

Thomas Munoz Vasquez

Big Data Programming

Due March 3

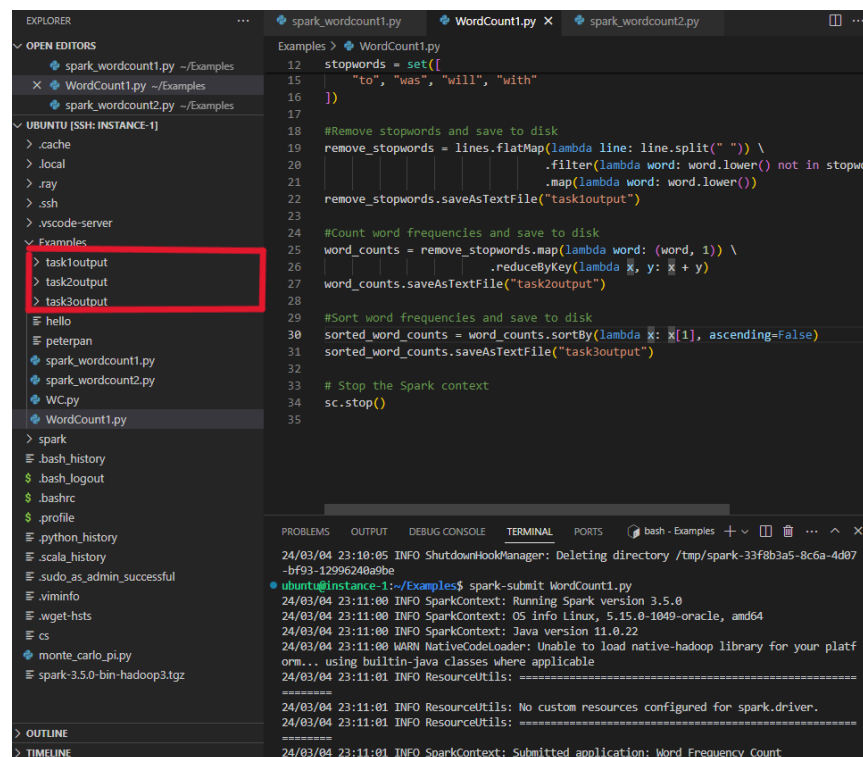
## Assignment 2

The objective of this code is to remove stopwords, count the frequency of words, and sort the words based on their frequency in descending order. It was achieved by using python and spark in a virtual machine.

To achieve the removal of stopwords, we input the file name and define the stopwords in a set of unique characters. We use flatmap to split each line of text into a list of words based on spaces, we follow with filter to filter out the stopwords from the file and finally we use map to ensure all words are converted to lowercase.

To count the frequency of words, we use map to transform each RRD to a key-value pair with the starting value of 1, we use reduceByKey to count the frequency (times it appears in file).

Finally, we use sortby to sort their frequency in descending order, it is sort in ascending order by default, so we set ascending to false.



```
EXPLORER
  OPEN EDITORS
    spark_wordcount1.py ~/Examples
    WordCount1.py ~/Examples
    spark_wordcount2.py ~/Examples
  UBUNTU [SSH: INSTANCE-1]
    .cache
    .local
    .ray
    .ssh
    .vscode-server
    Examples
      task1output
      task2output
      task3output
    hello
    peterpan
    spark_wordcount1.py
    spark_wordcount2.py
    WC.py
    WordCount1.py
    spark
    .bash_history
    .bash_logout
    .bashrc
    .profile
    python_history
    .scala_history
    sudo_as_admin_successful
    viminfo
    wget-hsts
    cs
    monte_carlo_pi.py
    spark-3.5.0-bin-hadoop3.tgz
  OUTLINE
  TIMELINE

Examples > WordCount1.py
12 stopwords = set([
15     "to", "was", "will", "with"
16 ])
17
18 #Remove stopwords and save to disk
19 remove_stopwords = lines.flatMap(lambda line: line.split(" ")) \
20     .filter(lambda word: word.lower() not in stopwords) \
21     .map(lambda word: word.lower())
22 remove_stopwords.saveAsTextFile("task1output")
23
24 #Count word frequencies and save to disk
25 word_counts = remove_stopwords.map(lambda word: (word, 1)) \
26     .reduceByKey(lambda x, y: x + y)
27 word_counts.saveAsTextFile("task2output")
28
29 #Sort word frequencies and save to disk
30 sorted_word_counts = word_counts.sortBy(lambda x: x[1], ascending=False)
31 sorted_word_counts.saveAsTextFile("task3output")
32
33 # Stop the Spark context
34 sc.stop()
35

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - Examples
24/03/04 23:10:05 INFO ShutdownHookManager: Deleting directory /tmp/spark-33f8b3a5-8c6a-4d07-bf92-12996240a9be
● ubuntu@instance-1:~/Examples$ spark-submit WordCount1.py
24/03/04 23:11:00 INFO SparkContext: Running Spark version 3.5.0
24/03/04 23:11:00 INFO SparkContext: OS info Linux, 5.15.0-1049-oracle, amd64
24/03/04 23:11:00 INFO SparkContext: Java version 11.0.22
24/03/04 23:11:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/03/04 23:11:01 INFO ResourceUtils: =====
24/03/04 23:11:01 INFO ResourceUtils: No custom resources configured for spark.driver.
24/03/04 23:11:01 INFO ResourceUtils: =====
24/03/04 23:11:01 INFO SparkContext: Submitted application: Word Frequency Count
```

