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
(see separate document for installation of these)

- Apache Spark 2.2.0
- Python 2.7.14
- 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. 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
```

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

```
mkdir ~/wordcount
cd ~/wordcount
```

3. 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/* ./
```

© Julie Weeds 2019 adapted from Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

4. Now start the Spark Python command line tool –

   `~/spark/bin/pyspark`

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

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
17/07/01 13:51:32 WARN NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
17/07/01 13:51:32 WARN Utils: Your hostname, oxclo resolves to a loopback
address: 127.0.0.1; using 172.16.64.199 instead (on interface ens33)
17/07/01 13:51:32 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
17/07/01 13:51:38 WARN ObjectStore: Failed to get database global_temp, returning
NoSuchObjectException
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.1.1
      /_/

Using Python version 2.7.12 (default, Nov 19 2016 06:48:10)
SparkSession available as 'spark'.
>>>
```

   b. This is the "traditional" Spark Python command line tool. We aren't going to use this just now.

   c. Type `quit()` to leave.

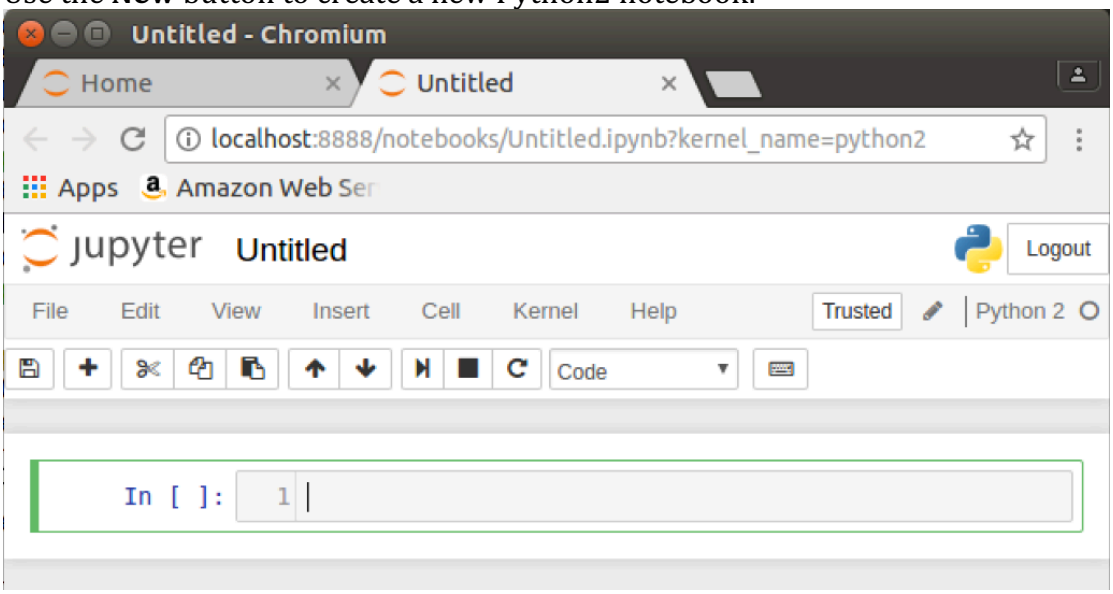
5. As we saw in the last exercise, the VM has a "notebook" system called Jupyter configured by default. The result is that instead of starting a command line repl[1], there is a web based editor / evaluator launched instead. To start this, type

   `jupyter notebook`

[1] Read Eval Print Loop

© Julie Weeds 2019 adapted from Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

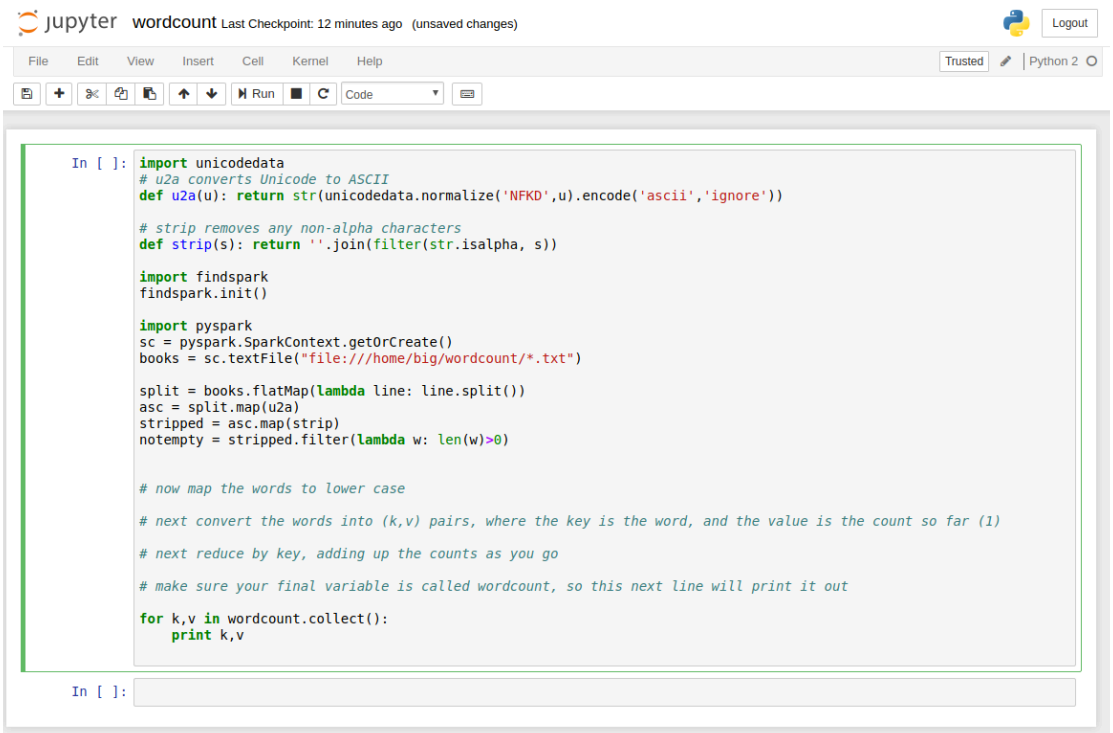
6. Use the **New** button to create a new Python2 notebook:



7. Click on the name of the notebook (currently "Untitled") and rename it to "wordcount"
8. 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:



© Julie Weeds 2019 adapted from Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

9. 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. Unfortunately some of the input is handled as Unicode by Python and we want to get rid of that.
    ```
    import unicodedata
    def u2a(u): return str(unicodedata.normalize('NFKD',u).
    encode('ascii','ignore'))
    ```

    d. We also want to remove any non-alphanumeric characters:

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

    e. With the preliminaries over, the next line loads the data in:
    ```
    books = sc.textFile("file://home/big/wordcount/*.txt")
    ```

    f. Then splits the lines into separate words

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

    g. Deals with the Unicode problem
    ```
    asc = split.map(u2a)
    ```

    And removes non-alpha characters
    ```
    stripped = asc.map(strip)
    ```

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

10. 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.

11. 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)`

12. 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.

© Julie Weeds 2019 adapted from Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

13. 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
```

