Exercise 2

Get started with Apache Spark and Python

Prior Knowledge
Unix Command Line Shell
Simple Python

Learning Objectives
Understand the Spark system
Understand the Jupyter Notebook model
Submit Spark jobs locally

Software Requirements

- Apache Spark 2.3.0
- Python 3.7
- Jupyter notebooks

We are going to do a wordcount against a set of books downloaded from Project Gutenberg. Wordcount is the definitive Big Data program (sort of Hello World for Big Data). We are going to be using Apache Spark's Python shell, in order to interactively test and run code.

1. You will probably first need to install Apache Spark. Download from here and store the unpacked version in your home directory
   http://archive.apache.org/dist/spark/spark-2.3.0/spark-2.3.0-bin-hadoop2.7.tgz

   For Spark to run you need to have Java 8 installed on your machine. It should work with more recent versions of Java but some people report problems.

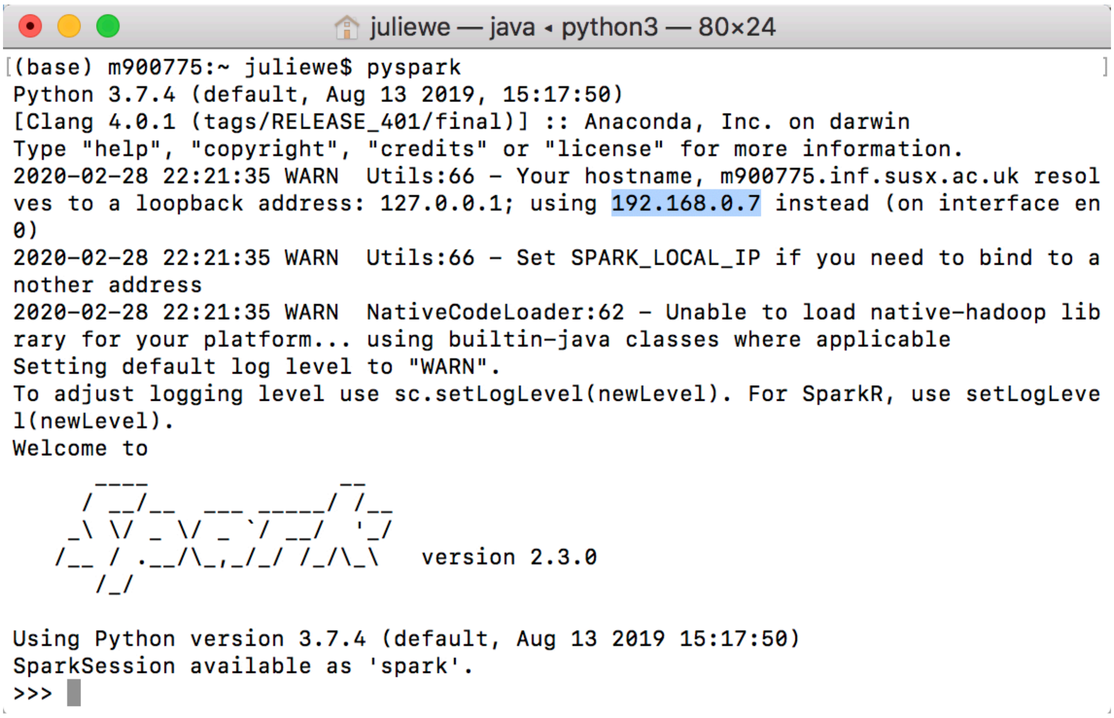
   Change .bash_profile variable settings. Add the following lines to your .bash_profile in your home directory

   ```
   export JAVA_HOME=$(/usr/libexec/java_home)
   export SPARK_HOME=~/spark-2.3.0-bin-hadoop2.7
   export PATH=$SPARK_HOME/bin:$PATH
   export PYSPARK_PYTHON=python3
   ```

   Run `source ~/.bash_profile` to source this file or open a new terminal to auto-source it.

