Exercises with Spark

The goal of this lab is to make you write Spark programs. This lab is particularly long, with exercises of increasing difficulty. you are not expected to finish it!

1 Opening Spark

There are several ways to use Spark. Don't spend too much time trying to use notebooks (when in doubt use Databricks)!

1.1 Using notebooks

You need to have a :
— a jupyter installation
— Spark and findspark installed.
— The correct value for `$SPARK_HOME` and `$PATH` variables (containing the path to your spark).
Then to create a Spark context you can use the following code :

```
import findspark
import pyspark
findspark.init()
sc = pyspark.SparkContext(appName="Spark Lab Session")
```

1.2 Using PySpark shell

You simply need to run the PySpark command and the shell will open. It is possible to open the shell in a notebook like interface see `https://www.sicara.ai/blog/2017-05-02-get-started-pyspark-jupyter-notebook-3-minutes`
The PySpark command can be launched on the Telecom machine `lame11.enst.fr` that can be accessed through ssh. Once logged in you need to use the following commands :

```
source /infres/ir510/hadoop/bin/hadoop_env
pyspark
```

1.3 Using Databricks

Go to `https://databricks.com/signup/signup-community` and register a new account, then create a cluster and open a workspace.

2 First steps with Spark

2.1 First RDD

DO: *Create a RDD containing the list L of the integers ranging from 0 to 2999 (both included).*
You can use the function `sc.parallelize(l)` to create an RDD containing the same elements as the list `l`.

2.2 Computing the sum of cubes

DO: *Using the RDD computed at the previous step, compute a second RDD representing C, the list of cubes of integers in L. In other words $C = \{0^3, 1^3, \ldots, 2999^3\}$. What is the sum of elements in C ?*

2.3 Last digits of elements in C

DO: *Compute an RDD containing a set of pairs* $(0, v_0), \ldots, (9, v_9)$ *where* v_i *is the number of integers in* C *having* i *for last digit.*

The last digit of an integer i can be computed using $(i\%10)$.

2.4 Digits of C

In Python, the list containing the digits of an integer can be computed using the following function :

```
def digits(i):
    return [e for e in str(i)]
```

DO: *Compute an RDD containing a set of pairs* $(0, v_0), \ldots, (9, v_9)$ *where* v_i *is the number of times the digit* i *appears in an integer of* C.

3 Approximating π

The number π can be defined as the area of a disk whose radius is 1 (using the formula $area = \pi R^2$ with $R = 1$).

3.1 Step 1 : computing set of pairs

To approximate π, you will first generate the list of all pairs (x,y) of integers ranging from 0 to $K - 1$ (both included), the greater the K the more precise the approximation but the more costly the computation therefore $K = 3000$ is a good compromise.

DO: *Compute an RDD containing all pairs* (x,y) *with* $0 \leq x, y < K$. *You should not materialize this list but you should start from the RDD computed in the previous section and use* `cartesian`.

3.2 Step 2 : counting the pairs

DO: *Compute the number of pairs* (x, y) *such that* $(2x+1)^2 + (2y+1)^2$ *is less then* $(2 * K)^2$. *How many pairs do you count for* $K = 3000$ *?*

3.3 Computing the approximation

The ratio between the number of pairs computed at the previous step and the total number of pairs is an approximation of $\pi/4$.

DO: *Compute the approximation of* π, *for* $K = 3000$ *you should obtain a value close to* 3.14159.

4 Using the Movie Lens dataset

4.1 Getting the dataset

The Movie Lens is a dataset in which users rate movies. It is already on the school HDFS at the address `/datasets/movie_small/` but if you work on your local machine you can find it here : `https://grouplens.org/datasets/movielens/`. For this lab session, I recommend to use the small dataset that can be downloaded here `http://files.grouplens.org/datasets/movielens/ml-latest-small.zip`.

