl=component&print=1
32	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/b-sc-b-a-b-a-b-com.html?tmpl=component&print=1
33	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/illuminati/magazine/magazine%201/index.html
34	200	b'https://www.facebook.com/sastra.univ', b'https://twitter.com/SastraUniv', b'https://www.sastra.edu', b'http://www.youtube.com/user/sastrauniversity'	https://www.sastra.edu/component/mailto/?tmpl=component&template=santosh&link=97050ac09e521297d3

Figure 4.2 Output of scrapy dataset stored in Microsoft excel

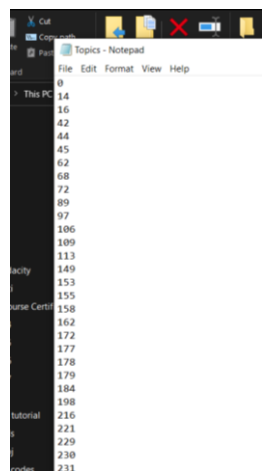
It is evident from the output in fig. 4.2 that the pages are mapped in one-to-many format from cell C to cell B. It can also be seen that many cells in column B are empty. These will be pre-processed in the next step to form edge list and other required files. The file size of the output from scrapy was 35 mb.

	A	B
1	Source	Target
2	0	0
3	0	1
4	0	2
5	0	3
6	0	4
7	0	5
8	0	6
9	0	7
10	0	8
11	0	9
12	0	10
13	0	11
14	0	12
15	0	13
16	0	14
17	0	15
18	0	16
19	0	17
20	0	18

(a)

	A	B
1	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/b-a-ll-b-b-a-ll-b-b-com-ll-b.html	0
2	https://www.facebook.com/sastra.univ	1
3	https://twitter.com/SastraUniv	2
4	https://www.sastra.edu/	3
5	http://www.youtube.com/user/strsastrauniversity	4
6	https://accounts.google.com/ServiceLogin?service=mail&continue=https://mail.google.com/mail	5
7	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/b-a-ll-b-b-a-ll-b-b-com-ll-b.html#s	6
8	https://www.sastra.edu/2014-01-29-07-16-11/tuition-fees/b-a-ll-b-b-a-ll-b-b-com-ll-b.html#s	7
9	http://sptc.sastra.edu	8
10	http://src.sastra.edu	9
11	https://www.sastra.edu/2014-01-29-07-13-19/web-archives.html	10
12	https://www.sastra.edu/2014-01-29-07-15-32.html	11
13	https://www.sastra.edu/2014-01-29-07-15-32/sastra.html	12
14	https://www.sastra.edu/2014-01-29-07-15-32/2014-01-29-11-53-32.html	13
15	http://sptc.sastra.edu/	14
16	https://www.sastra.edu/2014-01-29-07-15-32/mission-vision.html	15
17	https://www.sastra.edu/2014-01-29-07-15-32/message-from-chancellor.html	16
18	https://www.sastra.edu/2014-01-29-07-15-32/guiding-model.html	17
19	https://www.sastra.edu/2014-01-29-07-15-32/accreditation-ranking.html	18
20	https://www.sastra.edu/2014-01-29-07-15-32/mhrd-ugc-compliance.html	19

(b)



(c)

```
PS D:\USER FILES\DESKTOP\SASTRA\CSE SEM -7\BDA\mini proj\sitgraph> d; cd "d:\USER FILES\DESKTOP\SASTRA\CSE SEM -7\BDA\mini proj\sitgraph"; python -2020.12.424452561\pythonFiles\lib\python\debugpy\launcher "56980" "--" "d:\USER FILES\DESKTOP\SASTRA\CSE SEM -7\BDA\mini proj\sitgraph\converter.py"
No of Nodes = 27599
No of Edges = 509457
Files created Successfully!!!
PS D:\USER FILES\DESKTOP\SASTRA\CSE SEM -7\BDA\mini proj\sitgraph>
```

(d)

Figure 4.3 The snapshots shows the outputs of various files produced by converter.py that preprocessed the crawled data. (a) The data contains the edge list and are converted to unique integer numbers (b) This data contains node information where each website will be having unique integer value (c) This file contains teleport set data that are randomly generated and are assumed belonging to certain topic (d) Output generated by the python file showing total number of nodes and total number of edges.

There are total of 27599 nodes and more than 5 lakh edges in the network graph. The file size of node data was 2.48 MB, edgelist was 5.34 MB and Topics file was just 17.7 KB.

