Assignment set 2

Returning your solutions

Return your solutions via Moodle page. We do not count belated returnings. You have to use .pdf file type for any written answers, and .scala/.py for your code. Zip your code and pdf files together and upload. Do not return any project or object files, etc. Mark your **full name and student number** clearly to all your solution files. Each exercise will be graded pass/fail.

Spark API Documentation Links for Scala
http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.package

Spark API Documentation Links for Python
http://spark.apache.org/docs/latest/api/python/index.html

Programming Guidelines

You can download the skeleton file for Scala and Python.

This way we can import and run all of the code of all the students in a single Eclipse project.

Exercises 1-6 are about coding Spark. You can run them in your own computer. You should return your solutions as .scala or .py files, but remember to mark clearly numbers of the exercises. Using comments also helps us to understand what you have done.

TIP: If you create your own classes, make them Serializable and override hashCode(). Or just use case classes. This way Spark is able to use your classes properly, otherwise you may get a class not serializable exception.

Note: *In Python, stored objects will always be serialized with the Pickle library, so it does not matter whether you choose a serialized level. If you create classes, store them in a file not containing the main funciton. This will later allow you to Pickle the class objects. If you get encode error while storing Pickle objects, refer the second StackOverFlow link after Question 1*

Exercise 1

Download the Book-Crossing data set in CSV format from: https://moodle.helsinki.fi/pluginfile.php/915616/mod_forum/attachment/1239670/BX-dataset-fixed.tar.gz

Read users, books and ratings as RDDs using the textFile function. You should use separate serializable case classes for each csv files. The members of the classes are the columns of the csv files. Finally you will have three RDDs of three different object types and save these object RDDs (Hint: use booksrdd`.saveAsObjectFile` for Scala, and use booksrdd.`saveAsPickleFile` for Python).

- BX-Books has isbn, title, author, year, publisher, url1, url2, url3 (Hint: Inside map function skip these urls. Your case class for books should have isbn, title, author, year, publisher fields.)
- BX-Book-Ratings has user_id, isbn, rating (Your case class for ratings should have user_id, isbn, rating fields.)
- BX-Users has user_id, location, age (Your case class for users should have user_id, location, age fields.)

You should use two partition settings; without specifying partitions, and specifying 5 partitions inside the textFile function and discuss the performance gain in total execution time.

http://stackoverflow.com/questions/33639009/pyspark-using-object-in-rdd

http://stackoverflow.com/questions/9942594/unicodeencodeerror-ascii-codec-cant-encode-character-u-xa0-in-position-20

Exercise 2

Read users, books and ratings object RDDs from the disk, as you saved in the first exercise. Join all the RDDs so that your result is only one RDD where each element is a book with its corresponding reviews. The final structure could be something like RDD[(ISBN, title, author, year, publisher, Set(userID, location, age, rating))].

Hint: Reading ObjectFile or PickleFile can be done using the SparkContext object.

Be sure to choose correct data types for each field, e.g. ratings are Integers. You can use case classes to help your work. Bear in mind that although the value for age is an integer, some entries have a “NULL” value which cannot be parsed with the toInt() method. Use whatever placeholder value you deem appropriate instead, like -1.

Hint: Remember that you can rearrange fields within an RDD using map() and you can use the join() method to merge elements of the form (key, value) which have the same key.

Exercise 3

Write a function that computes how many reviews there are for books published between two given years. Apply the function to the resulting RDD from exercise 1 and find out the number of reviews for books published between 1992 and 1998.

Hint: You can use filter() to omit elements with other publication dates. You can use map() and reduceByKey() to count elements with the same key.

Exercise 4

Write a function that takes the RDD created in Exercise 1, and returns an RDD of the 20 authors with the highest average age among their reviewers. Filter out the entries with a "NULL" value. Use only operations over RDDs (count, sort, etc.); the use of for loops is not allowed.

Exercise 5

Repeat exercise one with dataframes. You will save three DataFrames for three textfiles. You do not need to construct the case class objects. Apply filtering to find out the number of reviews for books published between 1992 and 1998.

Exercise 6:

Write short notes on HDFS, Spark Tachyon, Remote direct memory access (RDMA), and Spark DataFrames. Each of them should be explained in 7 sentences.