CSE545: Fall-2017 Assignment 1

Big Data Analytics
Stony Brook University
CSE545 - Fall 2017

Assignment 1

Due: 9/28/2017, 3:45pm EST

Part I. MapReduce (50 points)

Here, you will complete a back-end for a MapReduce *system* and test it on a couple MapReduce jobs: word count (provided), and set difference (you must implement). Template code is provided here: http://www3.cs.stonybrook.edu/~has/CSE545/a1p1_lastname.py (Java template code is here: http://www3.cs.stonybrook.edu/~has/CSE545/assignments/srcHW1.zip)

Specifically, you must complete:

- Methods of the `class MyMapReduce:`
  - `partitionFunction(self,k)` #5 points
  - `reduceTask(self, kvs, from_reducer)` #10 points
  - `runSystem(self)` #20 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 SetDifferenceMR(MyMapReduce)` #15 points

  Set difference map and reduce. Your input will be of the form: [("R", [setR_contents]), ("S", [setS_contents]). Write a map and reduce which will find the difference between the two sets: R - S. Do not use any set operations already present (e.g. set(setR_contents).intersect(setS_contents)).

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 and set difference on a single machine.
- 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.
- SetDifference will always just start with just two records -- the idea is to test your code on a different type of MapReduce logic.

Part II. Spark (50 points)

Download and install Spark 2.2.0:
https://spark.apache.org/downloads.html
(Get the version pre-built for Hadoop as it includes necessary libraries. You do not need to have hadoop installed.)

This portion of the assignment is focused on implementing algorithms in Spark. As assignment 2 and the team project will work on a cluster, the goal here is to first learn to *code* in Spark (i.e. as a series of transformations).

(a) Implement WordCount and SetDifference in Spark. #20 points. Limit your transformations to just map, flatMap, reduceByKey, groupByKey, mapValues (do not need to use all). You can use a filter at the end. Use comments to clearly label "WordCount implementation below" and "Set Difference Implementation below". Start by creating an rdd using the same data as part 1. E.g. for set difference:

```
#Set Difference Implementation below:
```

```
data = [('R', [x for x in range(50) if random() > 0.5]),
           ('S', [x for x in range(50) if random() > 0.75]]
rdd = sc.parallelize(data)
#your code here:
rdd.transition….
```

We should be able to replace data with other sets (containing any types of comparable elements) and have it run correctly.

(b) Search for mentions of industry words in the blog authorship corpus. #30 points. The goal here is to first find all of the possible industries in which bloggers were classified. Then, to search each blogger's posts for mentions of those industries and, counting the mentions by month and year.
Download the corpus here: http://u.cs.biu.ac.il/~koppel/BlogCorpus.htm (Note: the site rate-limits the speed of the download. It will take several minutes.)
Unzip the corpus file and look at the contents of a few files before reading on. Each file in the corpus is named according to information about the blogger: `user_id.gender.age.industry.star_sign.xml`
Within each xml file, there is a "<date>" tag which indicates the date of a proceeding "<post>", which contains the text of an individual blog post.
You should take the following steps:

(i) *Get all possible industry names:*

(1) Create an rdd of all the filenames (your code should have a variable defined with the directory where all the files are stored)
(2) Use transformations until you are left with only a set of possible industries
(3) Use an action to export the rdd to a set and make this a spark broadcast variable

(ii) *Search for industry names in posts, recording by year-month:*

(1) Create an rdd for the contents of all files [i.e. sc.wholeTextFiles(file1,file2,...)]
(2) Use transformations to search all posts across all blogs for mentions of industries, and record the frequency each industry was mentioned by month and year. The industry names should only be matched, case insensitive, if they are next to a word boundary -- space or punctuation (e.g. "marketing" would match "I am in marketing sales" and "Marketing." but not "I like supermarketing." or "This is marketing5 now.").
(3) Use an action to print the recorded frequencies in this format:
[(industry1, ((year-month1, count), (year-month2, count), …),
(industry2, ((year-month1, count), (year-month2, count), …), …]

Notes: (may be updated as questions come in)

- For (b), you must follow the steps provided. The transformations you choose in order to achieve the objective efficiently are up to you.
- If using python spark (pyspark), then pip install pyspark is typically the easiest way to get spark. On windows, you can first install Anaconda to get "pip".
- The XML files do not abide by modern standards for well-formed XML. Such is life with most data. You may try using an xml parser with relaxed checks or use string patterns / regular expressions to capture the text. It is outside the scope of the assignment to fix typos (e.g. for dates) -- work with the data you are given.
- There are multiple solutions in spark. You should try to use an efficient solution but the intent is that you just use spark transformations to get to the results.

Guidelines.

All code should be placed in to three files:

1) "a1p1_lastname.py"
2) "a1p2a_lastname.py"
3) "a1p2b_lastname.py"

(replace ".py" with ".java" or ".scala" if appropriate).
We should be able to run the first file as is (i.e. "python a1p1_lastname.py"), as well as be able to import your MyMapReduce and SetDifferenceMR classes to test on different data. For the Spark files we should be able to run them with spark-submit. Use 3 separate jars for Java or Scala.

Submission. Three separate, **uncompressed** files should be submitted via blackboard.

Python 3.5 libraries permitted: default file IO libraries, numpy/scipy (only for arrays and algebra, no statistical tests or regression), pprint, multiprocessing. No language processing libraries may be used and only the default Spark 2.2.0 transformations and actions.

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.

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 any other resources though you may reference algorithm descriptions and method parameter definitions.