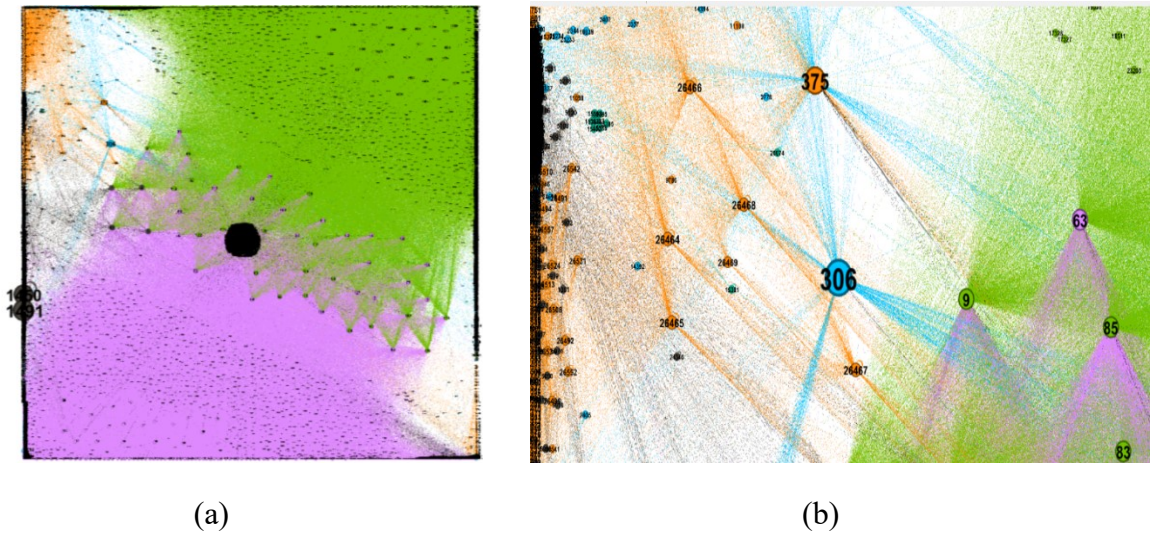


Figure 4.4 (a) The directed Web graph consisting of 27599 nodes. Nodes with high page rank value are shown bigger (b) A zoomed image of the web graph from the first graph. Here we can clearly see the edges.

