Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department

BDA/Odd Sem 202323/Experiment 7

Name : Mayur Pimpude	Class/Roll No. :D16AD/43	Grade :

Title of Experiment :
To study and implement a Page Rank algorithm using PySpark.

Objective of Experiment :
To implement the K-Means clustering algorithm in PySpark for the purpose of grouping data points into clusters, facilitating data segmentation, and gaining a deeper understanding of the inherent patterns and relationships within the dataset.

Outcome of Experiment :
Implementation of the K-Means clustering algorithm in PySpark, resulting in the formation of distinct clusters within the dataset and providing insights into the underlying patterns and structures in the data

Theory :

What is Page Rank?
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:

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. 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 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

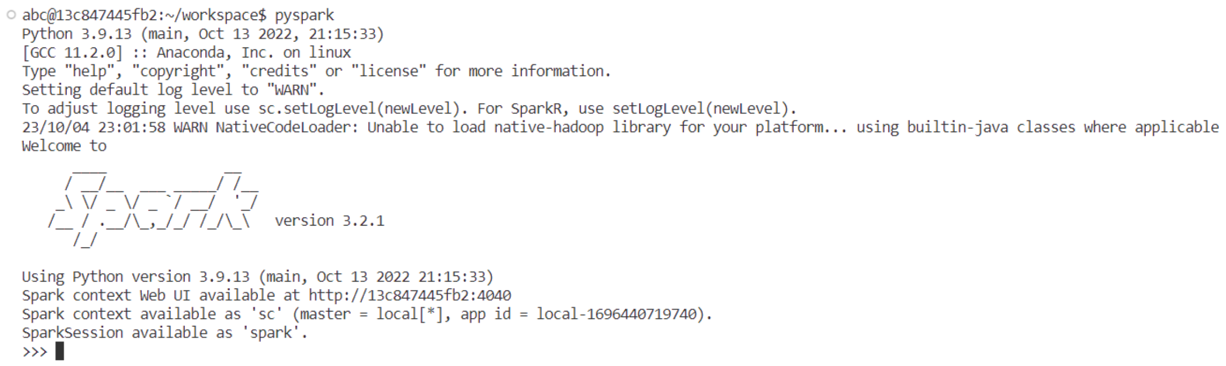
that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations
require several passes, called iterations through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.
A probability is expressed as a numeric value between 0 and 1. A 0.5 probability is commonly expressed as a 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.

Data set used:

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

Program :

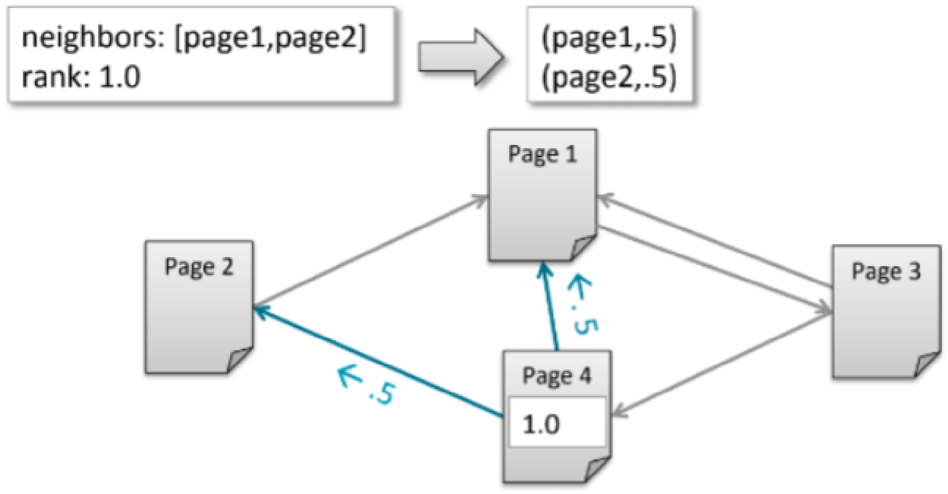
Execution:



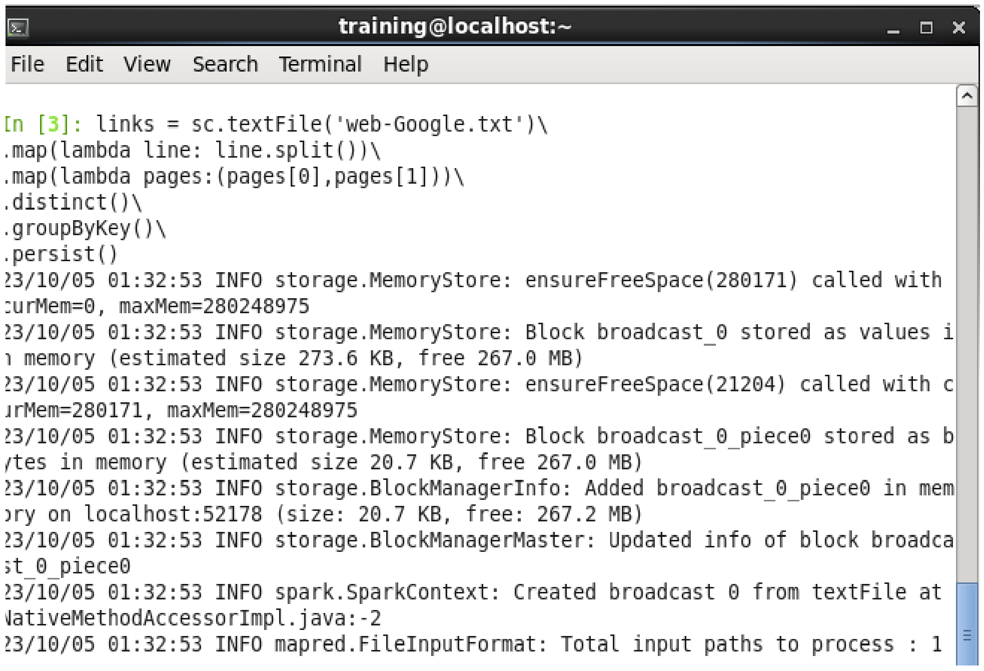
```
abc@13c847445fb2:~/workspace$ pyspark
Python 3.9.13 (main, Oct 13 2022, 21:15:33)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/10/04 23:01:58 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2.1
      /_/

Using Python version 3.9.13 (main, Oct 13 2022 21:15:33)
Spark context Web UI available at http://13c847445fb2:4040
Spark context available as 'sc' (master = local[*], app id = local-1696440719740).
SparkSession available as 'spark'.
>>> 
```

Step 1 -Writing compute contrib function:

```
>>> def computeContribs(neighbors,rank):
...       for neighbor in neighbors: yeild(neighbor, rank/len(neighbors))
...
```



Step 2 - Create a RDD named links with following command

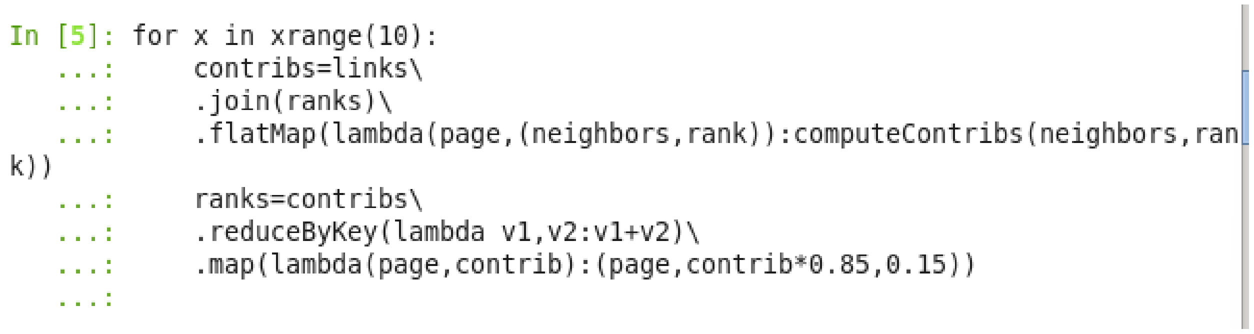


```
[n [3]: links = sc.textFile('web-Google.txt')\
.map(lambda line: line.split())\
.map(lambda pages:(pages[0],pages[1]))\
.distinct()\
.groupByKey()\
.persist()
23/10/05 01:32:53 INFO storage.MemoryStore: ensureFreeSpace(280171) called with
curMem=0, maxMem=280248975
23/10/05 01:32:53 INFO storage.MemoryStore: Block broadcast_0 stored as values i
n memory (estimated size 273.6 KB, free 267.0 MB)
23/10/05 01:32:53 INFO storage.MemoryStore: ensureFreeSpace(21204) called with c
urMem=280171, maxMem=280248975
23/10/05 01:32:53 INFO storage.MemoryStore: Block broadcast_0_piece0 stored as b
ytes in memory (estimated size 20.7 KB, free 267.0 MB)
23/10/05 01:32:53 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in mem
ory on localhost:52178 (size: 20.7 KB, free: 267.2 MB)
23/10/05 01:32:53 INFO storage.BlockManagerMaster: Updated info of block broadca
st_0_piece0
23/10/05 01:32:53 INFO spark.SparkContext: Created broadcast 0 from textFile at
NativeMethodAccessorImpl.java:-2
23/10/05 01:32:53 INFO mapred.FileInputFormat: Total input paths to process : 1
```