4.2 Getting the dataset into an RDD

To read the dataset use the following code snippet :

```
import re
future_pattern = re.compile("""([^,"]+|"[^"]+")(?=,|$)""")

def parseCSV(line):
    return future_pattern.findall(line)

path_data = "/datasets/movie_small" # CHANGE ME
ratingsFile = sc.textFile(path_data+"/ratings.csv").map(parseCSV)
moviesFile = sc.textFile(path_data+"/movies.csv").map(parseCSV)
```

DO: *Copy the code above and run it. Make sure that the data is loaded and explore the data using the function* $.take(2)$ *on the RDDs.*

4.3 Cleaning data

Both RDDs (`ratingsFile` and `moviesFile`) contain the data from the CSV and columns have been split, however the data is still raw : all fields are strings and the header of the dataset is still here.

DO: *Using* `filter` *and* `map` *remove the header of the CSV file and cast the third column of* `ratingsFile` *to* `float` *create an RDD containing the ratings and movies.*

4.4 10 best movies of all times

The RDD corresponding to `ratings.csv` contains 4 columns, the first corresponds to the id of the user rating the movie, the second is the id of the movie being rated, the third column is the actual rating while the fourth is the timestamp of the rating.

DO: *Compute the 10 movieId that have the best average rating. Hint : use sortBy and take to select the best movies*

4.5 Ordered list of movies with names

The RDD corresponding to `movies.csv` contains 3 columns : the first is a movieId, the second is the title of the movie and the third describes the genre of the movie.

DO: *Using the RDD containing the average rating and a join operation with the movie RDD, compute an RDD containing the ordered list movies with their name and average rating. What are the names of the 10 best movies ?*

Note : Don't forget to `collect()` *the RDD to see the list!*

4.6 Better ordered list

The previous RDD was unsatisfactory because some movies made it to the top using only one 5-rating.

DO: *Compute this list again by using the two following metrics :*

$$\text{grade}(m) = \frac{\sum_{r \in \text{ratings}(m)} r}{|\text{ratings}(m) + 1|} \qquad \text{grade}(m) = \frac{\sum_{r \in \text{ratings}(m)} r}{|\text{ratings}(m)|} \times log(\text{ratings}(m))$$

5 Movie recommendation

This section explores the topic of movie reommendation. We will make recommendation for $userId = 1$.

DO: *Compute the set of movies that user* 1 *rated, The RDD should contain pairs of movie titles and ratings.*

5.1 Similarity

Let x and y be two users. We note :
— rated the set of movies rated by both users x and y
— $ratX$ the ratings of x on those movies
— $ratY$ the ratings of y on those movies
We then define the similarity between users x and y as :

$$\text{simil}(x,y) = \frac{\sum_{m \in \text{rated}} (\text{rating}(x,m) - \text{mean}(ratX)) \times (\text{rating}(y,m) - \text{mean}(ratY))}{(\text{variance}(ratX) \times \text{variance}(ratY))^{0.5}} \times log(1 + \text{rated})$$

The similarity is based on Pearson correlation coefficient modified to take into account the number of rated items (the $log(1+\text{rated})$ factor avoids spurious correlations). Note that for true recommendation we set $simil(x,x) = 0$ so that our recommendation system is not polluted by auto recommendation.

DO: *Compute an RDD containing pairs* (x, s) *where* x *is a userId and* $s = simil(x, 1)$.

DO: *What is simil*$(1,1)$ *? simil*$(1,2)$ *? simil*$(1,3)$ *? simil*$(1,4)$ *?*

5.2 Grade of a movie

The recommendation grade of a movie m for user 1 will be the average of ratings by other users ponderated by similarities between users. As in section 4.6, we don't want this list polluted by averages computed over very few values therefore, in practice we compute the following value :

$$\text{grade}(m) = \frac{\sum_{x \in \text{rated}(x)} \text{rating}(x,m) \times \text{simil}(x,1)}{\alpha + \sum_{x \in \text{rated}(x)} \text{simil}(x,1)}$$

where $\alpha \approx 0.5$ is the average value of $simil(x,1)$.

DO: *Compute an RDD containing pairs of movie id and their grade.*

DO: *Compute an RDD containing pairs of movie titles and their grade.*

5.3 Finishing

1. What are the best movies for user 1 ?
2. What are the best movies for user 1 that he/she has not rated ?
3. How your recommendation compare with the rating of user 1 ?