

CS-562 2024

1st Assignment

Exercise 0: Setting up the environment

-Nothing to submit here-

a) Spark can be installed quite simply on your PC following the next steps:

1. Download the latest pre-built version from here:

<http://spark.apache.org/downloads.html>

2. Unzip and move Spark to directory:

```
cd ~/Downloads/  
tar xzvf spark-2.2.0-bin-hadoop2.7.tgz mv spark-2.2.0-bin-hadoop2.7/ spark  
sudo mv spark/ /usr/lib/
```

3. Make sure Java is installed. If not install Java.

```
sudo apt-add-repository ppa:webupd8team/java  
sudo apt-get update  
sudo apt-get install oracle-java8-installer
```

4. Install SBT

```
echo "deb https://dl.bintray.com/sbt/debian/" | sudo tee -a /etc/apt/sources.list.d/sbt.list  
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv  
2EE0EA64E40A89B84B2DF73499E82A75642AC823  
sudo apt-get update  
sudo apt-get install sbt
```

5. Configure Spark.

```
cd /usr/lib/spark/conf/  
cp spark-env.sh.template spark-env.sh
```

→ Open spark-env.sh and add the following lines :

```
JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

```
SPARK_WORKER_MEMORY=4g
```

6. Configure .bashrc

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

```
export SBT_HOME=/usr/share/sbt-launcher-packaging/bin/sbt-launch.jar
```

```
export SPARK_HOME=/usr/lib/spark
```

```
export PATH=$PATH:$JAVA_HOME/bin
```

```
export PATH=$PATH:$SBT_HOME/bin:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

7. Now you are ready to interact with Spark:

type `/usr/lib/spark-shell` and start spark-shell

Get acquainted with Spark and run some simple tasks. Visit Spark's documentation (<http://spark.apache.org/documentation.html>) and watch Screencasts 3 & 4 of the Screencast Tutorial Videos under the Videos section.

b) In this section we are going to write simple Scala functions to get used to the language. In all the next exercises you must use only Scala to answer the questions.

1. Write a function that returns the second to last element of a list of integers.
e.g. `list=(1,2,3,4,5,6) → returns : 5`
2. Sum the values of a list of integers
3. Compute the max, min & average of a list of integers
4. Eliminate consecutive duplicates of a list elements. If a list contains repeated elements they should be replaced with a single copy of the element. The element order should not be changed.
e.g. `list=(1,2,3,3,5,5,8,10) → returns : (1,2,3,5,8,10)`

Download the following six works by Shakespeare from Project Gutenberg (total 22 MB):

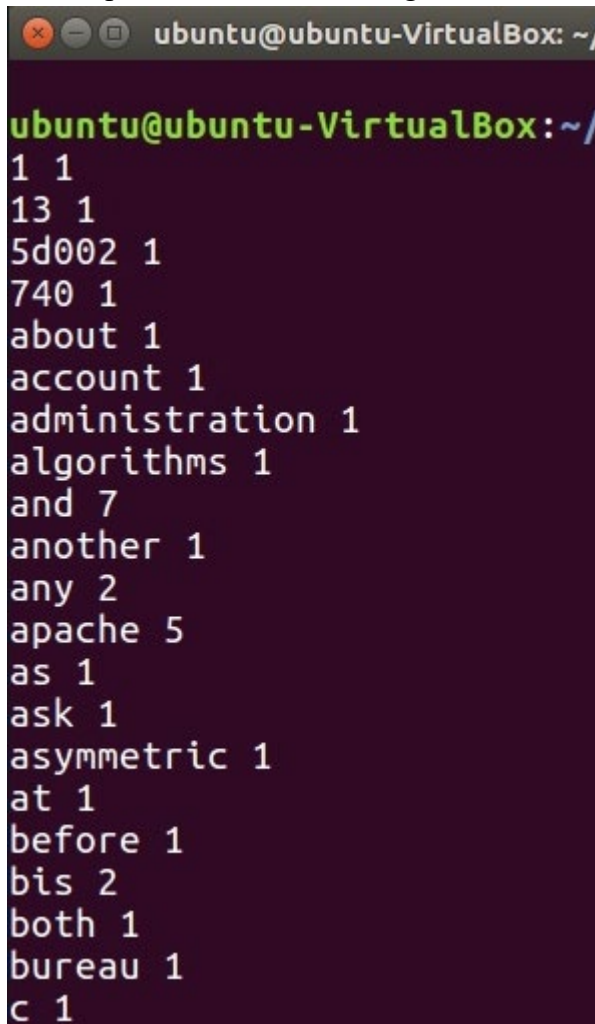
- pg100.txt <http://www.gutenberg.org/cache/epub/100/pg100.txt>
- pg31100.txt <http://www.gutenberg.org/cache/epub/31100/pg31100.txt>
- pg3200.txt <http://www.gutenberg.org/cache/epub/3200/pg3200.txt>
- pg2253.txt <http://www.gutenberg.org/cache/epub/2253/pg2253.txt>

- pg1513.txt <http://www.gutenberg.org/cache/epub/1513/pg1513.txt>
- pg1120.txt <http://www.gutenberg.org/cache/epub/1120/pg1120.txt>

Exercise 1 (5%) Word Count

The documents of the previous exercise form your document corpus for this assignment. The first exercise is a simple Word Count. What you have to do is load the files into your Spark Instance and try to count the occurrences of each unique word.

