COSC 589 - Web Search and Sense-Making

Assignment 7 (Optional)
Due on Wednesday April 25, 11:59pm

Task: Topic-Oriented PageRank

Introduction:

In this assignment, we will implement the Topic-Oriented PageRank algorithm on the Wikipedia data.

Requirements:

>100GB free disk space in your machine.

Instructions:

Write a TopicPageRank.scala file to calculate and sort the topic-oriented pagerank scores for the English Wikipedia pages. The topic we select for this homework is “The Football League Players”. The steps are the following:

1. Read in the output files of your Assignment 8, which contains the link graph in Wikipedia. The format should be:
   - One page per line
   - In each line, you have the title of a page and a list of the titles of the outlinks in the page
   - Each outline title is inside [[]], and separated by a tab “\t”.
   - The title and the list of links is separated by a delimiter. We recommended “\t”.
2. Read in the titles of the pages that are on the topic of “The Football League Players” from the provided file.
3. Implement the Topic-Oriented PageRank algorithm. Assign a jump probability of 100% to the pages on a topic and assign the other pages 0%. The formula looks like:

$$r(\alpha) = \delta \cdot \left(\sum_{\beta \to \alpha} \frac{r(\beta)}{out(\beta)} + \sum_{\gamma \in \Gamma \cap T} \frac{r(\gamma)}{|T|} \right) + (1 - \delta) \cdot \sum_{\alpha' \in T} \frac{r(\alpha')}{|T|}$$

where T is the set of pages that are on the topic. Alpha is a page.

The algorithm looks like:

```
val links =  // Load RDD of (page title, outlinks) pairs
var ranks = // Load RDD of (page title, rank) pairs
val topicPages = bTopicPages.value // get RDD of (page title) from a broadcast variable

for (i <- 0 to ITERATION) {
    val contribs = links.join(ranks).flatMap {
      case (title, (links, rank)) =>
        links.map(dest => (dest, rank / links.size))
    }
    onTopicRank = contribs.reduceByKey( _+_ ).filter( x =>
topicPages.contains(x._1)).mapValues(0.15 + 0.85 * _ )
    offTopicRank = contribs.reduceByKey( _+_ ).filter( x => !
topicPages.contains(x._1)).mapValues(0.85 * _ )
    rank = onTopicRank.union(offTopicRank)
}
```

4. Calculate the topic-oriented pagerank scores for all the pages in the English Wikipedia. Use ITERATION = 10.
5. Sort the topic-oriented pagerank scores for the pages in the descending order.
6. Save the results. The format is like:

```
[[Association football]]        7.557400770279294
[[association football]]        6.509902882920945
[[Midfielder]]  5.322665101195386
[[FA Cup]]      5.275953121181659
[[Football League Cup]] 5.1405548413603395
[[Defender (association football)]]     4.781431725368732
[[Premier League]]      4.460411498158475
[[Football League Championship]]        4.138733764918339
[[Football League One]] 4.120904050031456
[[Football League Two]] 3.8326805491447398
[[Football League Trophy]]      3.689606029687729
[[Forward (association football)]]      3.245154610067562
```

6. You are welcome to use the following code template:

```
import scala.util.matching.Regex
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.rdd.RDD
import org.apache.hadoop.io.compress.GzipCodec
import org.apache.spark.broadcast.Broadcast

object TopicPageRank {
```

```
def main(args: Array[String]) {
  val sparkConf = new SparkConf().setAppName("TopicPageRank")
  val sc = new SparkContext(sparkConf)

  // load link graphs
  val input = sc.textFile("./linkgraph/*.gz")  // your output directory from the assignment 7
  val links =  // Load RDD of (page title, links) pairs
  val ranks = // Load RDD of (page title, rank) pairs


  // load topic pages
  val football = sc.textFile("./football/part*").map(r => "[[" + r + "]]").toArray
  // create a broadcast variable to hold the football pages
  val  bTopicPages = sc.broadcast(football.toSet)

  // Implement your Topic-Oriented PageRank algorithm
  val ITERATION = 10
  …

  // Sort pages by their PageRank scores
  ranks.sortBy …

  // save the page title and pagerank scores in compressed format (save your disk
space). Using "\t" as the delimiter.
  ranks.map(r => r._1 + "\t" + r._2).saveAsTextFile("./topicpageranks",
classOf[GzipCodec])

 }
}
```

What to Submit:

- Your code
- Screen captures of the beginning of your pageranked pages, in descending order (e.g. the first 20 lines on the screen. Hint: Use 'gunzip part-00000.gz" to unzip, then view the documents and screen capture)

What NOT to Submit:

- Your input or output files

Where to submit:

- Canvas

When:

- Due on 04/25/2018, 11:59pm.