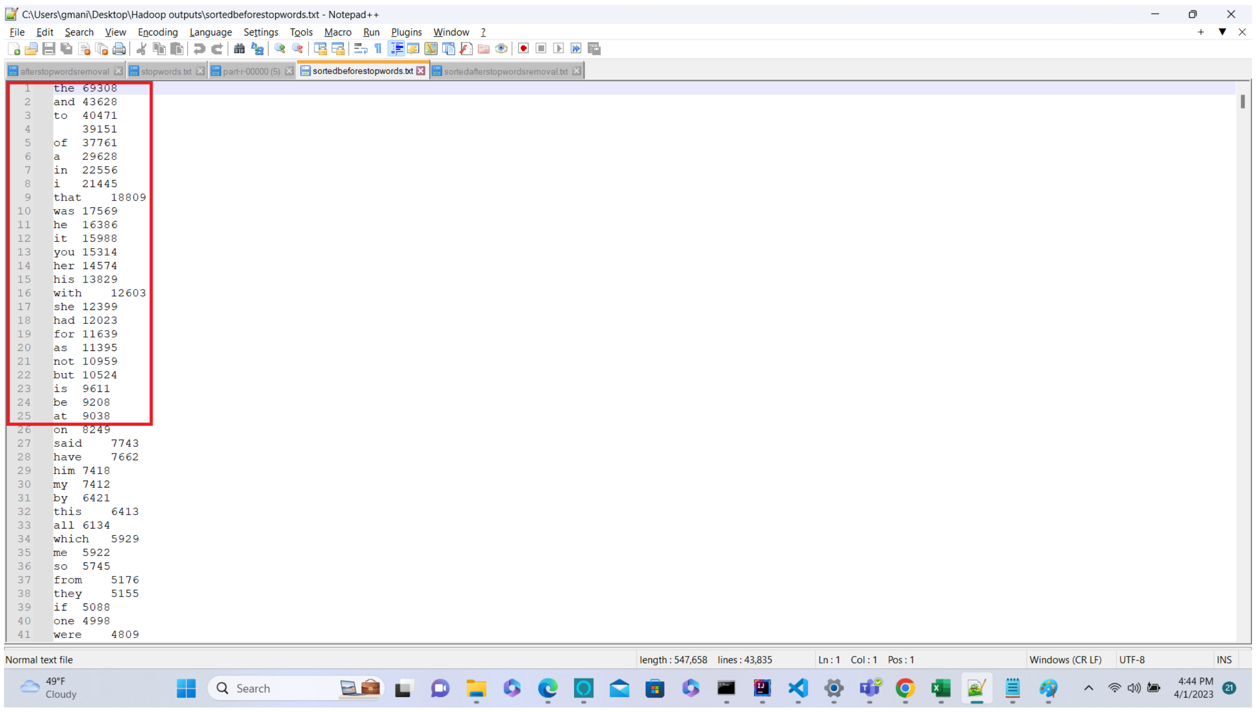
CSE 4/587 Assignment 1 - Spring 2023

Manikanta Kalyan Gokavarapu
50465129
mgokavar@buffalo.edu

1. What are the 25 most common words and the number of occurrences of each when you do not remove stopwords?

Sorted result of my WordCount application without removing stop words.

Below are the 25 most common words and their number of occurrences of each, before removing the stop key words.



```
the	69308
and	43628
to	40471
	39151
of	37761
a	29628
in	22556
i	21445
that	18809
was	17569
he	16386
it	15988
you	15314
her	14574
his	13829
with	12603
she	12399
had	12023
for	11639
as	11395
not	10959
but	10524
is	9611
be	9208
at	9038
on	8249
said	7743
have	7662
him	7418
my	7412
by	6421
this	6413
all	6134
which	5929
me	5922
so	5745
from	5176
they	5155
if	5088
one	4998
were	4809
```