   Now start the Spark Python command line tool by `typing pyspark`

You should see a lot of log come up, ending in something like:



```
(base) m900775:~ juliewe$ pyspark
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
2020-02-28 22:21:35 WARN  Utils:66 - Your hostname, m900775.inf.susx.ac.uk resol
ves to a loopback address: 127.0.0.1; using 192.168.0.7 instead (on interface en
0)
2020-02-28 22:21:35 WARN  Utils:66 - Set SPARK_LOCAL_IP if you need to bind to a
nother address
2020-02-28 22:21:35 WARN  NativeCodeLoader:62 - Unable to load native-hadoop lib
rary for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 3.7.4 (default, Aug 13 2019 15:17:50)
SparkSession available as 'spark'.
>>> 
```

2. If on mac or linux and you get an error on running pyspark, which is caused by a java.net.BindException, then you may need to update your hosts file as follows:
    a. Get your hostname using the `hostname` command.
    b. Use `sudo nano /etc/hosts` and add an entry to this file:

```
127.0.0.1       your_hostname
```

3. Once you have tested that Spark is installed correctly, you can quit the “traditional” Spark Python command line tool as we aren’t going to use this just now. Type `quit()` to leave

4. You will also need to install the python packages pyspark and findspark into your bigdata environment

```
conda activate bigdata
conda install pyspark
pip install findspark
```

5. Now, if you haven’t done so already, make sure you have an up-to-date version of the github repository for this course. At a terminal window:

```
git clone https://github.com/julieweeds/BigData.git
```

If you have already cloned the repository, you can ensure it is uptodate with:

```
cd BigData
git pull
```

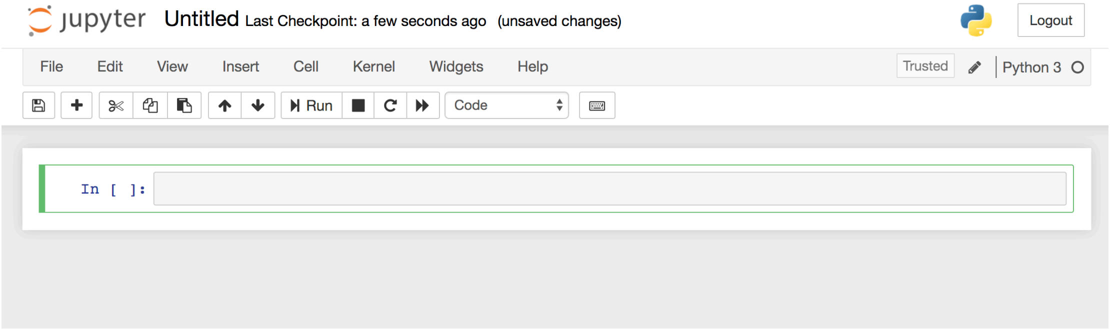
3. Let's make a directory to store our code.

```
mkdir wordcount
cd wordcount
```

4. We need some books to do a wordcount on. I have included some in the BigData repository. Let's make symbolic links to make it easier to access them (as if they were in the same directory)

```
ln -s ~/BigData/datafiles/books/* ./
```

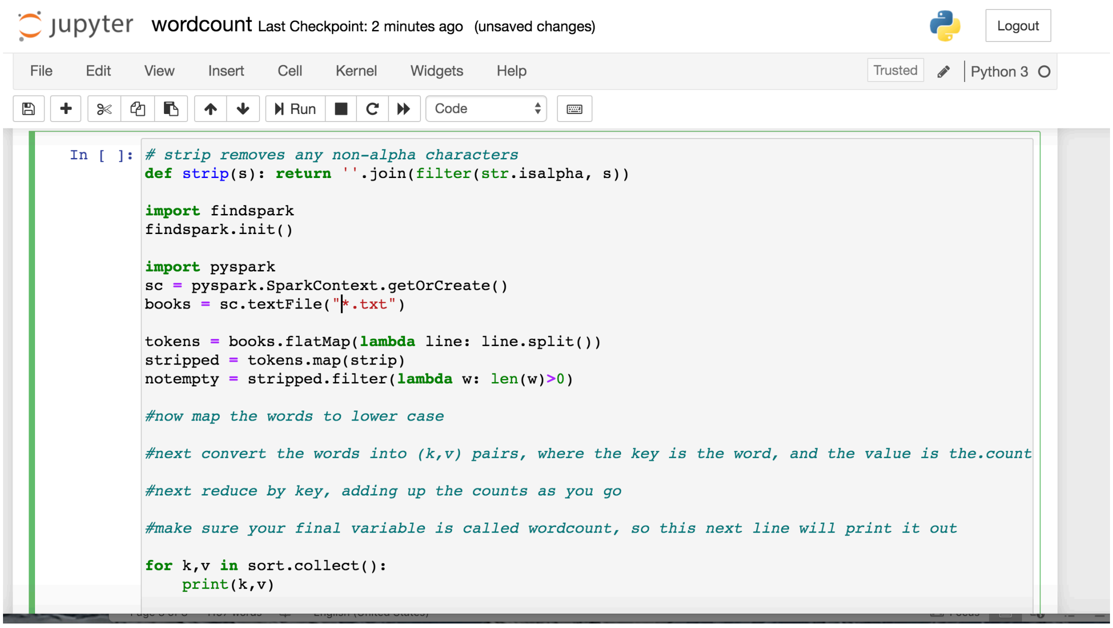
5. Start jupyter notebook and use the **New** button to create a new Python3 notebook:



