Homework #3 – Spark

Books Used:

AdvofSherlockHolmes.txt, AliceInWonderland.txt from HW#2

Basic Word Count and Extended Word Count have been implemented and was run on the above books. Jupyter Notebook was used to achieve this and the output to this is saved in Output folder. The output file was then used to print the 25 most common words which is in the ipynb file.

Analysis:

1) In the PySpark REPL/ Jupyter notebook , run your basic word count program on a single text file.
   a. What are the 25 most common words? Include a screenshot of program output to back-up your claim.

OUTPUT: Top 25 common words in Sherlock Holmes and Alice in Wonderland.

```
Top 25 Common Words in Sherlock Holmes:
the: 5412
and: 2794
of: 2724
to: 2702
a: 2575
I: 2533
in: 1706
that: 1557
was: 1361
his: 1096
is: 1073
you: 1034
he: 1014
it: 977
my: 901
have: 894
with: 843
had: 807
as: 776
which: 753
at: 737
for: 698
be: 610
not: 605
from: 485

Top 25 words in Alice in Wonderland:
the: 1683
and: 782
to: 778
a: 667
of: 604
she: 485
said: 416
in: 406
it: 357
was: 329
you: 306
I: 249
as: 246
that: 225
Alice: 221
with: 214
at: 209
her: 204
had: 176
all: 168
be: 153
on: 148
for: 147
or: 136
not: 129
```

Top 25 common words was done in Jupyter Notebook itself since there was no mention about where we should output. Refer to the ipynb file for full code and more information. The action was performed on both text files (books) since the instructions were not clear if the output should be from both books combined or just one.

b. How many stages is execution broken up into? Explain why. Include a screenshot of the DAG visualization from Spark's WebUI to back-up your claim.

Answer:

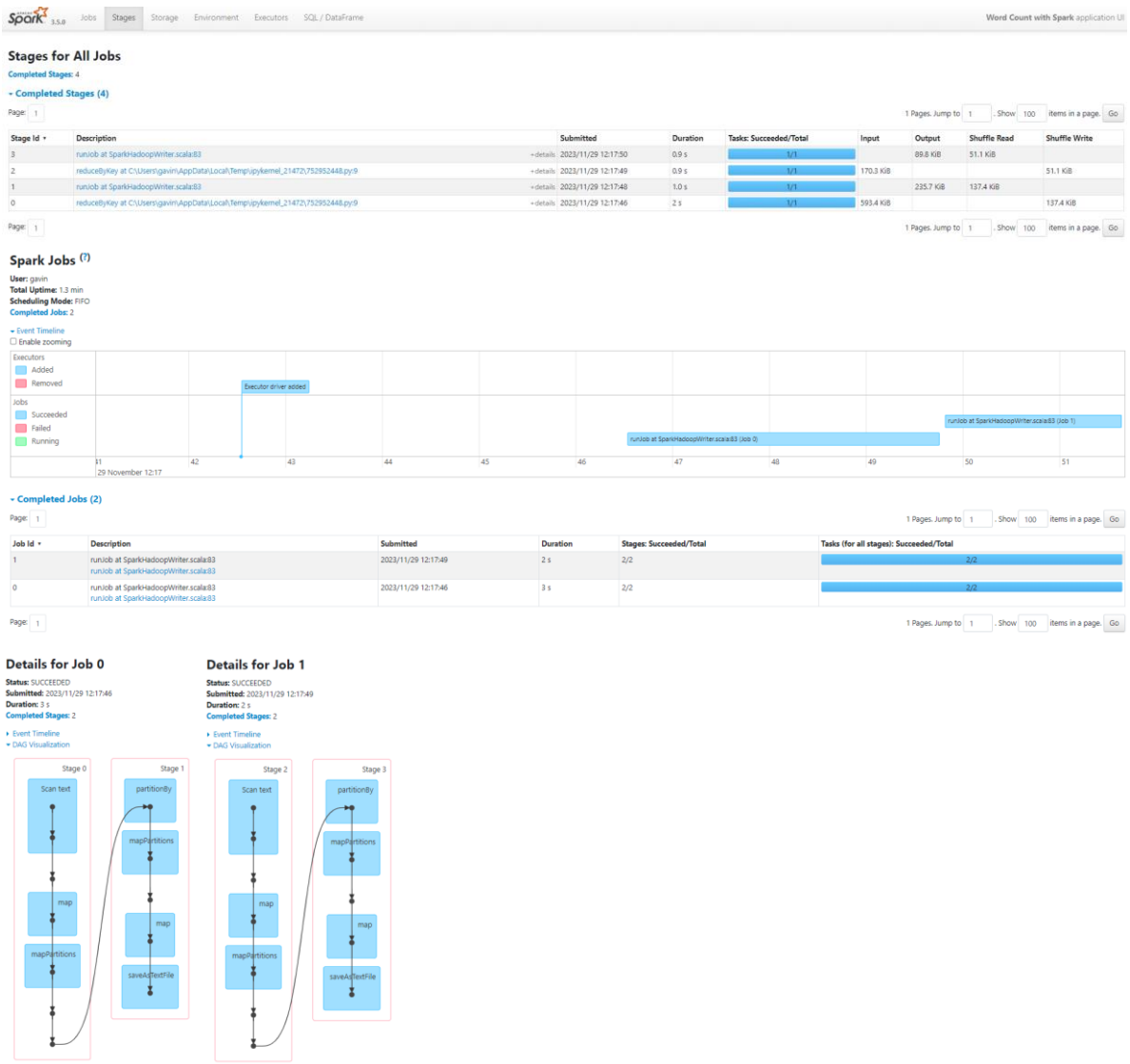
There is Job 0 and Job 1 because the same action was performed in both books. The execution of each Job is broken up into two stages to manage data shuffling and redistribution necessary for certain transformations, like reduceByKey, which aggregates data by key. Below is the explanation of 2 stages.