5. Create a ranks rdd storing the ranks data

```
In [4]: ranks=links.map(lambda(page,neighbors):(page,1.0))
```

6. Create a loop in order to calculate contribs and ranks

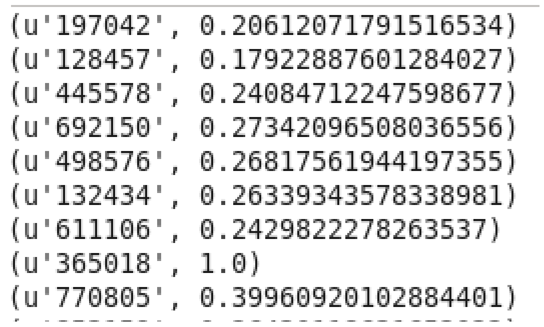


```
In [5]: for x in xrange(10):
   ...:     contribs=links\
   ...:     .join(ranks)\
   ...:     .flatMap(lambda(page,(neighbors,rank)):computeContribs(neighbors,ran
k))
   ...:     ranks=contribs\
   ...:     .reduceByKey(lambda v1,v2:v1+v2)\
   ...:     .map(lambda(page,contrib):(page,contrib*0.85,0.15))
   ...:
```

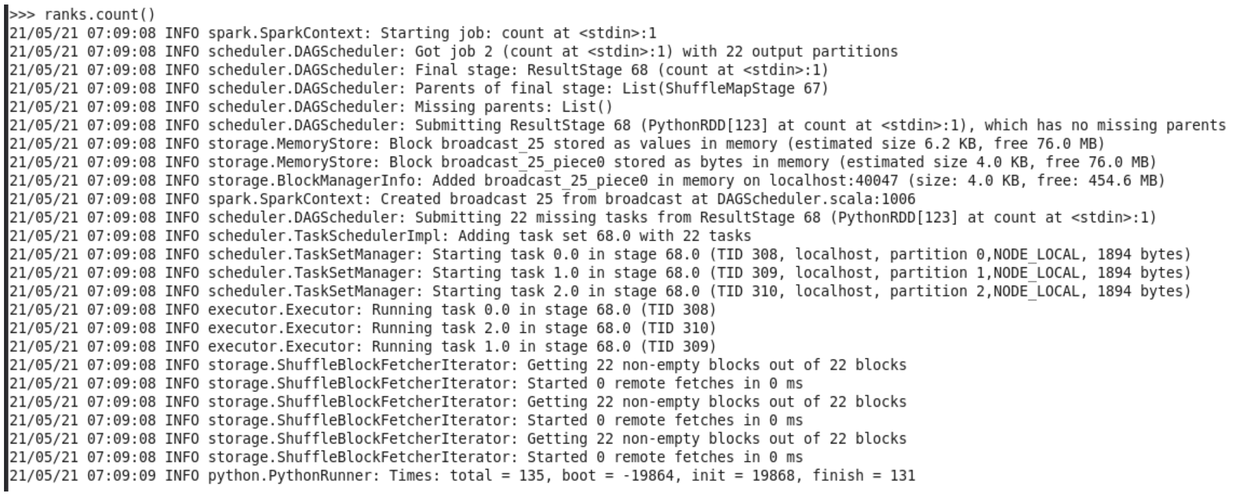
7. Code to collect all ranks

```
In [6]: for rank in ranks.collect(): print rank
```