5. Click on the name of the notebook (currently "Untitled") and rename it to "wordcount"
6. There is a starter of the code you need in github repository for this course. At the command line:

```
cat ~/BigData/code_jw/starters/wcnote.py
```

Paste the contents into the cell [1] so it looks like this:



```
# strip removes any non-alpha characters
def strip(s): return ''.join(filter(str.isalpha, s))

import findspark
findspark.init()

import pyspark
sc = pyspark.SparkContext.getOrCreate()
books = sc.textFile("*.txt")

tokens = books.flatMap(lambda line: line.split())
stripped = tokens.map(strip)
notempty = stripped.filter(lambda w: len(w)>0)

#now map the words to lower case

#next convert the words into (k,v) pairs, where the key is the word, and the value is the.count

#next reduce by key, adding up the counts as you go

#make sure your final variable is called wordcount, so this next line will print it out

for k,v in sort.collect():
    print(k,v)
```

7. There are some aspects that are not filled in that you need to write. Basically, this is a data-processing pipeline (also known as a directed acyclic graph)

   a. *Let's look at the parts that are there already.*

   b. We already have a SparkContext object defined in the notebook (in a program you need to define one, which we will see later)

   c. We define a strip function so that we can remove any non-alphanumeric characters:

   ```
   def strip(s): return ''.join(filter(str.isalpha, s))
   ```

   d. With the preliminaries over, the next line loads the data in:
   `books = sc.textFile("*.txt")`

   e. Then splits the lines into separate words

   ```
   tokens = books.flatMap(lambda line: line.split())
   ```

   f. Removes non-alpha characters by mapping the strip function onto each token in tokens
   `stripped = tokens.map(strip)`

   and removes empty items:
   `notempty = stripped.filter(lambda w: len(w)>0)`

8. Now it is time for you to do something!

   Convert all the words to lower case, using a map operation. In python, if *str* is a string, then `str.lower()` is the same string in lower case.

9. Now you need to get ready for a reduce. In order to do a reduce, we need some form of *key, value* pairs. I recommend using *tuples* which are simply (k,v) in Python (the brackets group the items into a tuple). Remembering how reduce works, we need to map each word to a count. Before reducing, that count is 1. So, we need a lambda that takes a word `w` and returns `(w,1)`

10. Now we can do a reduce that adds all those counts together. So that counts accumulated for each key, you can use the .reduceByKey() method.

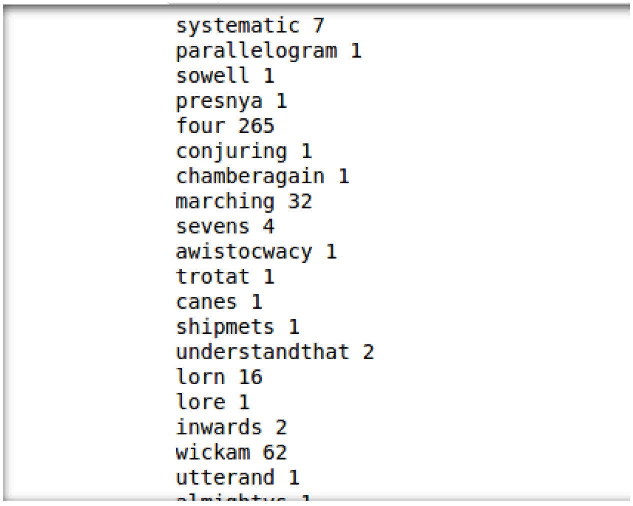
11. Finally, we need to collect the results and print them. In Spark, they may be distributed across different RDD partitions on different machines, so the collect() method brings them together.

```
for k,v in wordcount.collect(): print (k,v)
```

