INF344 TP3: PageRank

Oana Balalau
Mauro Sozio
firstname.lastname@telecom-paristech.fr

June 3, 2014

In this exercise, we will write a code in MapReduce/Hadoop for the PageRank algorithm. In input we are given a web graph G where each line "j i" represents a link $j \to i$. You should implement:

1. an algorithm for removing dead-ends, that is, nodes with no out-going edges (which correspond to pages that do not have links to other pages in the current graph). You might need several iterations to remove all dead-ends. Let H be the graph with no dead-ends so obtained.

2. an algorithm that builds the stochastic matrix M associated to the web graph H. Recall that, if there is a link $j \to i$ in H and j has k successors (that is k links to k pages) then $M_{ij} = 1/k$ otherwise $M_{ij} = 0$. Use the sparse representation of a matrix where zeros are not represented.

3. the PageRank algorithm where you should put all pieces together (removing dead-ends, from graph to matrix, and matrix-vector multiplication). The input of the PageRank algorithm is H. See the last slide of our class for a full pseudocode. In this solution the nodes that were removed are not reintroduced.

Hadoop classes that are needed:

- Configuration class - it is useful to set configuration properties. A configuration property has a name (String) and a value (int, long, double). We use an instance of the Configuration class to communicate to the

Hadoop jobs some necessary parameters (path to input, output, intermediary folders, values of beta and epsilon). For our exercise the values are already attributed in the file PublicTests.java. You should not modify them .

In a cluster, each Mapper and Reducer has its own Configuration object. In this situation, modifying a configuration parameter from the map or reduce function will not be permanent and at the end of the job you will lose the information.

`http://hadoop.apache.org/docs/r2.3.0/api/org/apache/hadoop/conf/Configuration.html`.

- Counter class - it is used to get statistics from the data that you are analysing. Hadoop has build-in counters for every job, which, for example, compute how many records a map/reduce task had as input and produced as output. These counters are displayed at the end of the task. You can define your own counter(Java enum) in order to get information from the job. The counter is global, so all changes are aggregated at the end to produce the final value. You should use a counter to check the stopping condition for the first part of the exercise.

  `http://hadoop.apache.org/docs/r2.3.0/api/org/apache/hadoop/mapred/Counters.html`.

- MultipleOutputs class - useful when we want to generate additional output. In a general situation, it could be that we want to have a file per key or a file per specific property of the value. In our case we use two outputs in order the generate in a single job the matrix and the initial vector.

  `http://hadoop.apache.org/docs/r2.3.0/api/org/apache/hadoop/mapreduce/lib/output/MultipleOutputs.html`.

As an extra information for future projects, you can also use the class MultipleInputs when you want to have different Mappers and InputFormat for specific paths.

`https://hadoop.apache.org/docs/r2.3.0/api/org/apache/hadoop/mapreduce/lib/input/MultipleInputs.html`

You need to submit the PageRank.java file on the submission server before Friday, June 6, 5pm.