Stage 0:

This stage includes reading the text file (Scan Text), which reads. The (Map) is applied in each line of the input to process the data. The (MapPartitions) applies to each partition of the input data allowing more complex operation to be performed independently on each partition.

Stage 1:

It separated from Stage 0 by a shuffle boundary (partitionBy). When data needs to be redistributed among different executors, a shuffle happens. This normally happens after a transformation such as reduceByKey, that is used for combining values from different partitions with the same key. mapPartitions and map helps with formatting before saveAsTextFile which writes/saves to the file system. Thus, the need to regroup data across the cluster to perform aggregations is why the partition into stages is necessary.



2) In the PySpark REPL/Jupyter notebook, run your word count extended program on all 2 text files.
   a. What are the 25 most common words? Include a screenshot of program output to back-up your claim.
   OUTPUT: Top 25 common words in Sherlock Holmes and Alice in Wonderland.

```
Top 25 Words in Sherlock Holmes:
('holmes', 466)
('man', 305)
('room', 171)
('time', 151)
('door', 144)
('good', 137)
('face', 128)
('matter', 125)
('hand', 120)
('house', 120)
('night', 118)
('case', 117)
('heard', 113)
('day', 107)
('sherlock', 101)
('morning', 101)
('gutenberg', 98)
('work', 92)
('left', 92)
('project', 89)
('long', 88)
('asked', 88)
('eyes', 87)
('street', 83)
('father', 83)

 Top 25 Words in Alice in Wonderland:
('alice', 402)
('gutenberg', 97)
('project', 88)
('queen', 76)
('thought', 74)
('time', 71)
('king', 63)
('turtle', 59)
('mock', 57)
('began', 57)
('hatter', 56)
('gryphon', 55)
('rabbit', 53)
('work', 53)
('head', 50)
('thing', 49)
('voice', 48)
('1', 46)
('looked', 45)
('mouse', 44)
('duchess', 42)
('round', 41)
('tone', 40)
('dormouse', 40)
('great', 39)
```

Top 25 common words was done in Jupyter Notebook itself since there was no mention about where we should output. Refer to the ipynb file for full code and more information. The action was performed on both text files (books) since the instructions were not clear if the output should be from both books combined or each separately.

b. How many stages is execution broken up into? Explain why. Include a screenshot of the DAG visualization from Spark's WebUI to back-up your claim.