8. Output

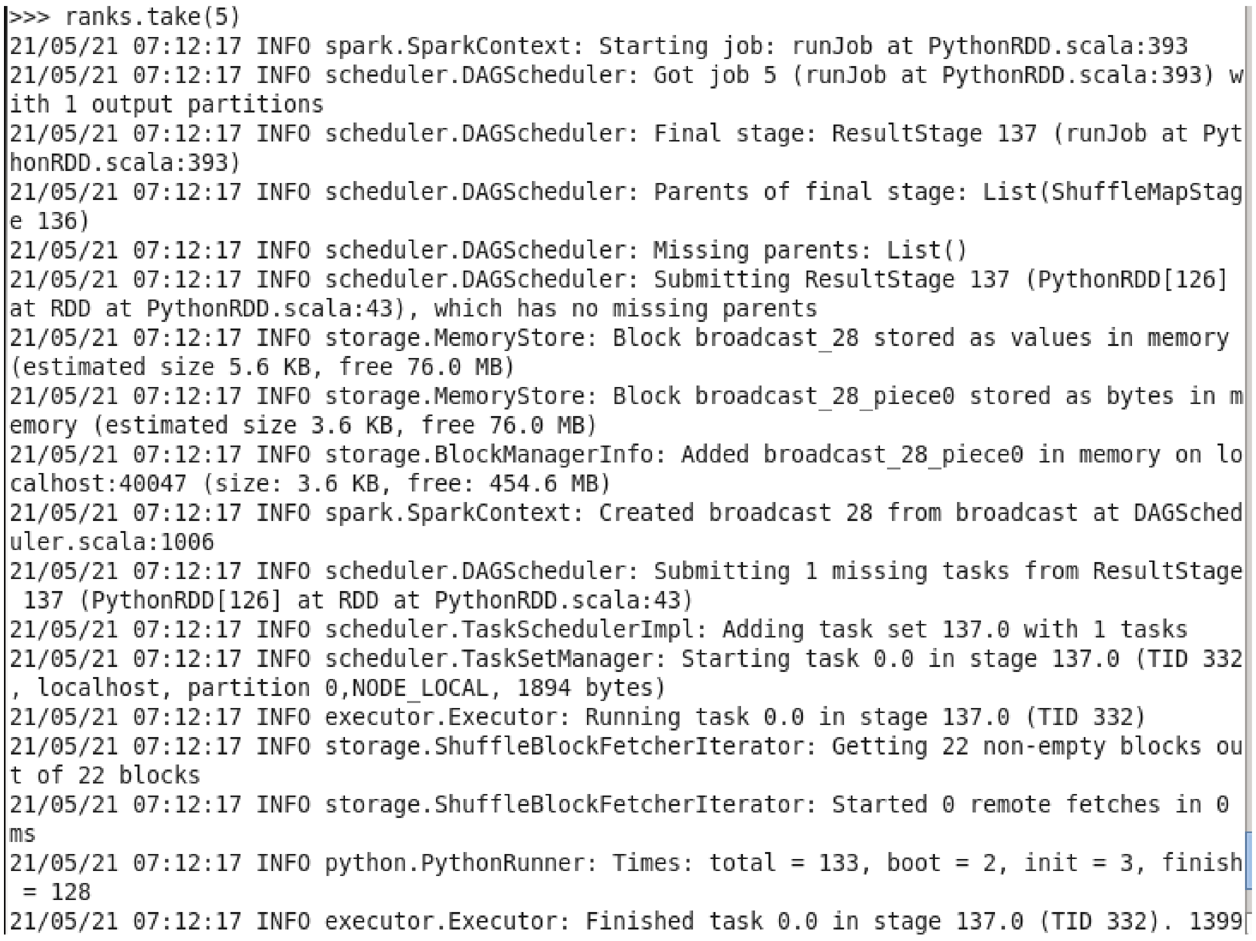


```
(u'197042', 0.20612071791516534)
(u'128457', 0.17922887601284027)
(u'445578', 0.24084712247598677)
(u'692150', 0.27342096508036556)
(u'498576', 0.26817561944197355)
(u'132434', 0.26339343578338981)
(u'611106', 0.2429822278263537)
(u'365018', 1.0)
(u'770805', 0.39960920102884401)
```