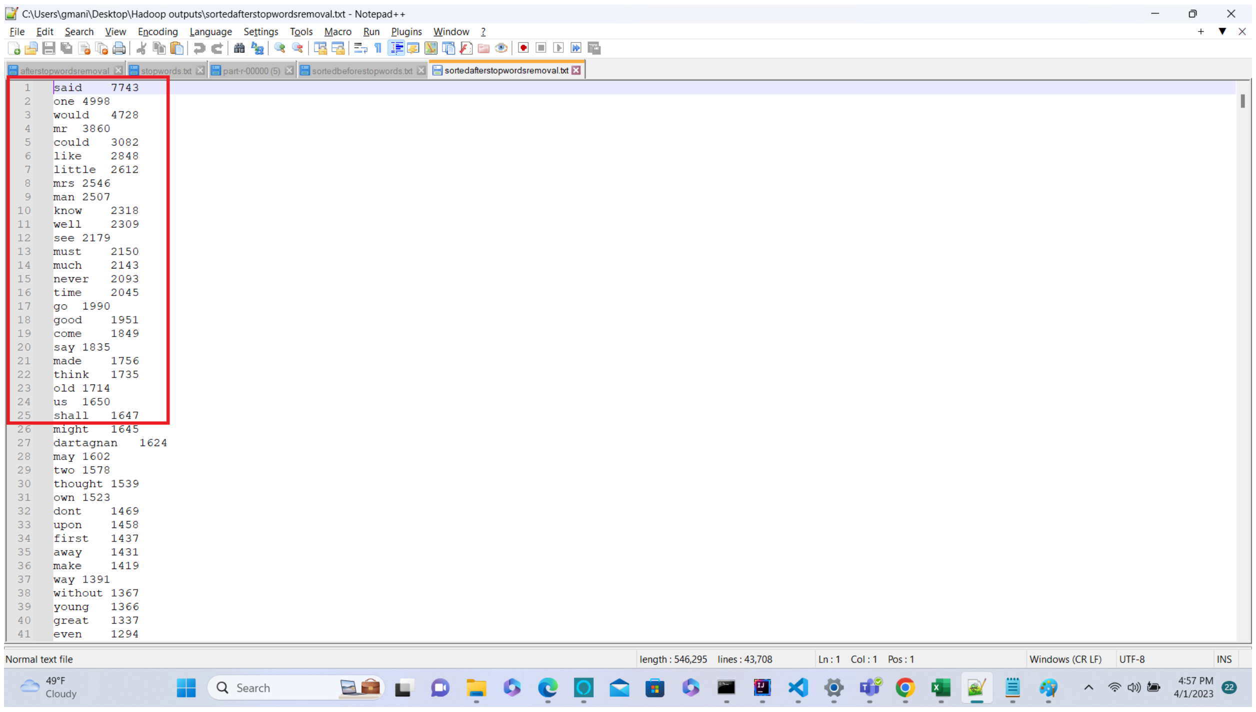
Info on how I achieved the above results:
Generally the mapper outputs i.e key value pairs are fed into the reducer logic of the code and after that I have used command line sort to sort the data based on values and saved it to a text file.

Reducer code: (just for reference)

```
static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {

   private IntWritable result = new IntWritable();

   public void reduce(Text key, Iterable<IntWritable> values, Context context)
           throws IOException, InterruptedException {
       int sum = 0;
       for (IntWritable val : values) {
           sum += val.get();
       }
       result.set(sum);
       context.write(key, result);
   }

}
```

2. What are the 25 most common words and the number of occurrences of each when you do remove stopwords?

Sorted result of my WordCount application after removing stop words.

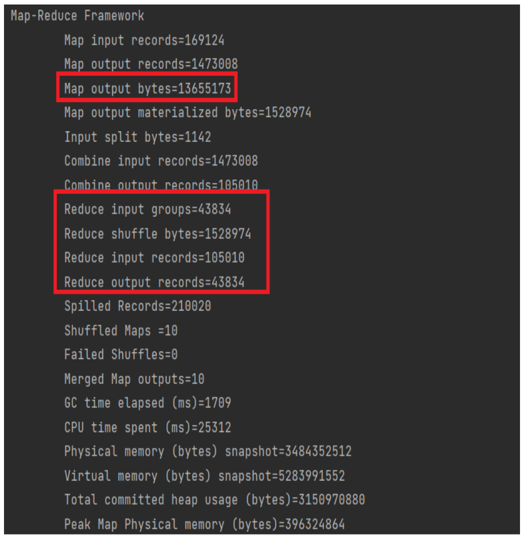

C:\Users\gmani\Desktop\Hadoop outputs\sortedafterstopwordsremoval.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
afterstopwordsremoval
stopwords.txt
part-r-00000 (5)
sortedbeforestopwords.txt
sortedafterstopwordsremoval.txt
Normal text file
length : 546,295 lines : 43,708
Ln : 1 Col : 1 Pos : 1
Windows (CR LF) UTF-8 INS
49°F
Cloudy
Search
4:57 PM
4/1/2023

```
said	7743
one	4998
would	4728
mr	3860
could	3082
like	2848
little	2612
mrs	2546
man	2507
know	2318
well	2309
see	2179
must	2150
much	2143
never	2093
time	2045
go	1990
good	1951
come	1849
say	1835
made	1756
think	1735
old	1714
us	1650
shall	1647
might	1645
dartagnan	1624
may	1602
two	1578
thought	1539
own	1523
dont	1469
upon	1458
first	1437
away	1431
make	1419
way	1391
without	1367
young	1366
great	1337
even	1294
```

Info on how I achieved the result.
In the code I have added all the required patterns or stopwords that the WordCount application needs to remove/ignore. I have added a list of nearly 127 stop words to code. After removing/ignoring the stop words during the mapping phase. I got the output file and I have sorted the output file using command line sort based on values instead of keys.

