Aim: To implement PageRank using pyspark

Theory:
PageRank is an algorithm used to measure the importance of web pages in a network, such as the internet. It was developed by Larry Page and Sergey Brin, the founders of Google, and is a fundamental concept in web search and graph analysis. PageRank works by analyzing the link structure of web pages to determine their relative importance within the network.

Uses of PageRank:

1. Web Search: PageRank is a core component of Google's search algorithm. It helps determine the ranking of web pages in search engine results pages (SERPs). Pages with higher PageRank are considered more important and are more likely to appear at the top of search results.
2. Recommender Systems: PageRank is used in recommendation systems, such as suggesting products, articles, or videos to users. By analyzing user interactions and the link structure between items, PageRank can identify relevant recommendations.
3. Social Network Analysis: PageRank is applied to social networks to identify influential users or nodes. In a social network, individuals with a high PageRank are considered influential and may be targeted for marketing or outreach.
4. Citation Analysis: PageRank is used in academic and research contexts to assess the importance of research papers or academic journals. Papers with a higher PageRank are considered more influential in their field.
5. Community Detection: PageRank can be used to identify communities or clusters of nodes within a network. Nodes with higher PageRank may serve as hubs connecting different communities.

Steps to Perform PageRank with Example:

Here are the fundamental steps to perform PageRank, along with a simplified example:

1. Data Representation: Represent the network of web pages as a graph, where web pages are nodes, and hyperlinks between pages are edges. For this example, let's consider a small network of web pages.

```
A -> B
B -> A, C
C -> B
```

2. Initialization: Assign an initial PageRank value to each web page. In the first iteration, all pages are given an equal PageRank value.

```
PR(A) = PR(B) = PR(C) = 1
```

3. Iterative Computation:

- In each iteration, calculate the new PageRank for each web page based on the PageRank of linking pages.
- Distribute the PageRank of linking pages equally among their outgoing links.
- Apply a damping factor (usually 0.85) to account for random browsing.

Iteration 1:

```
PR(A) = 0.15 + 0.85 * (PR(B)/1)
PR(B) = 0.85 * (PR(A)/2) + 0.85 * (PR(C)/1)
PR(C) = 0.85 * (PR(B)/1)
```

Iteration 2:

```
PR(A) = 0.15 + 0.85 * (PR(B)/1.5)
PR(B) = 0.85 * (PR(A)/2) + 0.85 * (PR(C)/1)
PR(C) = 0.85 * (PR(B)/1)
```

Continue iterating until convergence.

4. Convergence: Stop the iterations when the PageRank values stabilize (converge). This means that further iterations do not significantly change the PageRank values.
5. Output: The final PageRank values represent the importance of web pages within the network. Higher PageRank indicates greater importance.

Example:
Consider a simplified web network with three pages: A, B, and C. The links between pages are as follows:

```
A -> B
B -> A, C
C -> B
```

We will perform two iterations of PageRank calculation.
Iteration 1:
- Initialize PR(A) = PR(B) = PR(C) = 1.
- Calculate the new PageRank values:

```
PR(A) = 0.15 + 0.85 * (1/1) = 1
PR(B) = 0.85 * (1/2) + 0.85 * (1/1) = 0.925
PR(C) = 0.85 * (1/1) = 0.85
```

Iteration 2:
- Update the PageRank values:

```
PR(A) = 0.15 + 0.85 * (0.925/2) = 0.96375
PR(B) = 0.85 * (0.96375/2) + 0.85 * (0.85/1) = 0.905125
PR(C) = 0.85 * (0.905125/1) = 0.76935625
```

After two iterations, the PageRank values have converged. The final PageRank values indicate the importance of each web page in the network.
PageRank is a fundamental concept with numerous applications, particularly in web search and network analysis. It helps identify the most important elements within a network by considering the link structure and connectivity. The damping

factor ensures that users may randomly jump to any page, accounting for the "random surfer" model. While this example is highly simplified, real-world PageRank implementations handle much larger networks and require more sophisticated algorithms and distributed computing frameworks like PySpark.

Code:

```
def computeContribs(neighbors, rank):
	for neighbor in neighbors:
		yield(neighbor, rank / len(neighbors))
linkfile = sc.textFile("file:/home/cloudera/Desktop/Pagerank.txt")
links = linkfile.map(lambda line: line.split())\
	.map(lambda pages: (pages[0], pages[1]))\
	.distinct()\
	.groupByKey()\
	.persist()
ranks = links.map(lambda (page, neighbors):(page, 1, 0))
n = 15
for x in range(n):
	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))
for pair in ranks.take(15):
	print pair
```

Output:

```
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))
...
>>> linkfile = sc.textFile("file:/home/cloudera/Desktop/Pagerank.txt")
>>> links = linkfile.map(lambda line: line.split())\
... .map(lambda pages: (pages[0], pages[1]))\
... .distinct()\
... .groupByKey()\
... .persist()
23/10/15 01:57:40 WARN shortcircuit.DomainSocketFactory: The short-circ
>>> ranks = links.map(lambda (page, neighbors):(page, 1, 0))
>>> n = 15
```

```
SyntaxError: invalid syntax
>>> for x in range(n):
...     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))
...
>>> for pair in ranks.take(15):
...     print pair
...
(u'page6', 0.51343021620547935)
(u'page4', 0.51343021620547935)
(u'page5', 0.58095604508082921)
(u'page2', 0.64217474671055641)
(u'page3', 0.4275649602417404)
(u'page1', 0.4275649602417404)
```

Conclusion:

In conclusion, PageRank is a critical algorithm for assessing the importance of web pages and nodes in various networks. It is widely used in web search, recommendation systems, social network analysis, and more. PageRank's iterative approach distributes importance through the network based on the link structure, converging to stable values that reflect the relative significance of nodes. However, while PageRank provides valuable insights, it has limitations such as sensitivity to initial conditions, requiring large-scale computations for real-world applications, and potential bias toward older pages.