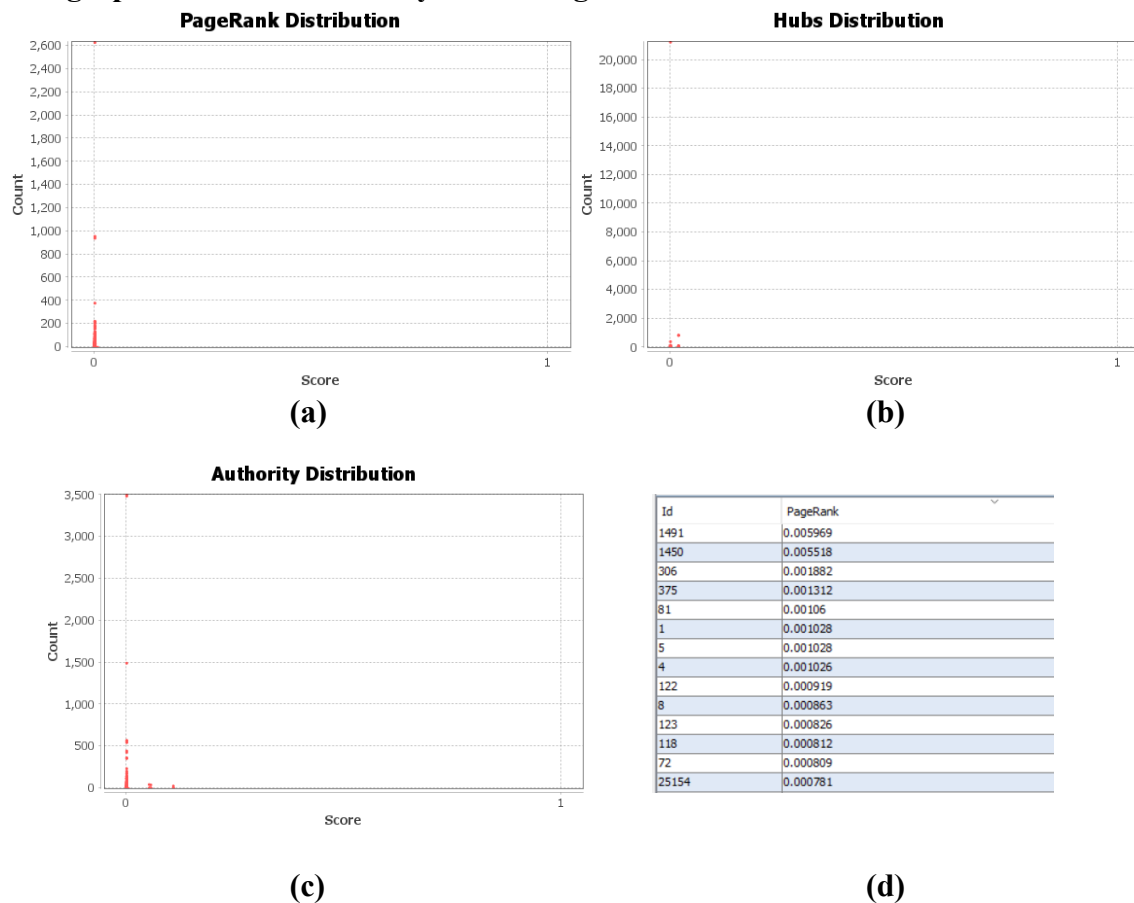


Figure 4.5 The distribution of nodes for (a) PageRank algorithm (b) & (c) HITS algorithm that was calculated with the graph data automatically in Gephi tool. (d) Shows the top pages with maximum page rank values.

```

C:\Windows\system32\cmd.exe

(base) C:\spark-3.0.1-bin-hadoop2.7\bin>spark-submit ./pagerank.py local[1] ../crawlerdata\mylist.txt 10
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/spark-3.0.1-bin-hadoop2.7/jars/spark-unsafe_2.12-3.0.1.jar) to
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
local[1]

*****

Top 10 pages and ranks

45.665405013923774 ---rank for link--- http://sastra.edu.
45.634119841403496 ---rank for link--- https://www.sastra.edu/staffprofiles/index.php.
12.450703532553145 ---rank for link--- http://www.sastra.edu.
9.879186326097159 ---rank for link--- http://www.sastra.edu/.
8.323568055361331 ---rank for link--- http://webstream.sastra.edu/sastrapw/.
8.075516305069973 ---rank for link--- https://accounts.google.com/ServiceLogin?service=mail&continue=https://mail.google.com/mail/.
8.075516305069973 ---rank for link--- https://www.facebook.com/sastra.univ.
8.066931550365377 ---rank for link--- http://www.youtube.com/user/strsastrauniversity.
7.281018253847204 ---rank for link--- http://mail.sastra.edu.
7.277491282892847 ---rank for link--- http://sptc.sastra.edu.
6.508282218153357 ---rank for link--- http://mail.sastra.ac.in.

*****

(base) C:\spark-3.0.1-bin-hadoop2.7\bin>

```

Figure 4.6 The top 11 pages based on page ranking algorithm run with the help of spark.

Fig. 4.6 shows the top 11 webpages ranked under page ranking algorithm using MapReduce. Total of 10 iterations was run on the local system with single thread in spark system. It used edge list data to generate this result.

```

(base) C:\spark-3.0.1-bin-hadoop2.7\bin>spark-submit ./topic_sensitive.py local[1] ../crawlerdata\mylist.txt 10 ../crawlerdata\Topics.txt
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/spark-3.0.1-bin-hadoop2.7/jars/spark-unsafe_2.12-3.0.1.jar) to
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release

*****

Top 10 pages and ranks

6.321732179673188 ---rank for link--- http://sastra.edu.
6.16551116225586 ---rank for link--- https://www.sastra.edu/staffprofiles/index.php.
1.9301342719114862 ---rank for link--- https://www.sastra.edu/core/may09/index.htm.
1.8879764514843695 ---rank for link--- https://www.sastra.edu/core/nses06/sessionIV/p2.htm.
1.6265435347156567 ---rank for link--- http://www.sastra.edu.
1.5428478270077794 ---rank for link--- https://www.sastra.edu/core/may11/index.htm.
1.5037456138259893 ---rank for link--- https://www.sastra.edu/core/nses06/sessionIII/p2.htm.
1.4893237279491893 ---rank for link--- https://www.sastra.edu/core/nses06/sessionIV/p1.htm.
1.445407169446866 ---rank for link--- https://www.sastra.edu/core/nses06/sessionIV/p2.htm.
1.444153668359664 ---rank for link--- https://www.sastra.edu/core/nses06/sessionV/p2.htm.

*****

(base) C:\spark-3.0.1-bin-hadoop2.7\bin>

```

Figure 4.7 The top 11 pages based on Topic sensitive page rank run with the help of spark

Fig. 4.7 shows values based on Topic sensitive page rank algorithm using MapReduce. From the initiating command we can infer that a total of 10 iterations was run with the help of edge list file and topics (Teleport Set) file. We can also see different set of top pages compared to that in Fig 4.6. One of the nodes listed in Teleport set had also made it to the top 10 pages.

CHAPTER 5

CONCLUSION AND FURTHER RESEARCH

In conclusion, the implementation of the Map-Reduce model for the Page Rank and Topic Sensitive Page Rank with the help of Spark is achieved. As an add on using Gephi tool HITS algorithm is also calculated. It is also to be noted that order of nodes/webpages obtained in Figure 4.5(d) and 4.6 perfectly matches even though it uses different implementation.

For further future works, the crawler can be more improvised to find absolute links and to avoid errors such as www.sastra.edu and sastra.edu in which both points to the same page. Such redundancies can be avoided. An efficiency comparison can also be plotted to compare the speed between Map-Reduce implementation and normal (iterative) implementation of the algorithm. Also, that the crawler can be further developed to crawl and extract files specifically for teleport set for specific set of topics.

In light of the instinct of subject topic sensitive page ranking a more modest change to the underlying page rank is considered. Viable computation of page rank is performed with Map-Reduce in pseudo distributed mode in Windows System. The Map-Reduce topic sensitive page ranking is proposed and henceforth implemented for the scraped information. The equation can be additionally altered for better precision and execution.