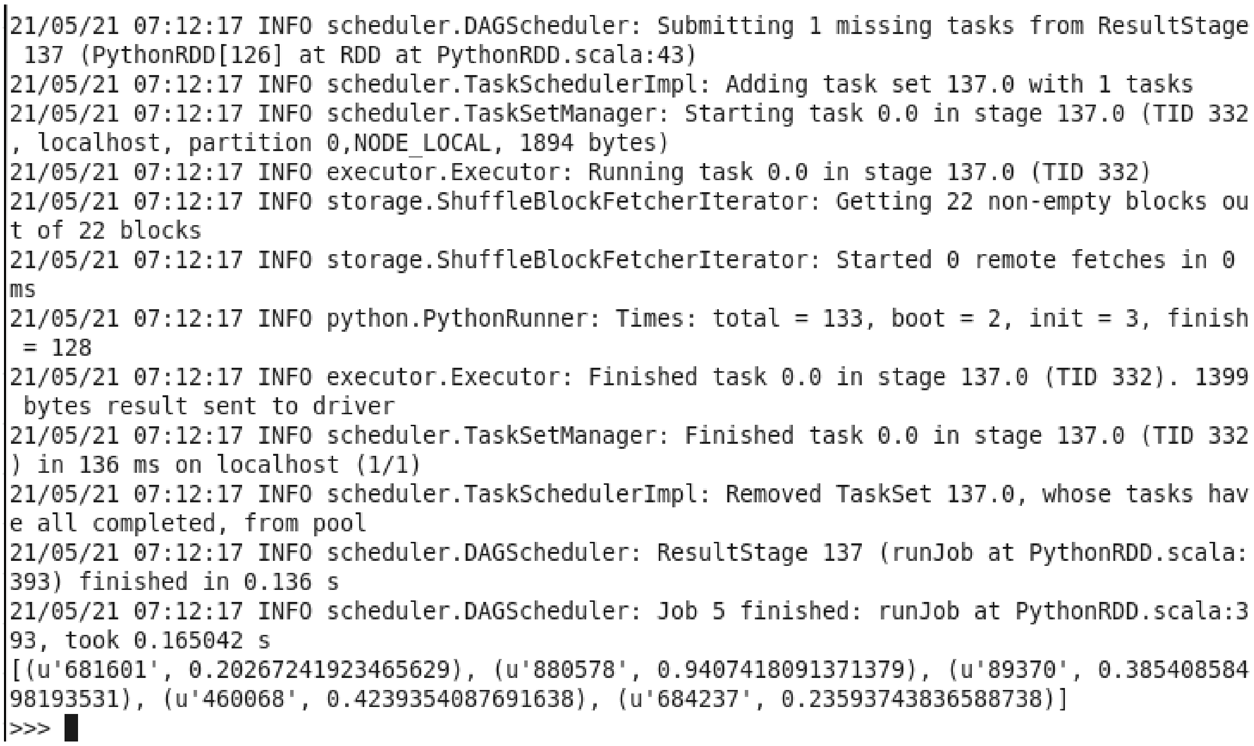


```
>>> ranks.count()
21/05/21 07:09:08 INFO spark.SparkContext: Starting job: count at <stdin>:1
21/05/21 07:09:08 INFO scheduler.DAGScheduler: Got job 2 (count at <stdin>:1) with 22 output partitions
21/05/21 07:09:08 INFO scheduler.DAGScheduler: Final stage: ResultStage 68 (count at <stdin>:1)
21/05/21 07:09:08 INFO scheduler.DAGScheduler: Parents of final stage: List(ShuffleMapStage 67)
21/05/21 07:09:08 INFO scheduler.DAGScheduler: Missing parents: List()
21/05/21 07:09:08 INFO scheduler.DAGScheduler: Submitting ResultStage 68 (PythonRDD[123] at count at <stdin>:1), which has no missing parents
21/05/21 07:09:08 INFO storage.MemoryStore: Block broadcast_25 stored as values in memory (estimated size 6.2 KB, free 76.0 MB)
21/05/21 07:09:08 INFO storage.MemoryStore: Block broadcast_25_piece0 stored as bytes in memory (estimated size 4.0 KB, free 76.0 MB)
21/05/21 07:09:08 INFO storage.BlockManagerInfo: Added broadcast_25_piece0 in memory on localhost:40047 (size: 4.0 KB, free: 454.6 MB)
21/05/21 07:09:08 INFO spark.SparkContext: Created broadcast 25 from broadcast at DAGScheduler.scala:1006
21/05/21 07:09:08 INFO scheduler.DAGScheduler: Submitting 22 missing tasks from ResultStage 68 (PythonRDD[123] at count at <stdin>:1)
21/05/21 07:09:08 INFO scheduler.TaskSchedulerImpl: Adding task set 68.0 with 22 tasks
21/05/21 07:09:08 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 68.0 (TID 308, localhost, partition 0,NODE_LOCAL, 1894 bytes)
21/05/21 07:09:08 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 68.0 (TID 309, localhost, partition 1,NODE_LOCAL, 1894 bytes)
21/05/21 07:09:08 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 68.0 (TID 310, localhost, partition 2,NODE_LOCAL, 1894 bytes)
21/05/21 07:09:08 INFO executor.Executor: Running task 0.0 in stage 68.0 (TID 308)
21/05/21 07:09:08 INFO executor.Executor: Running task 2.0 in stage 68.0 (TID 310)
21/05/21 07:09:08 INFO executor.Executor: Running task 1.0 in stage 68.0 (TID 309)
21/05/21 07:09:08 INFO storage.ShuffleBlockFetcherIterator: Getting 22 non-empty blocks out of 22 blocks
21/05/21 07:09:08 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
21/05/21 07:09:08 INFO storage.ShuffleBlockFetcherIterator: Getting 22 non-empty blocks out of 22 blocks
21/05/21 07:09:08 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
21/05/21 07:09:08 INFO storage.ShuffleBlockFetcherIterator: Getting 22 non-empty blocks out of 22 blocks
21/05/21 07:09:08 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
21/05/21 07:09:09 INFO python.PythonRunner: Times: total = 135, boot = -19864, init = 19868, finish = 131
```