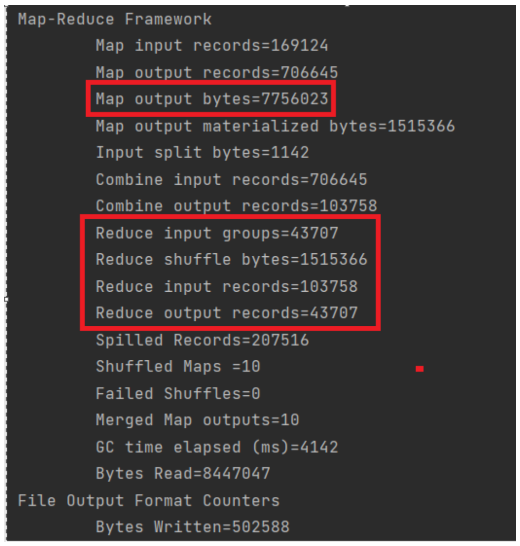
3. Based on the output of your application, how does removing stop words affect the total amount of bytes output by your mappers? Name one concrete way that this would affect the performance of your application.

Below is the mapper output bytes before stop words removal:



```
Map-Reduce Framework
        Map input records=169124
        Map output records=1473008
        Map output bytes=13655173
        Map output materialized bytes=1528974
        Input split bytes=1142
        Combine input records=1473008
        Combine output records=105010
        Reduce input groups=43834
        Reduce shuffle bytes=1528974
        Reduce input records=105010
        Reduce output records=43834
        Spilled Records=210020
        Shuffled Maps =10
        Failed Shuffles=0
        Merged Map outputs=10
        GC time elapsed (ms)=1709
        CPU time spent (ms)=25312
        Physical memory (bytes) snapshot=3484352512
        Virtual memory (bytes) snapshot=5283991552
        Total committed heap usage (bytes)=3150970880
        Peak Map Physical memory (bytes)=396324864
```

Below is the mapper output bytes after ignoring/removing stop words:

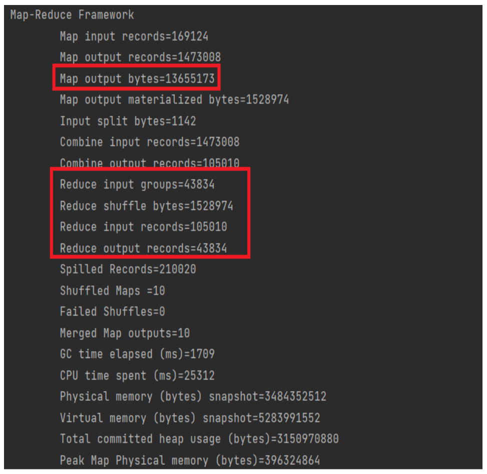


```
Map-Reduce Framework
        Map input records=169124
        Map output records=706645
        Map output bytes=7756023
        Map output materialized bytes=1515366
        Input split bytes=1142
        Combine input records=706645
        Combine output records=103758
        Reduce input groups=43707
        Reduce shuffle bytes=1515366
        Reduce input records=103758
        Reduce output records=43707
        Spilled Records=207516
        Shuffled Maps =10
        Failed Shuffles=0
        Merged Map outputs=10
        GC time elapsed (ms)=4142
        Bytes Read=8447047
File Output Format Counters
        Bytes Written=502588
```

- From the mapper outputs above we can see that map output bytes is significantly reduced after stop words removal from initial 13655173 bytes to 7756023 bytes. This implies that key space is reduced so after stop words removal we need less number of reducers to reduce the key value pairs as the number of Key value pairs generated from mappers are reduced after stop words removal.
- In this, the point to note is as we are removing most command words or high frequency words. There will be significant reduction of key value pairs from mapper output after stop words removal. So the application will do less computations and will run faster.

