Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai
Artificial Intelligence and Data Science Department

Big Data Analytics/Odd Sem 2023-23/Experiment 7

| **Name :** Shreya Singh | **Class/Roll No. :** D16AD/55 | **Grade :** |
|---|---|---|

**Title of Experiment :** PySpark

**Objective of Experiment :**
To teach the fundamental techniques and principles in achieving big data analytics with scalability and streaming capability. To enable students to have skills that will help them to solve complex real-world problems in decision support.

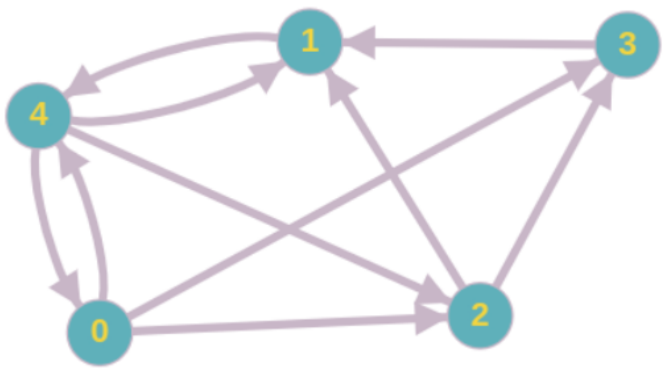
**Outcome of Experiment :**
Collect, manage, store, query and analyze various forms of Big Data. Interpret business models and scientific computing paradigms, and apply software tools for big data analytics.

**Problem Statement :**
To study and implement a Page Rank algorithm using PySpark.

**Theory:**

The PageRank algorithm or Google algorithm was introduced by Larry Page, one of the founders of Google. It was first used to rank web pages in the Google search engine. Nowadays, it is more and more used in many different fields, for example in ranking users in social media etc… What is fascinating with the PageRank algorithm is how to start from a complex problem and end up with a very simple solution. You just need to have some basics in algebra and Markov Chains. Here, we will use ranking web pages as a use case to illustrate the PageRank algorithm.

The web can be represented like a directed graph where nodes represent the web pages and edges form links between them. Typically, if a node (web page) i is linked to a node j, it means that i refers to j.




We have to define what is the importance of a web page. As a first approach, we could say that it is the total number of web pages that refer to it. If we stop to this criteria, the importance of these web pages that refer to it is not taken into account. In other words, an important web page and a less important one has the same weight. Another approach is to assume that a web page spreads its importance equally to all web pages it links to. By doing that, we can then define the score of a node j as follows:

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

where $r_i$ is the score of the node i and $d_i$ its out-degree.

PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references.

The numerical weight that it assigns to any given element E is referred to as the PageRank of E and denoted by PR(E).

A PageRank results from a mathematical algorithm based on the webgraph, created by all World Wide Web pages as nodes and hyperlinks as edges, taking into consideration authority hubs such as cnn.com or mayoclinic.org. The rank value indicates the importance of a particular page. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it ('incoming links'). A page that is linked to by many pages with high PageRank receives a high rank itself.Numerous academic papers concerning PageRank have been published since Page and Brin's original paper.
In practice, the PageRank concept may be vulnerable to manipulation. Research has been conducted into identifying falsely influenced PageRank rankings. The goal is to find an effective means of ignoring links from documents with falsely influenced PageRank.

Algorithm:

The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called "iterations", through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.
A probability is expressed as a numeric value between 0 and 1. A 0.5 probability is commonly expressed as "50% chance" of something happening. Hence, a document with a PageRank of 0.5 means there is a 50% chance that a person clicking on a random link will be directed to said document.

Dataset used:
Kaggle dataset link: https://www.kaggle.com/pappukrjha/google-web-graph

```
# FromNodeId    ToNodeId
0       11342
0       824020
0       867923
0       891835
11342   0
11342   27469
11342   38716
11342   309564
11342   322178
11342   387543
11342   427436
11342   538214
11342   638706
11342   645018
11342   835220
11342   856657
11342   867923
11342   891835
824020  0
824020  91807
824020  322178
824020  387543
824020  417728
824020  438493
824020  500627
824020  535748
824020  695578
824020  867923
824020  891835
867923  0
867923  11342
```

## Output Screenshots :

Move the data into hadoop file system:

```
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Desktop/web-Google.txt .
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 3 items
drwxr-xr-x   - cloudera cloudera          0 2023-10-07 07:18 Desktop
drwxr-xr-x   - cloudera cloudera          0 2023-10-07 07:19 PageRank
-rw-r--r--   1 cloudera cloudera   75380115 2023-10-07 08:02 web-Google.txt
[cloudera@quickstart ~]$
```