Show first 5 values:

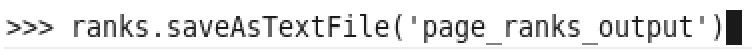


```
>>> ranks.take(5)
21/05/21 07:12:17 INFO spark.SparkContext: Starting job: runJob at PythonRDD.scala:393
21/05/21 07:12:17 INFO scheduler.DAGScheduler: Got job 5 (runJob at PythonRDD.scala:393) w
ith 1 output partitions
21/05/21 07:12:17 INFO scheduler.DAGScheduler: Final stage: ResultStage 137 (runJob at Pyt
honRDD.scala:393)
21/05/21 07:12:17 INFO scheduler.DAGScheduler: Parents of final stage: List(ShuffleMapStag
e 136)
21/05/21 07:12:17 INFO scheduler.DAGScheduler: Missing parents: List()
21/05/21 07:12:17 INFO scheduler.DAGScheduler: Submitting ResultStage 137 (PythonRDD[126]
at RDD at PythonRDD.scala:43), which has no missing parents
21/05/21 07:12:17 INFO storage.MemoryStore: Block broadcast_28 stored as values in memory
(estimated size 5.6 KB, free 76.0 MB)
21/05/21 07:12:17 INFO storage.MemoryStore: Block broadcast_28_piece0 stored as bytes in m
emory (estimated size 3.6 KB, free 76.0 MB)
21/05/21 07:12:17 INFO storage.BlockManagerInfo: Added broadcast_28_piece0 in memory on lo
calhost:40047 (size: 3.6 KB, free: 454.6 MB)
21/05/21 07:12:17 INFO spark.SparkContext: Created broadcast 28 from broadcast at DAGSched
uler.scala:1006
21/05/21 07:12:17 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage
 137 (PythonRDD[126] at RDD at PythonRDD.scala:43)
21/05/21 07:12:17 INFO scheduler.TaskSchedulerImpl: Adding task set 137.0 with 1 tasks
21/05/21 07:12:17 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 137.0 (TID 332
, localhost, partition 0,NODE_LOCAL, 1894 bytes)
21/05/21 07:12:17 INFO executor.Executor: Running task 0.0 in stage 137.0 (TID 332)
21/05/21 07:12:17 INFO storage.ShuffleBlockFetcherIterator: Getting 22 non-empty blocks ou
t of 22 blocks
21/05/21 07:12:17 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0
ms
21/05/21 07:12:17 INFO python.PythonRunner: Times: total = 133, boot = 2, init = 3, finish
 = 128
21/05/21 07:12:17 INFO executor.Executor: Finished task 0.0 in stage 137.0 (TID 332). 1399
```