Answer:

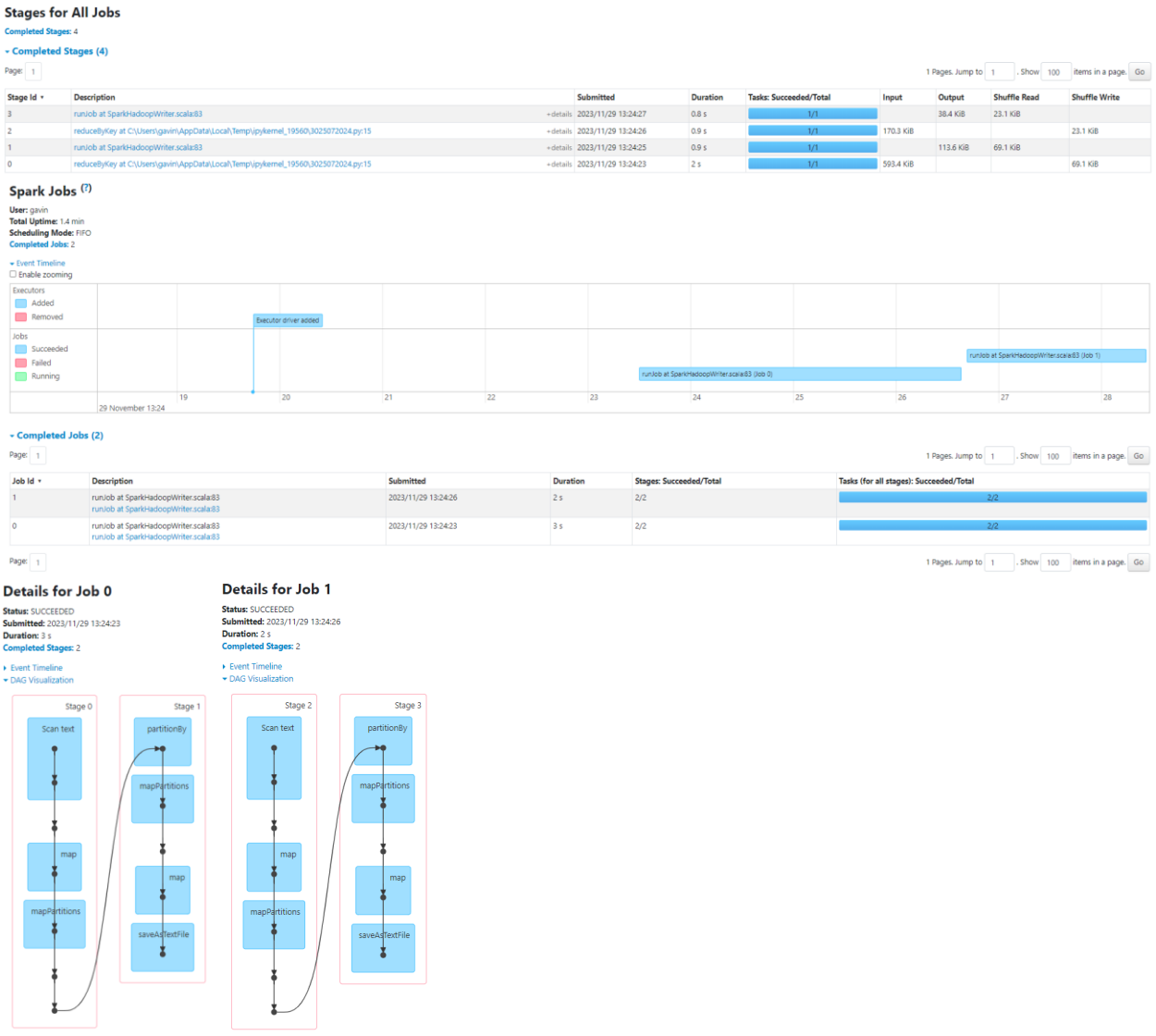
There is Job 0 and Job 1 because the same action was performed in both books. The execution of each Job in Extended Word Count is also broken up into two stages due to the transformations that cause a shuffle of data across the cluster. Below is the explanation of 2 stages.