Figure 1. Running command and creation of tasks outputs.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - Examples + - [ ] [ ] ... ^ x
24/03/04 23:11:29 INFO MemoryStore: MemoryStore cleaned
24/03/04 23:11:29 INFO BlockManager: BlockManager stopped
24/03/04 23:11:29 INFO BlockManagerMaster: BlockManagerMaster stopped
24/03/04 23:11:29 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommit
Coordinator stopped!
24/03/04 23:11:29 INFO SparkContext: Successfully stopped SparkContext
24/03/04 23:11:29 INFO ShutdownHookManager: Shutdown hook called
24/03/04 23:11:29 INFO ShutdownHookManager: Deleting directory /tmp/spark-e5ae0f11-f0f7-45db
-8350-41c05e673405
24/03/04 23:11:29 INFO ShutdownHookManager: Deleting directory /tmp/spark-c7ccd406-5294-4b35
-90b4-1025b35e9b66
24/03/04 23:11:29 INFO ShutdownHookManager: Deleting directory /tmp/spark-c7ccd406-5294-4b35
-90b4-1025b35e9b66/pyspark-211f4f52-cb3e-46ed-b33f-51fbf0b47ef9
```

Figure 2. Terminal output (runtime 29 secs)

```
EXPLORER ... spark_wordcount1.py WordCount1.py part-00000 X spark_wordcount2.py
OPEN EDITORS
  spark_wordcount1.py ~/Examples
  WordCount1.py ~/Examples
  X part-00000 ~/Examples/task1output
  spark_wordcount2.py ~/Examples
UBUNTU [SSH: INSTANCE-1]
  > .cache
  > .local
  > .ray
  > .ssh
  > .vscode-server
  > Examples
    > task1output
      .SUCCESS
      .SUCCESS.crc
      .part-00000.crc
      part-00000
    > task2output
      .SUCCESS
      .SUCCESS.crc
      .part-00000.crc
      part-00000
    > task3output
      hello
      peterpan
      spark_wordcount1.py
      spark_wordcount2.py
      WC.py
      WordCount1.py
    > spark
      .bash_history
      $ .bash_logout
      $ .bashrc
      $ .profile
      .python_history
    > OUTLINE
    > TIMELINE
Examples > task1output > part-00000
36859 start
36860 our
36861 website
36862 which
36863 has
36864 main
36865 pg
36866 search
36867 facility:
36868 www.gutenberg.org.
36869
36870 website
36871 includes
36872 information
36873 about
36874 project
36875 gutenberg",
36876 including
36877 how
36878 make
36879 donations
36880 project
36881 gutenberg
36882 literary
36883 archive
36884 foundation,
36885 how
36886 help
36887 produce
36888 our
36889 new
36890 ebooks,
36891 how
36892 subscribe
36893 our
36894 email
36895 newsletter
36896 hear
36897 about
36898 new
36899 ebooks.
36900
```

Figure 3. Task1Output (remove stopwords).

The screenshot shows the VS Code interface with the Explorer panel on the left and the Output panel on the right. The Explorer panel shows the file structure of the project, including the 'part-00000' file. The Output panel displays the output of the 'task2output' task, which is a list of word frequencies in the format (word, count). The output is as follows:

```

1 ('project', 83)
2 ('gutenberg', 25)
3 ('ebook', 8)
4 ('peter', 241)
5 ('pan', 16)
6 ('', 2186)
7 ('use', 22)
8 ('anyone', 4)
9 ('anywhere', 2)
10 ('united', 15)
11 ('states', 11)
12 ('most', 45)
13 ('other', 73)
14 ('parts', 4)
15 ('world', 5)
16 ('cost', 2)
17 ('almost', 37)
18 ('restrictions', 2)
19 ('whatsoever', 2)
20 ('you', 450)
21 ('may', 48)
22 ('copy', 8)
23 ('it', 63)
24 ('give', 27)
25 ('away', 30)
26 ('re-use', 2)
27 ('under', 31)
28 ('terms', 23)
29 ('license', 12)
30 ('included', 2)
31 ('online', 4)
32 ('www.gutenberg.org', 4)
33 ('located', 7)
34 ('states', 4)
35 ('have', 251)
36 ('check', 4)
37 ('laws', 11)
38 ('country', 5)
39 ('where', 46)
40 ('before', 44)
41 ('using', 8)
42 ('ebook', 8)

```

Figure 4. Task2output ( count word frequencies<key/value>.)

The screenshot shows the VS Code interface with the Explorer panel on the left and the Output panel on the right. The Explorer panel shows the file structure of the project, including the 'part-00000' file. The Output panel displays the output of the 'task3output' task, which is a list of sorted word frequencies. The output is as follows:

```

1 ('', 2186)
2 ('he', 1029)
3 ('she', 574)
4 ('had', 504)
5 ('his', 471)
6 ('you', 450)
7 ('her', 370)
8 ('i', 253)
9 ('have', 251)
10 ('were', 243)
11 ('peter', 241)
12 ('all', 236)
13 ('said', 218)
14 ('so', 215)
15 ('would', 212)
16 ('wendy', 201)
17 ('him', 186)
18 ('one', 178)
19 ('when', 172)
20 ('them', 165)
21 ('i', 159)
22 ('we', 154)
23 ('from', 147)
24 ('could', 139)
25 ('been', 135)
26 ('who', 134)
27 ('which', 128)
28 ('what', 124)
29 ('did', 117)
30 ('up', 115)
31 ('out', 115)
32 ('do', 110)
33 ('said', 108)
34 ('about', 107)
35 ('little', 95)
36 ('darling', 94)
37 ('hook', 94)
38 ('now', 93)
39 ('see', 88)
40 ('like', 86)
41 ('only', 86)
42 ('project', 83)

```

Figure 5. Task3output (Sorted word frequencies).