Start Pyspark , Compute Contrib function , RDD named links:

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Python version 2.6.6 (r266:84292, Jul 23 2015 15:22:56)
SparkContext available as sc, HiveContext available as sqlContext.
>>> def computeContribs(neighbors,rank):
...     for neighbor in neighbors:
...             yield(neighbor,rank/len(neighbors))
...
>>> links = sc.textFile('web-Google.txt')\
...     .map(lambda line: line.split())\
...     .map(lambda pages: (pages[0],pages[1]))\
...     .distinct()\
...     .groupByKey()\
...     .persist()
23/10/07 09:47:48 WARN shortcircuit.DomainSocketFactory: The short-circuit local
 reads feature cannot be used because libhadoop cannot be loaded.
>>>
```

Ranks RDD storing the ranks data , Loop in order to calculate contribs and ranks:

```
>>> ranks = links.map(lambda (page,neighbors):(page,1.0))
>>> for x in xrange(10):
...     contribs=links\
...     .join(ranks)\
...     .flatMap(lambda (page,(neighbors,rank)):computeContribs(neighbors,rank))
...     ranks = contribs\
...     .reduceByKey(lambda v1,v2: v1+v2)\
...     .map(lambda (page,contrib):(page,contrib * 0.85 + 0.15))
...
>>>
```

Collect all ranks:

```
>>> for rank in ranks.collect():
...     print rank
...
```

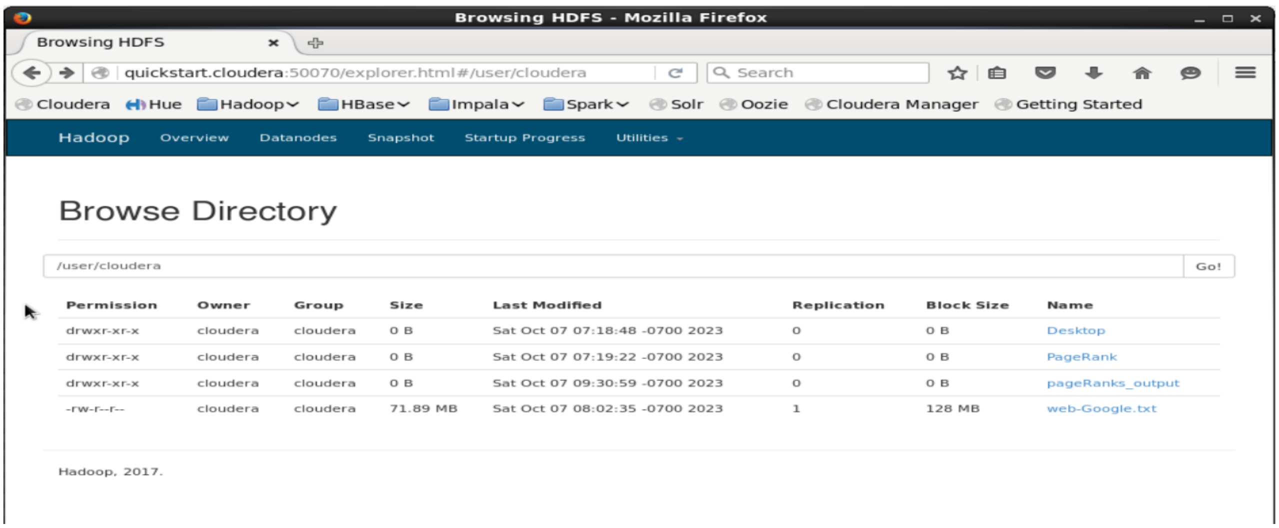
Output:

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
(u'149656', 0.35906672327214939)
(u'464223', 0.33927151061586563)
(u'466414', 1.0656390152080852)
(u'3167', 1.6474226393641223)
(u'714355', 0.15944526747562174)
(u'132360', 4.3554611258711775)
(u'530649', 0.35447850030282335)
(u'756526', 0.26509633768545837)
(u'457641', 0.28889405356367082)
(u'675726', 0.33862063292960587)
(u'222333', 0.22508766688286641)
(u'883451', 0.40845178733478638)
(u'81005', 0.37860461572393816)
(u'67126', 0.19390151353387275)
(u'546376', 0.19336086517533133)
(u'740385', 0.40048043276156842)
(u'375509', 0.21550772698786863)
(u'608111', 0.75946629874746363)
(u'795945', 0.16766361393797 13)
(u'675898', 0.78453154123309721)
(u'761359', 0.33508330211175 98)
(u'606845', 0.68922336438702247)
(u'374452', 0.2008977085145 03)
(u'81108', 0.51207866113690448)
(u'137239', 0.21567502729108273)
(u'793392', 0.30893984632749 76)
(u'560031', 0.23849945066588274)
(u'495526', 0.79419764701731 71)
(u'713080', 0.19502942557444561)
(u'301874', 0.22732535856752234)
(u'722925', 0.32768168484677562)
(u'52302', 0.90860477474829149)
(u'321384', 0.29811004234831201)
(u'359462', 0.68947688407187224)
(u'641747', 0.22639973548712683)
(u'866700', 0.38601235112657939)
(u'326031', 0.27340506598853426)
(u'552009', 0.34063089230697891)
(u'366455', 0.27430888049837682)
cloudera@quickstart:~    Browsing HDFS - Mozil...
```