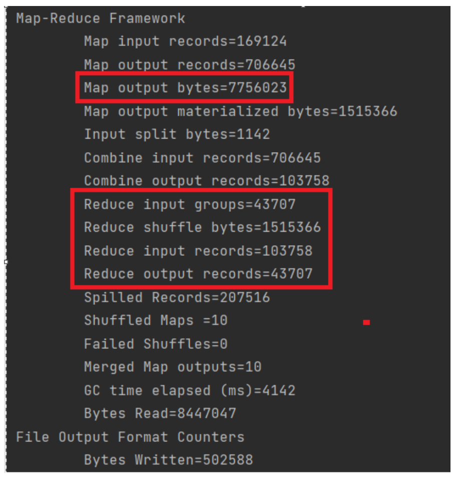
4. Based on the output of your application, what is the size of your keyspace with and without removing stopwords? How does this correspond to the number of stopwords you have chosen to remove?

Below is the mapper output bytes before stop words removal:



```
Map-Reduce Framework
        Map input records=169124
        Map output records=1473008
        Map output bytes=13655173
        Map output materialized bytes=1528974
        Input split bytes=1142
        Combine input records=1473008
        Combine output records=105010
        Reduce input groups=43834
        Reduce shuffle bytes=1528974
        Reduce input records=105010
        Reduce output records=43834
        Spilled Records=210020
        Shuffled Maps =10
        Failed Shuffles=0
        Merged Map outputs=10
        GC time elapsed (ms)=1709
        CPU time spent (ms)=25312
        Physical memory (bytes) snapshot=3484352512
        Virtual memory (bytes) snapshot=5283991552
        Total committed heap usage (bytes)=3150970880
        Peak Map Physical memory (bytes)=396324864
```

Below is the mapper output bytes after ignoring/removing stop words:



```
Map-Reduce Framework
        Map input records=169124
        Map output records=706645
        Map output bytes=7756023
        Map output materialized bytes=1515366
        Input split bytes=1142
        Combine input records=706645
        Combine output records=103758
        Reduce input groups=43707
        Reduce shuffle bytes=1515366
        Reduce input records=103758
        Reduce output records=43707
        Spilled Records=207516
        Shuffled Maps =10
        Failed Shuffles=0
        Merged Map outputs=10
        GC time elapsed (ms)=4142
        Bytes Read=8447047
File Output Format Counters
        Bytes Written=502588
```

- Keyspace generally refers to the set of all possible keys that can be generated from the data processing or set of keys we can get in our entire application. So keyspace is indicated by "**Reduce input/output groups"** value in the application because these are the number of unique keys that are generated after the mapper phase is completed**.**
- From the above outputs we can see that the key space before ignoring stop words is **43834**. Implying 43834 unique keys are generated after the mapper phase. In the similar way key space after ignoring stop words is **43707. So in total the key space is reduced after ignoring stop words.**
- So as we used 127 stop words these 127 stopwordkeys are ignored in the mapper phase so the number of unique keys that will go into the reducer decreased by 127 keys. We confirm the same by subtracting reducer input group values from before and after stopwords removal (43834 - 43707) which is equal to 127.

5. Let's now assume you were going to run your application on the entirety of Project Gutenberg. For this question, assume that there are 100TB of input data, the data is spread over 10 sites, and each site has 20 mappers. Assume you ignore all but the 25 most common words that you listed in question 2. Furthermore, assume that your combiners have been run optimally, so that each combiner will output at most 1 key-value pair per key.

a. How much data will each mapper have to parse?
Data available = 100 TB
Data at each site = 100/10 = 10 TB
No.of mappers in each site = 20 mappers
Data that each mapper has to parse = 10TB/20 mappers = **500 GB.**

b. What is the size of your keyspace?
As we require only 25 most common words we ignore all of the rest of the words. So the no.of unique keys that will be obtained after the mapper phase and before the reducer phase is 25 . So, keyspace is given as **25.**

c. What is the maximum number of key-value pairs that could be communicated during the barrier between mapping and reducing?

- In general, each mapper will have a combiner and the output of each combiner goes to the barrier.

- So each mapper output gives us some number of key value pairs with key space 25 (no.of unique keys)

- As each combiner runs optimally and gives **one key-value pair per key** implying for 25 keys we will get **25 key value pairs for each combiner**.

- So, intotal we have 200 combiners . so the total number of key value pairs from all combiners that go into barrier phase is equal to 25 * 200 = **5000 key value pairs.** (25 * 20 * 10)

d. Assume you are running one reducer per site. On average, how many key-value pairs will each reducer have to handle?