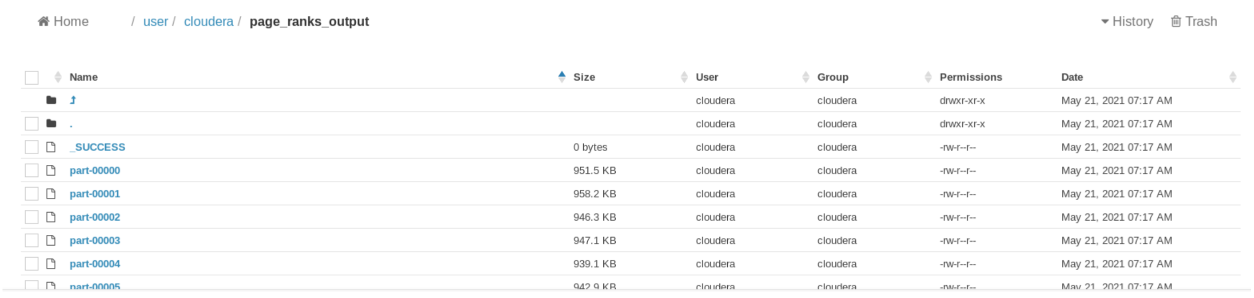


```
21/05/21 07:12:17 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage
 137 (PythonRDD[126] at RDD at PythonRDD.scala:43)
21/05/21 07:12:17 INFO scheduler.TaskSchedulerImpl: Adding task set 137.0 with 1 tasks
21/05/21 07:12:17 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 137.0 (TID 332
, localhost, partition 0,NODE_LOCAL, 1894 bytes)
21/05/21 07:12:17 INFO executor.Executor: Running task 0.0 in stage 137.0 (TID 332)
21/05/21 07:12:17 INFO storage.ShuffleBlockFetcherIterator: Getting 22 non-empty blocks ou
t of 22 blocks
21/05/21 07:12:17 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0
ms
21/05/21 07:12:17 INFO python.PythonRunner: Times: total = 133, boot = 2, init = 3, finish
 = 128
21/05/21 07:12:17 INFO executor.Executor: Finished task 0.0 in stage 137.0 (TID 332). 1399
 bytes result sent to driver
21/05/21 07:12:17 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 137.0 (TID 332
) in 136 ms on localhost (1/1)
21/05/21 07:12:17 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 137.0, whose tasks hav
e all completed, from pool
21/05/21 07:12:17 INFO scheduler.DAGScheduler: ResultStage 137 (runJob at PythonRDD.scala:
393) finished in 0.136 s
21/05/21 07:12:17 INFO scheduler.DAGScheduler: Job 5 finished: runJob at PythonRDD.scala:3
93, took 0.165042 s
[(u'681601', 0.20267241923465629), (u'880578', 0.9407418091371379), (u'89370', 0.385408584
98193531), (u'460068', 0.42393540876916380), (u'684237', 0.23593743836588738)]
>>> 
```