12. Try running the cell, by clicking ⏭

13. Be patient. I suggest you look at the command window and wait until you see spark start working.

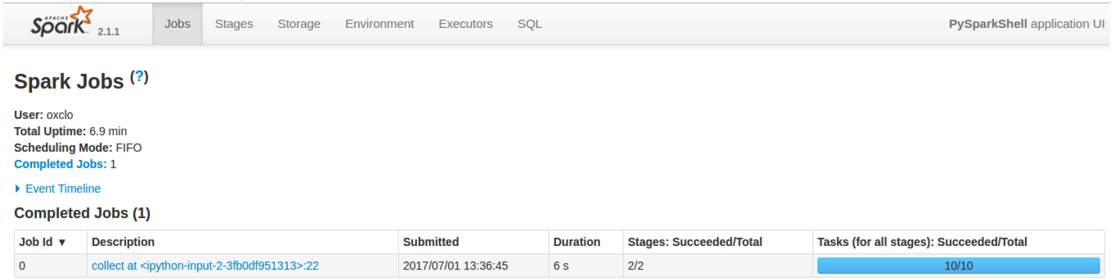
14. You should see a word count appear below cell 1:



```
systematic 7
parallelogram 1
sowell 1
presnya 1
four 265
conjuring 1
chamberagain 1
marching 32
sevens 4
awistocwacy 1
trotat 1
canes 1
shipmets 1
understandthat 2
lorn 16
lore 1
inwards 2
wickam 62
utterand 1
almightys 1
```

15. While the pyspark is still running browse to http://localhost:4040

16. You will see the Spark web console:



Spark Jobs (?)

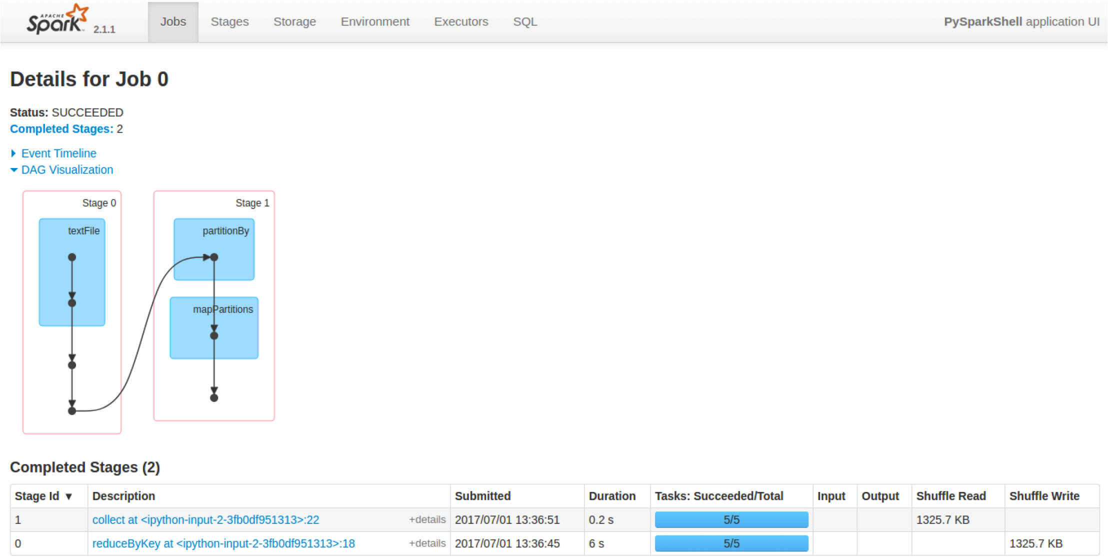
User: oxclo
Total Uptime: 6.9 min
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

Completed Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <ipython-input-2-3fb0df951313>:22	2017/07/01 13:36:45	6 s	2/2	10/10

17. Click on the blue link "collect at ipython-input"
This shows you how Spark converted your code into stages:



Details for Job 0

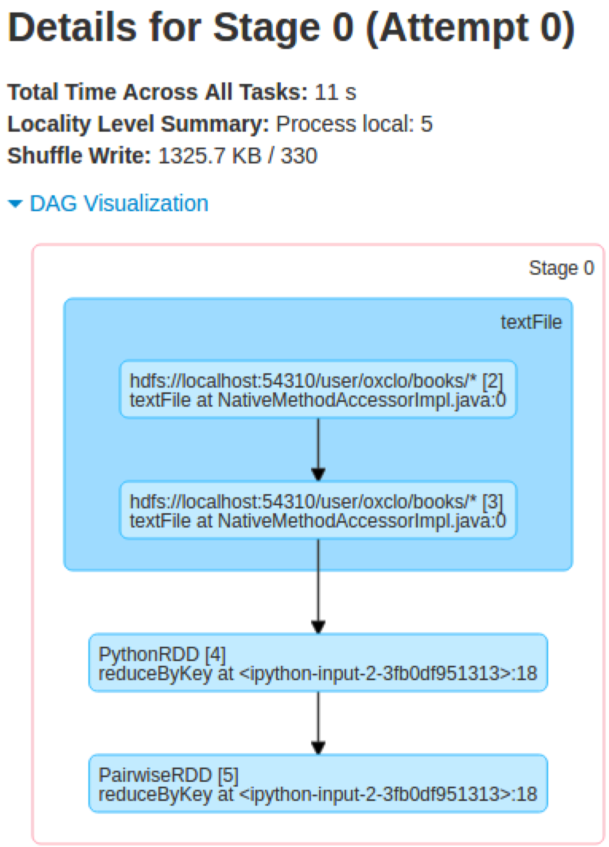
Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▾ DAG Visualization