The output should be something like this:

A terminal window screenshot from an Ubuntu VirtualBox. The prompt is 'ubuntu@ubuntu-VirtualBox: ~/'. The output shows a list of words and their counts, sorted by count in descending order. The words and counts are: 1 1, 13 1, 5d002 1, 740 1, about 1, account 1, administration 1, algorithms 1, and 7, another 1, any 2, apache 5, as 1, ask 1, asymmetric 1, at 1, before 1, bis 2, both 1, bureau 1, c 1.

```
ubuntu@ubuntu-VirtualBox: ~/
1 1
13 1
5d002 1
740 1
about 1
account 1
administration 1
algorithms 1
and 7
another 1
any 2
apache 5
as 1
ask 1
asymmetric 1
at 1
before 1
bis 2
both 1
bureau 1
c 1
```

Exercise 2 (20%) Frequent Terms and Stop Words

The documents of the previous exercise form the your document corpus for this assignment. The second task is almost as easy as the word count. For the given set of documents as a collection of words, you are asked to collect all the stop words (https://en.wikipedia.org/wiki/Stop_words). Implement the needed Scala program to

identify the stop words (words with frequency > 4000 that are valid english words -remove non-alphanumeric terms-). For the given document corpus:

1. Print the k most frequent words among with their frequency (assume $k=10$ & 50).
2. Store **all the stopwords** in a single csv file using Scala code.
3. Find the average frequency of stopwords in each document.

Hints:

- Use as base the word count (Exercise 1)
- Ignore letter casing
- You can validate your StopWords list by comparing it with : <http://bit.ly/2hw8xx1>
- Use map() & reduceByKey() functions
- Use take() and sortBy()

Exercise 3 (30%)

A variation of an inverted index.

You are asked to implement an **inverted index**(<http://bit.ly/2wWhkKM>) for the document corpus described above. An inverted index provides for each distinct word in a document corpus, the filenames that contain this word, along with some other information (e.g., count/position within each document). For example, assume you are given the following corpus, consisting of doc1.txt, doc2.txt and doc3.txt:

doc1.txt: “This is a very useful program. It is also quite easy.”
 doc2.txt: “This is my first MapReduce program.”
 doc3.txt: “Its result is an inverted index.”

An inverted index would contain the following data (in random order):

<u>ID</u>	<u>Term</u>	<u>Document</u>
1	this	doc1.txt, doc2.txt
2	is	doc1.txt,doc2.txt,doc3.txt
3	a	doc1.txt
4	program	doc1.txt,doc2.txt
x

1. Implement a simple inverted index for the given document corpus, as shown in the previous Table, skipping the words of stopwords.csv.

- How many unique words exist in the document corpus (excluding stop words)? Which counter(s) reveal(s) this information? Define your own counter for the number of words appearing in a single document only. What is the value of this counter?

Hints:

- First you have to identify the unique terms and their documents, and then you merge the results.
- Experiment on different cases and select the most efficient.
- Use filter()

Exercise 4 (30%) An extension of an inverted index.

Extend the inverted index of Exercise 3, in order to keep the frequency of each word for each document. The new output should be of the form:

<u>ID</u>	<u>Term</u>	<u>Document #Frequency</u>
1	this	doc1.txt #1 , doc2.txt #1
2	is	doc1.txt #2, doc2.txt #2, doc3.txt #1
3	a	doc1.txt #1
4	program	doc1.txt #1 ,doc2.txt #1
x

which means that the word frequency should follow a single ‘#’ character, which should follow the filename, for each file that contains this word. Also, sort the document list based on the Frequency.

Exercise 5 (15%) Relative Frequencies.

Compute the relative frequencies of each word that occurs in the six documents in the previous corpus and output the top 100 word pairs sorted by decreasing order of relative frequency.

The relative frequency (RF) of word B given word A is defined as follows:

$$f(B | A) = \frac{\text{count}(A, B)}{\text{count}(A)} = \frac{\text{count}(A, B)}{\sum_{B'} \text{count}(A, B')}$$

where count(A,B) is the number of times A and B co-occur in a line and count(A) the number of times A occurs with anything else. Intuitively, given a document corpus, the relative

frequency captures the proportion of times the word B appears in the same line as A.

Submission (deadline: 18/10/2024):

Create a folder including all the .scala files you used and a PDF report for the answers. Send an email to hy562@csd.uoc.gr (**not the mailing list!**) with subject : **Assign1__StudentIDnumber.**