Homework 1

The purpose of this first homework is to set up the environment for developing Spark code on your machine, and to get acquainted with Spark by implementing some simple MapReduce algorithms.

Make sure you do the **two preparation steps** described below before doing the actual **assignment**, which is described at the end of the page.

Preparation step 1: machine setup

To set up your pc to develop and run Spark programs, follow carefully the instructions given in this Machine setup guide. The guide will also instruct you to download the template for the homework (TemplateHW1.java for Java users, and TemplateHW1.py for Python users) together with a sample input dataset.

Preparation step 2: familiarization with the template

To fully understand the template code that you just downloaded as instructed during machine setup, refer to this Spark programming guide, which introduces you to functional programming and to the main Spark methods relevant for the homeworks. *The guide will be continuously updated as the course proceeds, so to provide always a useful reference*.

The template contains a simple program which does the following things:

- Receives in input an integer (the number of partitions that the program wants to use) and the path to a file, relative to the home directory. The file contains several documents, one document per line. A document is just a set of words separated by single space.
- Sets the Spark configuration and context.
- Reads the input integer into a variable K and reads the input file into an RDD `docs` whose elements (strings) are the documents contained in the file. The RDD is subidvided at random into K partitions by invoking method `repartition`.
- Prints the number of documents in the RDD `docs`.
- Executes 3 instances of Word count:
    - The first instance implements the standard 1-round MapReduce algorithm (Slides 21-22 on MapReduce), using method `reduceByKey` to implement the reduce phase.
    - The second instance implements the 2-round MapReduce algorithm (Slides 29-33 on MapReduce), using random keys in [0,K-1]. Note that K is the value passed as input (in the slides K=sqrt(N) was used).
    - The third instance also implements the 2-round MapReduce algorithm, but rather than using the partition obtained by assigning a random key to each intermediate key-value pair, it uses the partition made by Spark.
- After each instance, the program prints the number of distinct words in the input documents. And after the last instance it also prints the (floor of the) average length of the distinct words.

Run the program and check the result (for Java user, the results will be found at the bottom of the Intellij window following a number of lines of log messages).

You are encouraged to use the template as a skeleton for the Homework 1 assignment described below.

Assignment

You are asked to implement different versions of the *class count algorithm* from Slides 35-36 on MapReduce. A precise specification of your assignment is the following.

You must write a program **GxxHW1.java** (for Java users) or **GxxHW1.py** (for Python users), where xx is your two-digit group number, which receives in input an integer K and path to a text file containing a collection of pairs (i,gamma_i). The text file stores one pair per line which contains the key i (a progressive integer index starting from 0) and the value gamma_i (a string) separated by single space, with no parentheses or comma. (*Note that while in the problem defined in the slides the i-th input pair contained also an object o_i, here the objects are disregarded.*)

The program must do the following things:

1. Reads the input set of pairs into an RDD `pairStrings` of strings, and subdivides it into K parts;
2. Runs the following two versions of the Class count algorithm:
   - *Version with deterministic partitions*: implements the algorithm described in class except that in Round 1 each key i is mapped into "i mod K" rather than "i mod sqrt(N)". (**Hint:** you can modify the second version of Word count in the template.)

     Output: this version must print the following lines:

     ```
     VERSION WITH DETERMINISTIC PARTITIONS
     Output pairs = List of final (class, count) pairs, sorted by class.
     ```

   - *Version with Spark partitions*: implements the algorithm described in class except that in the reduce phase of Round 1 each reducer processes one of the partitions of the RDD defined by Spark. Together with the output pairs (class, count) the algorithm must produce a special pair ("maxPartitionSize",N_max), where N_max is the max number of pairs that in Round 1 are processed by a single reducer. (**Hint:** you can modify the third version of Word count in the template.)

     Output: this version must print the following lines:

     ```
     VERSION WITH SPARK PARTITIONS
     Most frequent class =  pair (class, count) with max count
     Max partition size =  N_max.
     ```

     Note: for the most frequent class, ties must be broken in favor of the smaller class in alphabetical order).

On this example.txt text file you find an example of output of your program on a small input dataset. To test your program on a larger input dataset use file input_10000.txt, which contains 10000 pairs encompassing 10 distinct classes. The output on this dataset is shown in file output_10000.txt. (Note that the values of max partition size might vary.)

Make sure to add short but explicative comments to your code, when needed.

SUBMISSION INSTRUCTIONS. Submit your homework using the submission form of the **Homework 1 section** in the Moodle page of the course. Each group has to submit just a single file (**GxxHW1.java** or **GxxHW1.py** depending on whether you are Java or Python users, where xx is your ID group). Make sure that your code be free from compiling and run-time errors, otherwise your score will be penalized.

*If you have questions about the assignment, contact the teaching assistants (TAs) by email to bdc-course@dei.unipd.it . The subject of the email must be "**HW1 - Group xx**", where xx is your ID group. If needed, a zoom/skype meeting between the TAs and the group will be organized.*

Last update: 29/03/2020 Back to home page