COSC–254 Data Mining
Homework 01 – Intro to Hadoop
Due: Wednesday, February 6, 2019, 1.59pm

The goal of this homework is to help you familiarize with Hadoop.

Before starting Verify that you are able to connect to the Hadoop cluster on campus (`hadoop2.amherst.edu`) via SSH or remote desktop. If you cannot connect, please email Matteo & Alex (`mriondato@`, `aeinarsson19@`) *as soon as possible*.

Exercise 0

- Let's get familiar with HDFS, the distributed filesystem of Hadoop.

  1. Download the file `http://rionda.to/courses/cosc-254-s19/assigs/hw01/wordcount.txt` to your home directory on the Hadoop cluster;
  2. List the contents of your HDFS user directory:

     ```
     $ hadoop fs -ls
     ```

     It should be empty, i.e., the command does not print anything on the screen.

     In case you get an error, try running

     ```
     $ kinit
     ```

     retype your password and try listing the contents again. If it still does not work, please email Matteo & Alex (`mriondato@`, `aeinarsson19@`) *as soon as possible*.
  3. Create a new subdirectory `wocoin` inside your HDFS user directory:

     ```
     $ hadoop fs -mkdir wocoin
     ```

     List the contents of your HDFS user directory again to make sure that the new directory exists.
  4. Copy the `wordcount.txt` file to the newly created directory:

     ```
     $ hadoop fs -copyFromLocal wordcount.txt wocoin/
     ```
  5. Print the content of the file that you just copied:

     ```
     $ hadoop fs -cat wocoin/wordcount.txt
     ```
  6. Copy back the `wordcount.txt` file from HDFS to a file `wordcount-copy.txt` on the *local* (i.e., non-HDFS) filesystem:

     ```
     $ hadoop fs -copyToLocal wocoin/wordcount.txt wordcount-copy.txt
     ```

     List the contents of your current directory to ensure that it contains the new file.
  7. Create an empty file in your HFDS user directory:

     ```
     $ hadoop fs -touchz empty
     ```

     List the content of the directory to ensure that the file was created.

8. Create an empty file `test` in your HFDS user directory:

   ```
   $ hadoop fs -touchz test
   ```

   List the content of the directory to ensure that the file was created.

9. Remove the empty file you just created:

   ```
   $ hadoop fs -rm empty
   ```

For a list of operations available on HDFS, see the output of `hadoop fs`.

You can also use the HDFS web GUI at `https://hadoop2.amherst.edu/filebrowser/`.

- The second part of this exercise will get you started with running jobs on Hadoop. Follow the Hadoop MapReduce tutorial at `http://bit.ly/dmmrt`, up to and including the WordCount 1.0 example (including the walk through). Make sure that you can follow the code in the WordCount example, as in the future you will have to write Hadoop code from scratch. Use the Hadoop API documentation at `http://bit.ly/dmmrad` to know more about each class being used. Copy the source code to your (local) home directory on the Hadoop cluster, compile, and run the example as shown in the tutorial using the `wordcount.txt` in the `wocoin` HDFS directory from Exercise 0 as input and a new HDFS directory `wocoout` as output directory. On the cluster, you don't need to (and should not) prefix the `hadoop` command with `bin/`. You can monitor the status of the jobs at `https://hadoop2.amherst.edu/jobbrowser/`.

Exercise 1 Write an Hadoop program which outputs, for each letter of the alphabet, the total number of (non-distinct) words in `wordcount.txt` that start with that letter. Ignore the letter case, i.e., consider all words as lower case. You can ignore all non-alphabetic characters.

Exercise 2 Modify the program from Exercise 1 to ignore all words in a list of *stopwords*. Download the list from `http://rionda.to/courses/cosc-254-s19/assigs/hw01/stopwords.txt`. You will have to use the Hadoop *DistributedCache* feature. The tutorial contains some information. A good example is available in the API doc for DistributedCache at `http://bit.ly/dmmrdc`.

Theoretically, you could perform the filtering of stopwords in either the Mapper, the Reducer, or the Combiner. Answer the following question (in a plain text or PDF file that you'll include in your submission): What is the communication complexity of each of these strategies in the worst case? Which one(s) should have the better communication complexity in a "normal" case ? You should obviously implement the one that is the least expensive in a "normal" case.

How to submit

Submit your work at `https://www.cs.amherst.edu/submit` as a *single* archive file with name `username.ext` where `username` is your user name and `ext` is one of `.zip`, `.tar.bz2`, or `.tar.gz`.

The archive must contain a *single* directory with name `username`. This directory must contain two subdirectories with names 1 and 2, respectively. All files (source code or otherwise) for each of the exercises must be in the directory with the same name. You can find an example archive at `http://rionda.to/courses/cosc-254-s19/assigs/hw01/mriondato.tar.gz`.

Please post to the Moodle forum if you have problems with the submission.