14. Try running the cell, by clicking

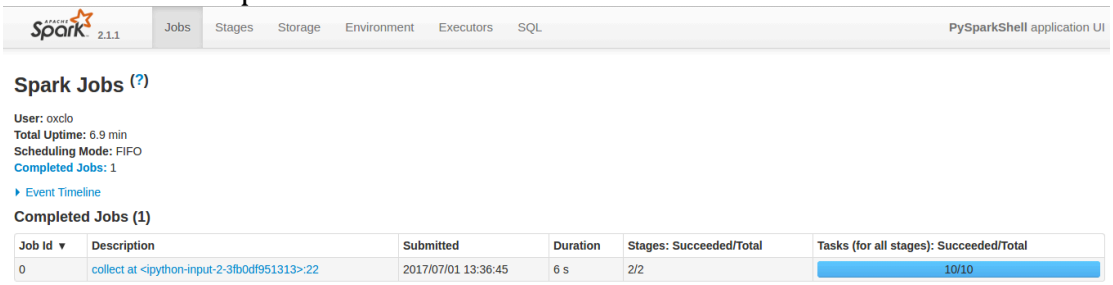
15. Be patient. I suggest you look at the command window and wait until you see spark start working.

16. 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
```

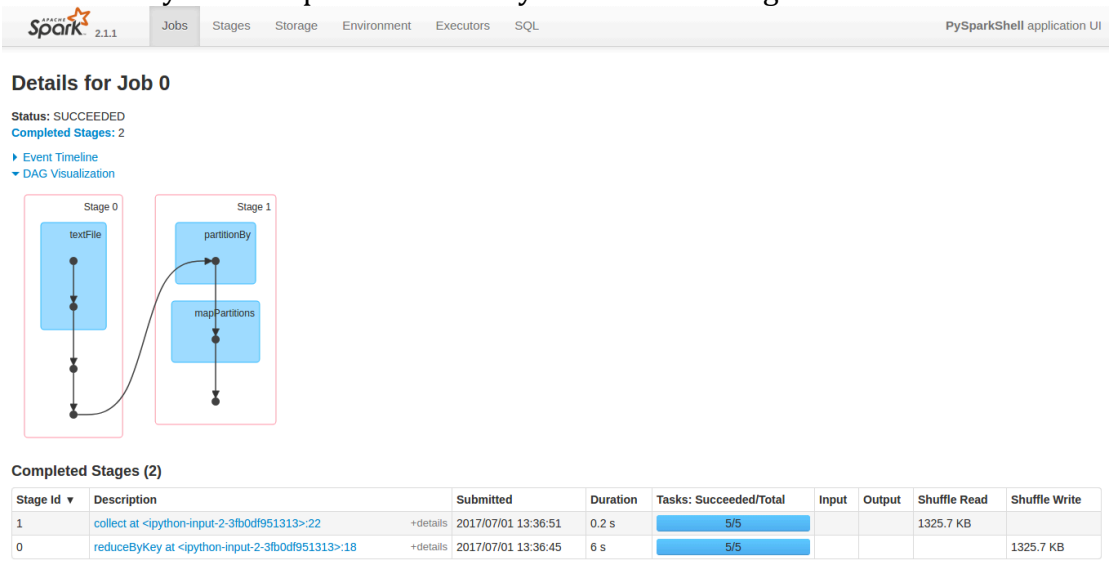
17. While the pyspark is still running browse to http://localhost:4040

18. You will see the Spark web console:

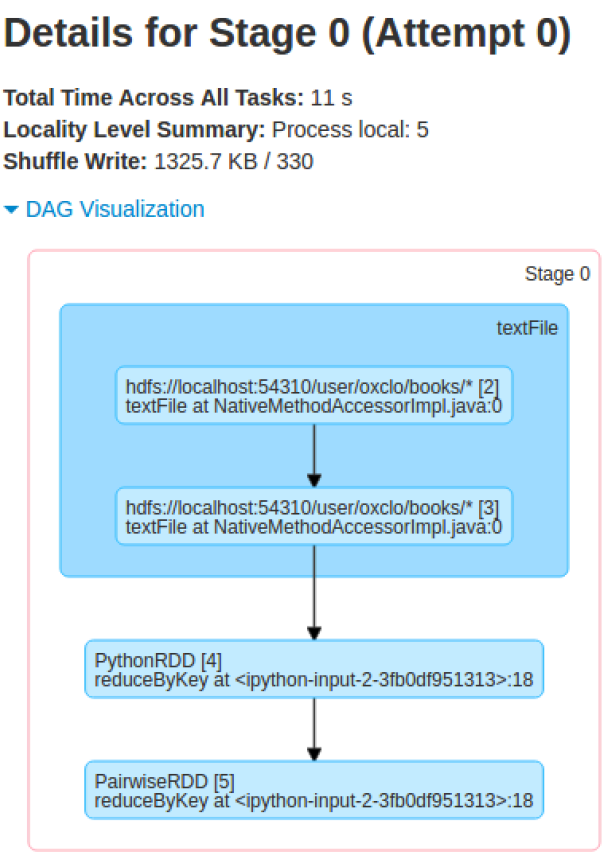


© Julie Weeds 2019 adapted from Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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

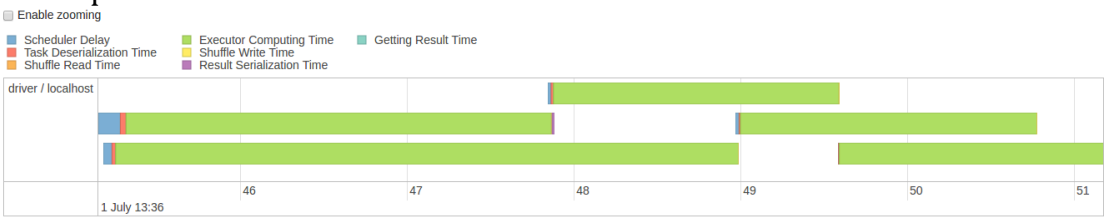


20. Click on Stage 0



© Julie Weeds 2019 adapted from Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

21. And expand the Event Timeline:



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

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

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

25. Using a text editor, copy and paste that code into a file called
    `/home/big/wordcount/wc.py`

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

27. You will notice that the output is masked by all the Spark logging. You can send the output to a file

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

28. Alternatively, you can hide the Spark logging and pipe the output into a useful utility called *less* that let's you page through it like this:

    ```
    ~/spark/bin/spark-submit --master local[*] wc.py 2> /dev/null | less
    ```

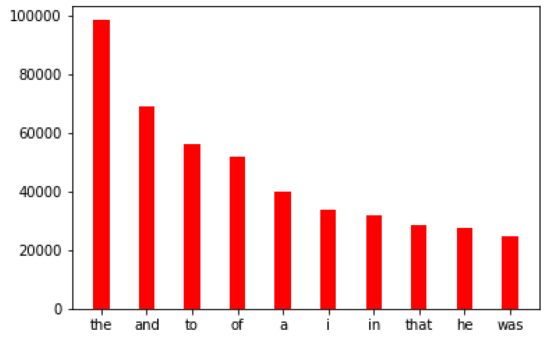
29. **Congratulations, the lab is complete!**

Extensions

30. 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?

© Julie Weeds 2019 adapted from Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

31. 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:



32. 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.

© Julie Weeds 2019 adapted from Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/