Stage 0:

In this stage, the text file is read, and a map function is applied to every line converting it to lowercase and removing punctuation. Then, for each partition the MapPartitions function is used to filter out the stop words from the given StopWordsList and counting individual words within each partition.

Stage 1:

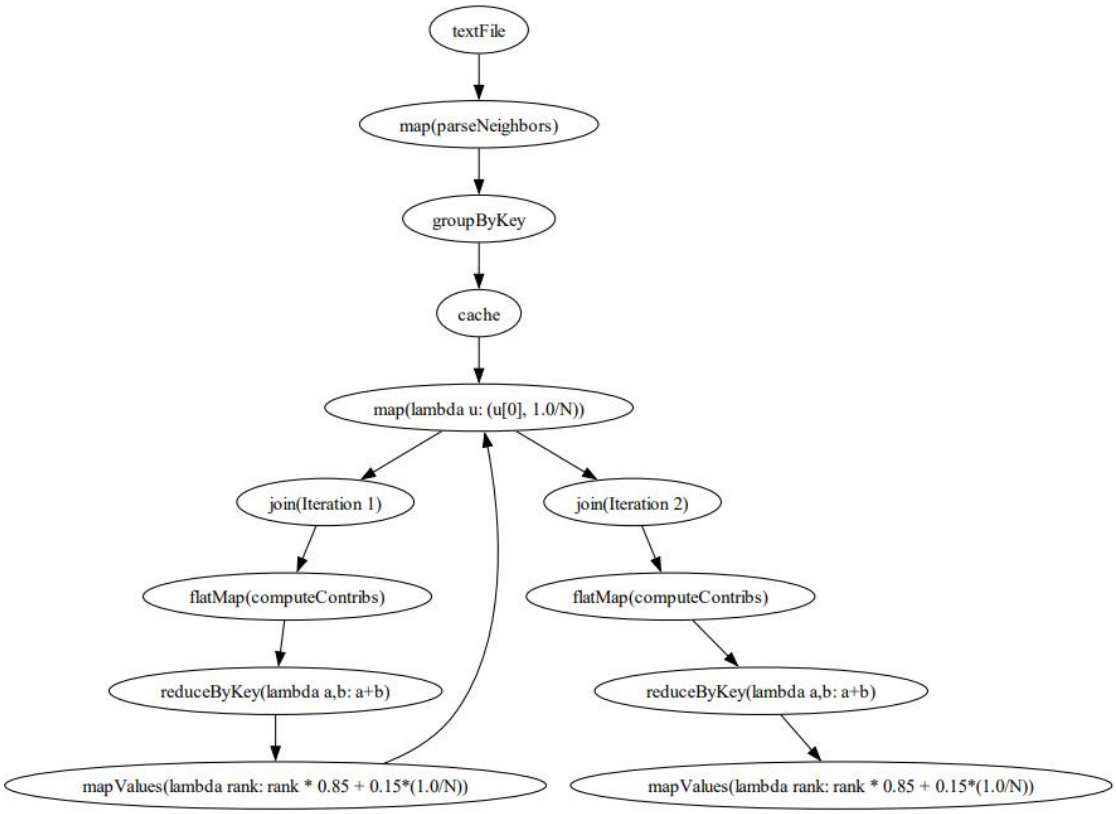
It separated from Stage 0 by a shuffle boundary caused by reduceByKey. The shuffle collects word count from all partitions by redistributing data around cluster based on keys(words). The data is then formatted by mapPartions and map before the aggregated word counts are saved into the filesystem by saveAsTextFile. Similar to the Basic Word Count, the partitioning into stages is needed to organize the workflow, especially for shuffling between stages to group key-value pairs for the reduction and aggregation.



3. Given the above spark application, draw the lineage graph DAG for the RDD ranks on line 12 when the iteration variable i has a value of 2. Include nodes for all intermediate RDDs, even if they are unnamed.

Answer:

Below is the linear graph DAG for iteration variable i of value 2. It includes nodes for all intermediate RDDs.



P.S. Although the count() action also produces an RDD, it is removed from the graph because it is an action rather than a transformation that adds to the ranks RDD's lineage. Hence, the graph does not directly display the count() action.