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
(u'465424', 0.25371064727868681)
(u'533846', 0.73499594380152833)
(u'178845', 0.82440207103463126)
(u'40574', 0.36893577023839097)
(u'688401', 0.15138612284601208)
(u'219540', 0.52086559746269279)
(u'862612', 0.1682011789239215)
(u'225840', 0.22112641401014263)
(u'578427', 1.9556828100899866)
(u'778461', 0.16614734675556039)
(u'772689', 0.25596658934028033)
(u'640067', 0.16888128780597281)
(u'494840', 0.16121393336472756)
(u'469947', 0.17602449281860807)
(u'557956', 0.15761193051022804)
(u'751932', 0.25512148001903612)
(u'439276', 0.37354030602742794)
(u'577774', 0.59186709809304527)
(u'13820', 0.46509307311203074)
(u'465394', 0.39391337311985208)
(u'103094', 0.20685839318121693)
(u'526431', 0.15166997681471525)
(u'242883', 0.39889751369909454)
(u'384934', 0.81738301227365884)
(u'5631', 0.63490126285120863)
(u'634794', 0.23873780776863762)
(u'225664', 1.6756683479047332)
(u'257481', 0.28509560395135675)
(u'245352', 0.17493958820692246)
(u'65395', 0.19594183330052109)
(u'686962', 1.4770699228362925)
(u'838440', 0.21160243776847804)
```

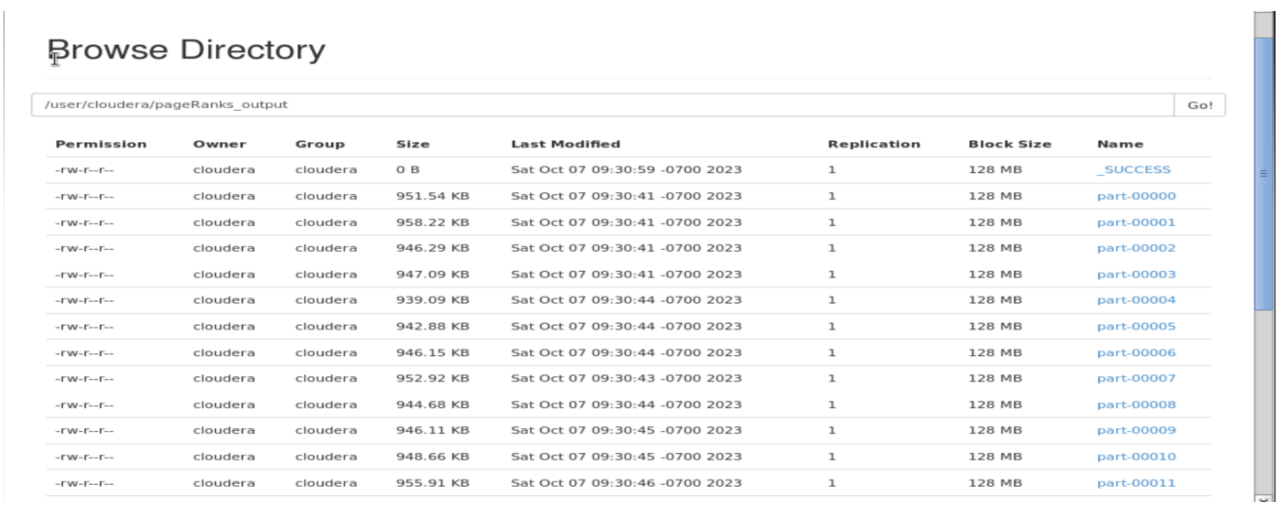
Saving the RDD as text file:

```
>>> ranks.count()
654830
>>> ranks.take(5)
[(u'681601', 0.20267241923465629), (u'880578', 0.9407418091371379), (u'89370', 0.385408584981935331), (u'460068', 0.4239354087
691638), (u'684237', 0.23593743836588738)]
>>> ranks.saveAsTextFile('pageRanks_output')
>>> 
```

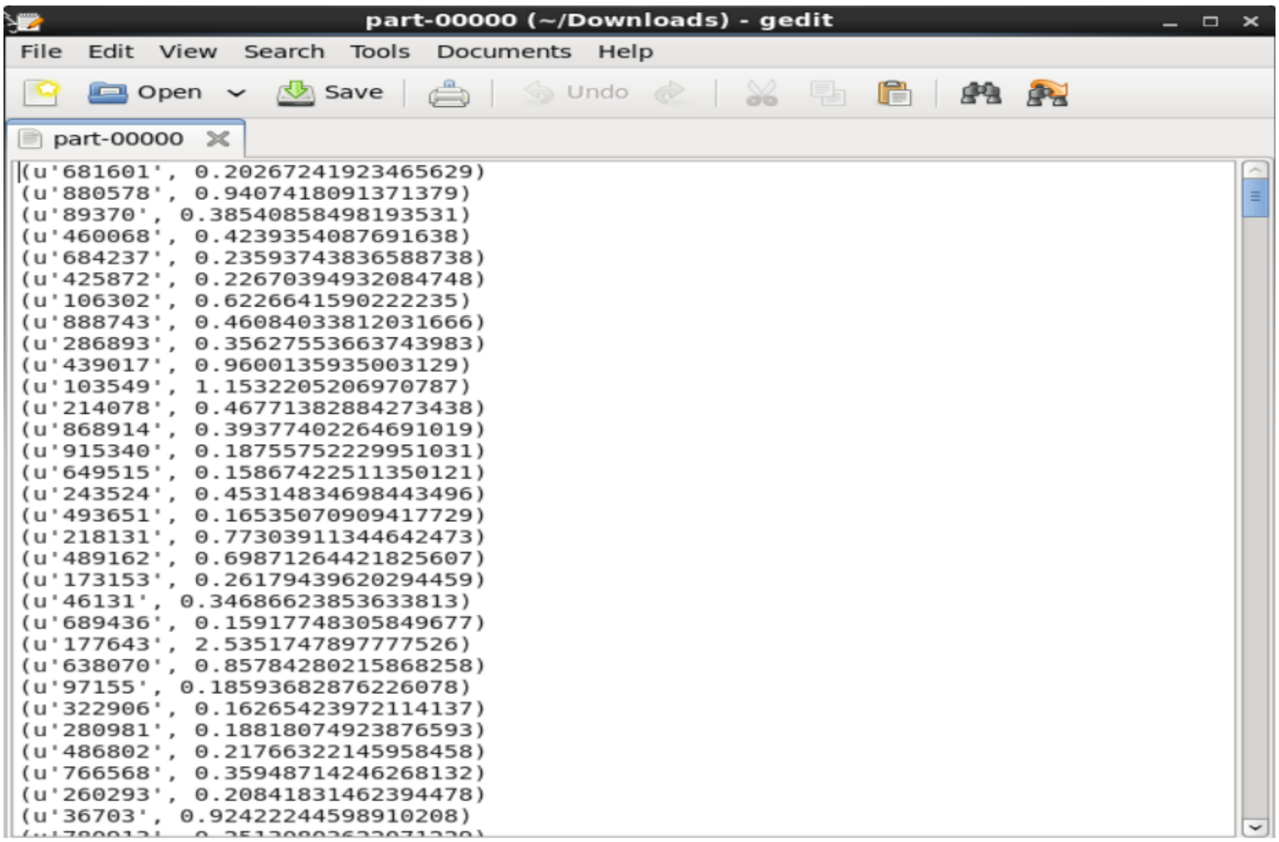



| Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name |
|---|---|---|---|---|---|---|---|
| drwxr-xr-x | cloudera | cloudera | 0 B | Sat Oct 07 07:18:48 -0700 2023 | 0 | 0 B | Desktop |
| drwxr-xr-x | cloudera | cloudera | 0 B | Sat Oct 07 07:19:22 -0700 2023 | 0 | 0 B | PageRank |
| drwxr-xr-x | cloudera | cloudera | 0 B | Sat Oct 07 09:30:59 -0700 2023 | 0 | 0 B | pageRanks_output |
| -rw-r--r-- | cloudera | cloudera | 71.89 MB | Sat Oct 07 08:02:35 -0700 2023 | 1 | 128 MB | web-Google.txt |

Hadoop, 2017.




| Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name |
|---|---|---|---|---|---|---|---|
| -rw-r--r-- | cloudera | cloudera | 0 B | Sat Oct 07 09:30:59 -0700 2023 | 1 | 128 MB | _SUCCESS |
| -rw-r--r-- | cloudera | cloudera | 951.54 KB | Sat Oct 07 09:30:41 -0700 2023 | 1 | 128 MB | part-00000 |
| -rw-r--r-- | cloudera | cloudera | 958.22 KB | Sat Oct 07 09:30:41 -0700 2023 | 1 | 128 MB | part-00001 |
| -rw-r--r-- | cloudera | cloudera | 946.29 KB | Sat Oct 07 09:30:41 -0700 2023 | 1 | 128 MB | part-00002 |
| -rw-r--r-- | cloudera | cloudera | 947.09 KB | Sat Oct 07 09:30:41 -0700 2023 | 1 | 128 MB | part-00003 |
| -rw-r--r-- | cloudera | cloudera | 939.09 KB | Sat Oct 07 09:30:44 -0700 2023 | 1 | 128 MB | part-00004 |
| -rw-r--r-- | cloudera | cloudera | 942.88 KB | Sat Oct 07 09:30:44 -0700 2023 | 1 | 128 MB | part-00005 |
| -rw-r--r-- | cloudera | cloudera | 946.15 KB | Sat Oct 07 09:30:44 -0700 2023 | 1 | 128 MB | part-00006 |
| -rw-r--r-- | cloudera | cloudera | 952.92 KB | Sat Oct 07 09:30:43 -0700 2023 | 1 | 128 MB | part-00007 |
| -rw-r--r-- | cloudera | cloudera | 944.68 KB | Sat Oct 07 09:30:44 -0700 2023 | 1 | 128 MB | part-00008 |
| -rw-r--r-- | cloudera | cloudera | 946.11 KB | Sat Oct 07 09:30:45 -0700 2023 | 1 | 128 MB | part-00009 |
| -rw-r--r-- | cloudera | cloudera | 948.66 KB | Sat Oct 07 09:30:45 -0700 2023 | 1 | 128 MB | part-00010 |
| -rw-r--r-- | cloudera | cloudera | 955.91 KB | Sat Oct 07 09:30:46 -0700 2023 | 1 | 128 MB | part-00011 |



part-00000 (~/Downloads) - gedit

File Edit View Search Tools Documents Help

Open Save Undo

part-00000

