dfsfdBig Data Analytics
Stony Brook University
CSE545 - Fall 2016

Assignment 1

Assigned: 9/18/2016; Due: ~~9/29/2016~~10/1/2016, 3:45pm EST

Part I. MapReduce (65 points)

Here, you will complete a back-end for a MapReduce *system* and test it on a couple MapReduce jobs: word count (provided), and matrix multiplication (you must implement). Template code is provided here: http://www3.cs.stonybrook.edu/~has/CSE545/a1_lastname.py

Specifically, you must complete:

- Methods of the `class MyMapReduce:`
  - `partitionFunction(self,k)` #5 points
  - `reduceTask(self, kvs, from_reducer)` #15 points
  - `runSystem(self)` #30 points

  Some of these methods are already partially complete. "[TODO]" indicates sections needing completion. This is an abstract class, meaning it is not instantiated directly. One must inherit the class and override map and reduce methods (e.g. see WordCountMR). runSystem must make a separate process for each mapTask and reduceTask.
- `class MatrixMultMR(MyMapReduce)` #15 points
  Matrix multiplication of two matrices. Your input and output should be of the form: (("label", row, col), value) where "label" is a name for the matrix (there should only be two for input, one for output).

Notes: (may be updated as questions come in)

- The goal is to learn how MapReduce works by implementing a potential back-end MapReduce system. There are, of course, faster implementations for word count on a single machine.
- Due to python multiprocessing restrictions in passing data, we use "return" rather than "yield".
- Similarly, in this MapReduce system, all mappers will complete before any reducers start (this allows us to track the output of the mappers more clearly).
- You may add methods if desired, but all existing code must be used in your implementation (i.e. no deleting or bypassing any of the code provided). This ensures solutions remain true to the goal of the assignment and do not deviate from simulating a multi-node setup.
- Jupyter and ipython notebook handle namespace different than typical python code and this may present a problem for multiprocessing. If you get a module attribute not found issue, try to work directly with python (i.e. create a .py file and run it through the python interpreter).
- Start by tracing through the steps of `runSystem.`

- For MatrixMultiMR, if there is other information you want shared between mappers or reducers, you may override the MyMapReduce constructor and assign instance variables.

Part II. Minhashing (35 points)

Here, you will implement efficient minhashing, as described in MMDS 3.3.5 where one simulates the random permutations of minhashing by using hash functions. Specifically, you must complete the method "`minHash(documents, k)`" within the same template code. You should choose an appropriate number of hash functions to use. Output should be a matrix of the form: [[<row1 cells>], [<row2 values>], …] (or within a numpy.array)

Guidelines.

All code should be placed in a single file "a1_lastname.py" or "a1_lastname.java". Include your name at the top of the code as well. We should be able to run the file as is (i.e. "python a1_lastname.py"), as well as be able to import your MyMapReduce and MatrixMultMR classes to test on different data.

Submission. One single, uncompressed file should be submitted via blackboard.

Python libraries permitted: default file IO libraries, numpy/scipy (only for arrays and algebra, no statistical tests or regression), pprint, multiprocessing.

Testing: Our testing will consist of both the code contained in "main" within the template, as well as running with slight edits to the data. Sample output based on the template "main" will be provided approximately 1 week before the assignment is due.

Questions / Clarifications: Please post questions on Piazza, so other classmates may see the answers. Questions posted within 48 hours of the deadline are not guaranteed a response before the deadline.

Academic Integrity: As with all assignments (sans the team project), although you may discuss concepts with others, you must work independently and insure your work and code is not visible to any classmates. You may also not copy any partials solutions from the Web or other resources though you may reference algorithm descriptions and method parameter definitions.