第八次作业

目录

[PageRank 1](#_Toc10830802)

[环境 1](#_Toc10830803)

[技术方法 2](#_Toc10830804)

[结果 3](#_Toc10830805)

[心得 3](#_Toc10830806)

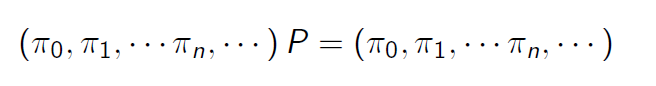
[代码 3](#_Toc10830807)

# PageRank

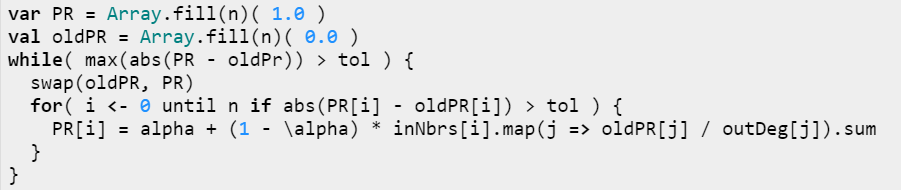
pagerank 模型模拟一个用户在互联网上浏览到每个网页的概率,根据网页相互之间的链接给网页赋值，迭代到收敛为止即为网页的概率。

Pagerank本质是马尔可夫链，每个网页即为每个状态，网页链接即为状态转移的概率，平稳分布即为最终结果

P转移概率矩阵 为状态i的平稳概率，即i节点的权重



解这个方程：



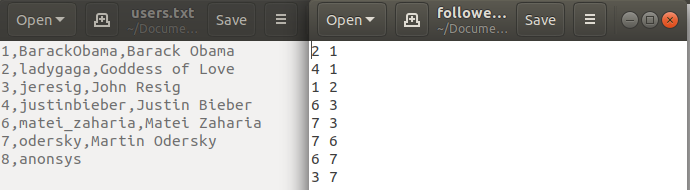
# 环境

eclipse

scala

# 技术方法

使用示例数据进行pagerank，根据社交网络的指向判断各节点的重要性



新建spark context

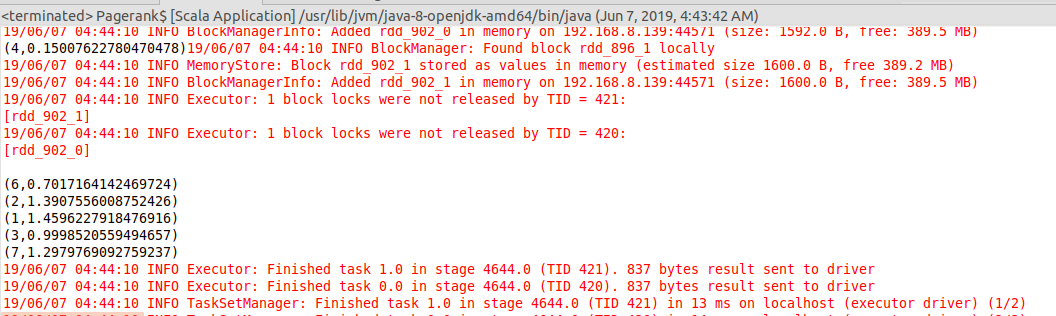


导入边信息，建立有向图，边没有属性，只有srcId和dstId

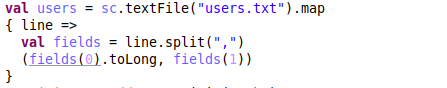


在该有向图进行pagerank，指定收敛阈值为0.0001，当每个点的权值变化小于该阈值时认为该算法收敛，返回值为RDD，每行为点的id和权值



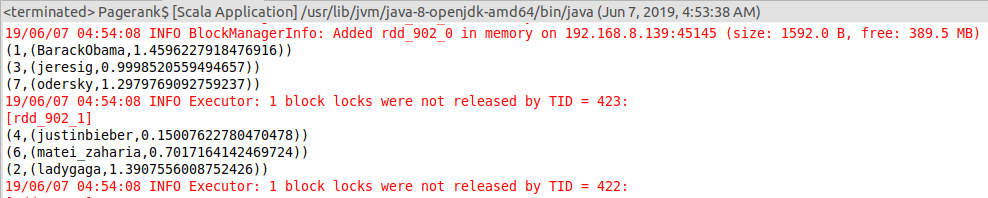


读入每个每个点的信息到一个RDD中，每行按逗号分割为id和该点的信息

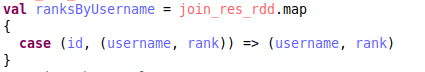


join两个RDD，得到新的RDD，每行为id，该点权值，该点信息



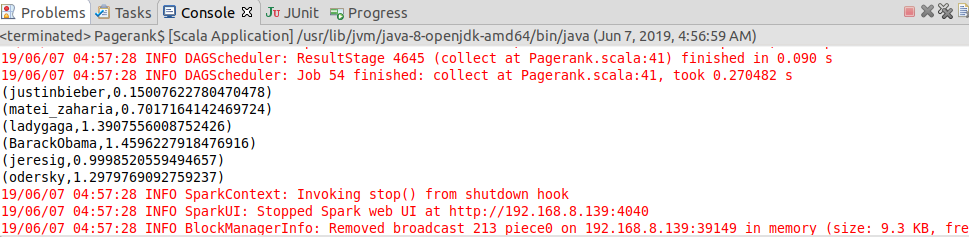


将join得到的RDD的每行map成该点信息和权值，即为结果



# 结果

每个节点的重要性



# 心得

graphx的pagerank函数有两种，静态pagerank迭代固定次数，动态pagerank迭代到收敛

两个RDD可进行join操作，结果形如(id, (username, rank))

# 代码

**scala**

**def** pagerank():Unit=

{

**val** conf = **new** SparkConf().setAppName("Pagerank").setMaster("local[2]")

**val** sc = **new** SparkContext(conf)

// Load the edges as a graph

**val** graph = GraphLoader.edgeListFile(sc, "followers.txt")

// Run PageRank

**val** ranks = graph.pageRank(0.0001).vertices

// ranks.foreach(println)

// Join the ranks with the usernames

**val** users = sc.textFile("users.txt").map

{ line =>

**val** fields = line.split(",")

(fields(0).toLong, fields(1))

}

**var** join\_res\_rdd=users.join(ranks)

// join\_res\_rdd.foreach(println)

**val** ranksByUsername = join\_res\_rdd.map

{

**case** (id, (username, rank)) => (username, rank)

}

// Print the result

println(ranksByUsername.collect().mkString("\n"))

}