```
(u'681601', 0.20267241923465629)
(u'880578', 0.9407418091371379)
(u'89370', 0.38540858498193531)
(u'460068', 0.4239354087691638)
(u'684237', 0.23593743836588738)
(u'425872', 0.22670394932084748)
(u'106302', 0.6226641590222235)
(u'888743', 0.46084033812031666)
(u'286893', 0.35627553663743983)
(u'439017', 0.9600135935003129)
(u'103549', 1.1532205206970787)
(u'214078', 0.46771382884273438)
(u'868914', 0.39377402264691019)
(u'915340', 0.18755752229951031)
(u'649515', 0.15867422511350121)
(u'243524', 0.45314834698443496)
(u'493651', 0.16535070909417729)
(u'218131', 0.77303911344642473)
(u'489162', 0.69871264421825607)
(u'173153', 0.26179439620294459)
(u'46131', 0.34686623853633813)
(u'689436', 0.15917748305849677)
(u'177643', 2.5351747897777526)
(u'638070', 0.85784280215868258)
(u'97155', 0.18593682876226078)
(u'322906', 0.16265423972114137)
(u'280981', 0.18818074923876593)
(u'486802', 0.21766322145958458)
(u'766568', 0.35948714246268132)
(u'260293', 0.20841831462394478)
(u'36703', 0.92422244598910208)
```

**Results and Discussions :**

The implementation of the Page Rank algorithm using PySpark yielded insightful results. By applying this algorithm, we were able to assign ranking scores to web pages in a network, revealing the importance and influence of each page within the web graph. The algorithm's iterative nature and distributed computing capabilities of PySpark allowed us to efficiently handle large-scale datasets, making it a valuable tool for web search and recommendation systems. These results provide a foundation for understanding and improving web page ranking, enhancing the relevance and quality of search engine results.