Saving the rdd as text file:

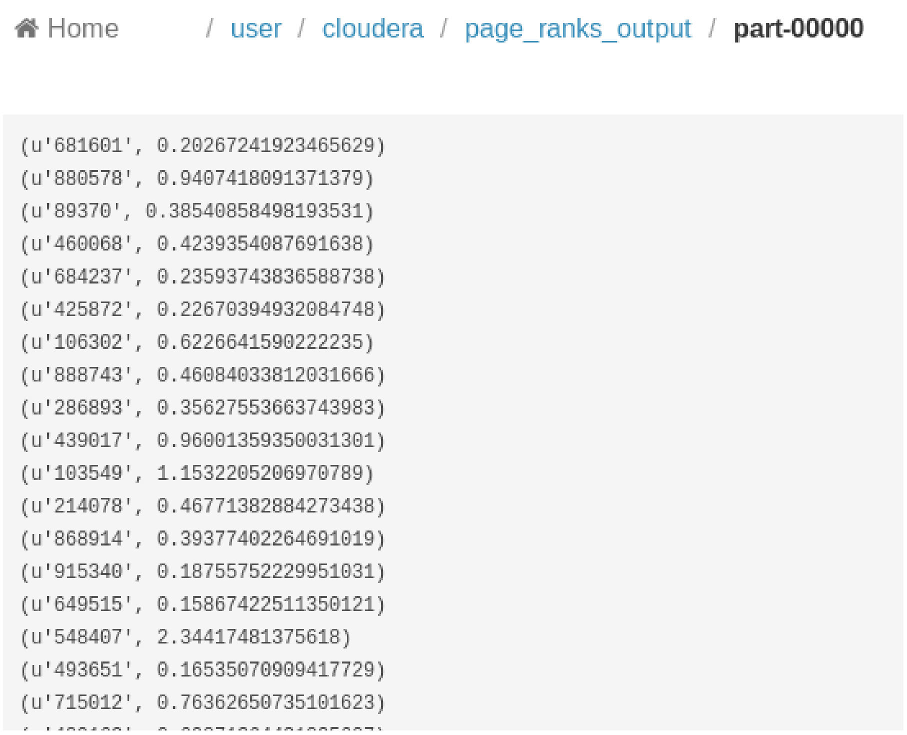


```
>>> ranks.saveAsTextFile('page_ranks_output')
```



Home / user / cloudera / page_ranks_output   History   Trash

Name	Size	User	Group	Permissions	Date
↥		cloudera	cloudera	drwxr-xr-x	May 21, 2021 07:17 AM
.		cloudera	cloudera	drwxr-xr-x	May 21, 2021 07:17 AM
_SUCCESS	0 bytes	cloudera	cloudera	-rw-r--r--	May 21, 2021 07:17 AM
part-00000	951.5 KB	cloudera	cloudera	-rw-r--r--	May 21, 2021 07:17 AM
part-00001	958.2 KB	cloudera	cloudera	-rw-r--r--	May 21, 2021 07:17 AM
part-00002	946.3 KB	cloudera	cloudera	-rw-r--r--	May 21, 2021 07:17 AM
part-00003	947.1 KB	cloudera	cloudera	-rw-r--r--	May 21, 2021 07:17 AM
part-00004	939.1 KB	cloudera	cloudera	-rw-r--r--	May 21, 2021 07:17 AM
part-00005	942.9 KB	cloudera	cloudera	-rw-r--r--	May 21, 2021 07:17 AM



Home / user / cloudera / page_ranks_output / 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', 0.96001359350031301)
(u'103549', 1.1532205206970789)
(u'214078', 0.46771382884273438)
(u'868914', 0.39377402264691019)
(u'915340', 0.18755752229951031)
(u'649515', 0.15867422511350121)
(u'548407', 2.34417481375618)
(u'493651', 0.16535070909417729)
(u'715012', 0.76362650735101623)
```

Results and Discussions :

Page rank can be used to rank pages according to the importance, by assigning the numerical weighting to each element of a hyperlinked set of documents. The algorithm can be easily implemented in hadoop clusters in spark.