CS5425 Assignment 2 Task 1 Report by A0212221E – Vincent Andrian Chandra

Task 1 contains 3 main steps:

1. Count words for each file
2. Search for common words
3. Results

1. Count Words

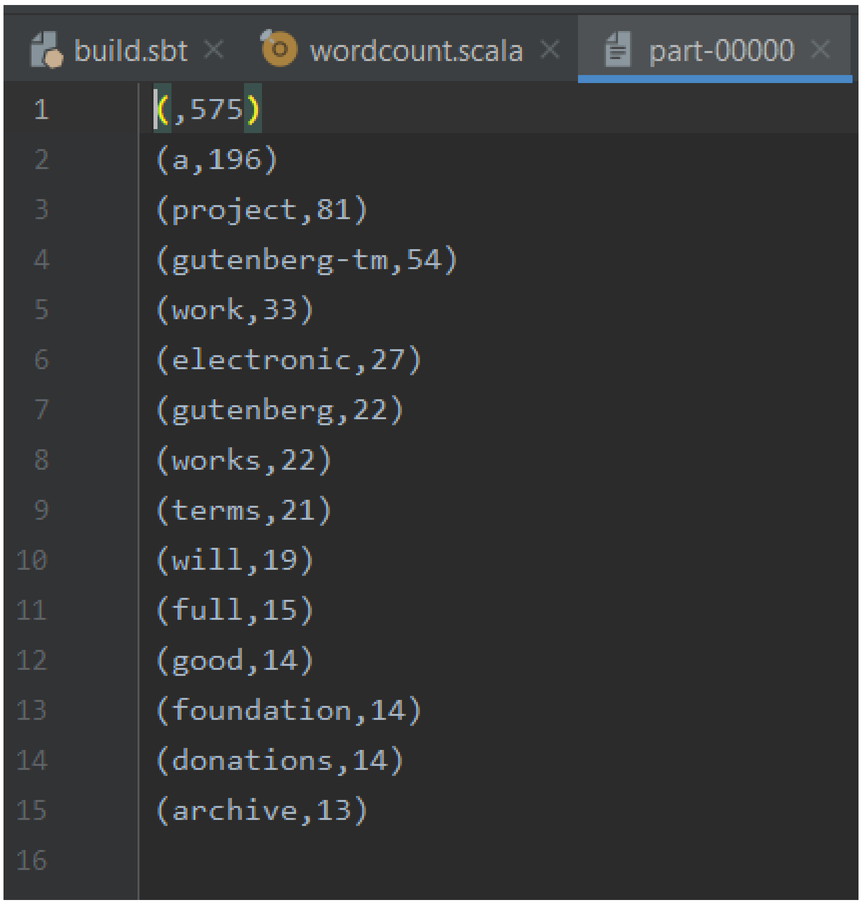
The first step is to read both files and stopword file. Stopwords are then stored in an array which will be used to filter out the stopwords on both files. After filtering, a mapreduce function similar to the one in Hadoop is called to count the number of words grouped by key (words).

2. Get Common Words

File 2 is joined with file 2 and will result in one word with 2 kind of counts. Then mapping is done with a condition that only words that exist in both files and the lower count among 2 files are taken. Finally, the mapped result is sorted in descending order and output only the top 15 result.

3. Result

Top 15 words are displayed and they are sorted in descending manner according to the count. However, empty space also gets counted as seen on the first row of the result. I have tried various delimiter for splitting the line but none of them removes it.



```
(,575)
(a,196)
(project,81)
(gutenberg-tm,54)
(work,33)
(electronic,27)
(gutenberg,22)
(works,22)
(terms,21)
(will,19)
(full,15)
(good,14)
(foundation,14)
(donations,14)
(archive,13)
```

3.1 Programming Comparison between Hadoop and Spark

Programming common words in Spark is much more efficient, resulting in much fewer lines than programming in Hadoop. Spark provides Mapreduce API similar to the one in Hadoop, but the former offers a lot more API than just Mapreduce which is very helpful to process the data. The only small downside to that is there is a need to learn and search for API that suits our need, rather than just program it purely using the language just like in Hadoop.

3.2 Runtime Execution Comparison between Hadoop and Spark

Performance wise, Spark has the upper hand because the read and write I/O only happens at the beginning when reading the files and writing the result at the end. On the other hand, Hadoop needs to have 4 jobs just to execute the same task (wordcount, common words, sort), which means more read and write network and disk I/O. Since all processing happens in memory, it is a lot faster than Mapreduce in Hadoop where the process mostly happens on disk.