Completed Stages (2)

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	collect at <ipython-input-2-3fb0df951313>:22 +details	2017/07/01 13:36:51	0.2 s	5/5			1325.7 KB	
0	reduceByKey at <ipython-input-2-3fb0df951313>:18 +details	2017/07/01 13:36:45	6 s	5/5				1325.7 KB

18. Click on Stage 0



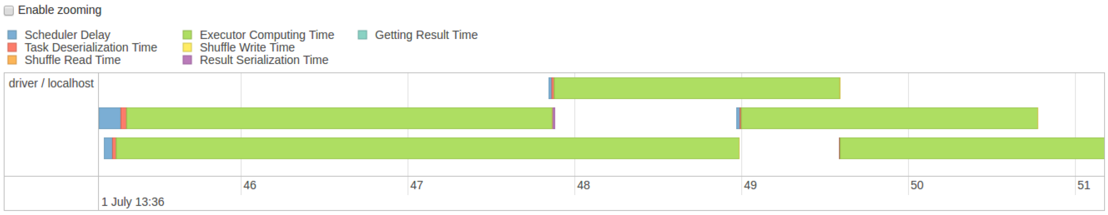
Details for Stage 0 (Attempt 0)

Total Time Across All Tasks: 11 s
Locality Level Summary: Process local: 5
Shuffle Write: 1325.7 KB / 330

▾ DAG Visualization

Stage 0
textFile
hdfs://localhost:54310/user/oxclo/books/* [2]
textFile at NativeMethodAccessorImpl.java:0
hdfs://localhost:54310/user/oxclo/books/* [3]
textFile at NativeMethodAccessorImpl.java:0
PythonRDD [4]
reduceByKey at <ipython-input-2-3fb0df951313>:18
PairwiseRDD [5]
reduceByKey at <ipython-input-2-3fb0df951313>:18

19. And expand the Event Timeline:



20. Make sure your code is saved from the notebook.

21. Quit the notebook shell by typing Ctrl-C on the command line, and then Y
Also close the notebook windows in the browser.

22. Now let's run the same code as a "job" instead of interactively.

23. Using a text editor, copy and paste that code into a file called
`bigData/wordcount/wc.py`

24. We run jobs locally on a single node directly on Spark:
The local[*] indicates to use as many threads as you have cores on your system:

```
spark-submit --master local[*] wc.py
```

25. You can send the output to a file

```
spark-submit --master local[*] wc.py > output.txt
```

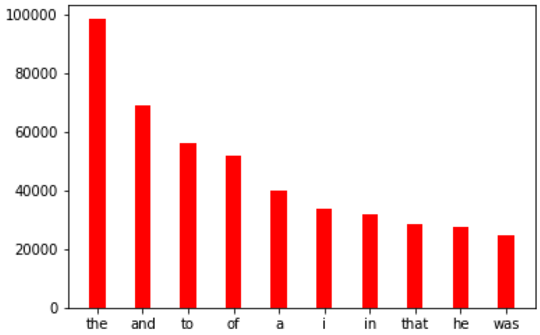
However, you will notiice that it is still hidden in all of the logging. Therefore, you will probably want to add a few lines of code to your python program to write the desired output to a file location rather than printing it to stdout.

26. **Congratulations, the lab is complete!**

Extensions

27. Re-load the code into the Jupyter notebook and now improve it to show the wordcount in descending order, starting with the most common words. How many instances of the word 'the' are there in the assembled books?

28. Have a look at matplotlib (https://matplotlib.org/users/intro.html) See if you can create the following graph of the top 10 most used words:



29. Adapt your code so that you produce a count of each character rather than of each word. Create a graph of the top 10 most used characters.