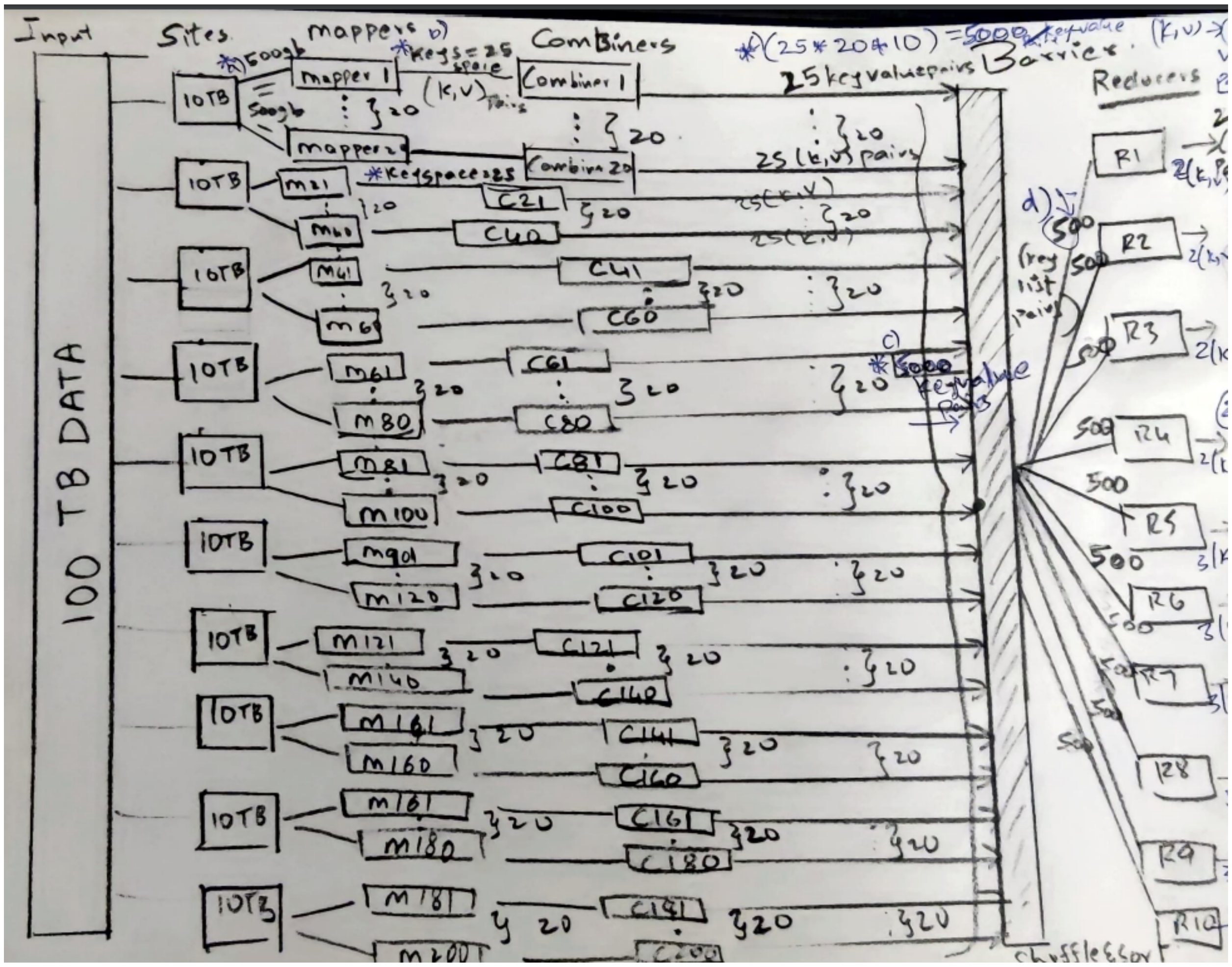
- Now we have 5000 key value pairs as combiner output and these go into the barrier phase where shuffle and sort happens after shuffle and sort we will have 25 keys and their respective lists.

- Considering one reducer per site we have 10 reducers so partitioner (customizable) divides these 25 keys and lists and sends them to reducer.

- So if we consider (key,list) pairs there will be 25 key-list pairs so each reducer gets two or three key-list pairs.

- But in this question as we are considering key-value pairs there will be intotal 5000 key value pairs and they go into 10 reducers implying **each reduce**r gets **500 key value pairs on average. So the answer is 500 key- value pairs.**

6. Draw the data flow diagram for question 5. The diagram should be similar to the diagram shown in the lecture. On your diagram, label the specific quantities you got for 5a,b,c, and d


Input
Sites
mappers b)
Combiners
Barrier
Reducers
100 TB DATA
10TB
500gb
*Keys=25 space
(K,V) Pairs
mapper 1
mapper 20
Combiner 1
Combiner 20
*Keyspace=25
25 keyvaluepairs
25 (k,v) pairs
*(25*20*10) = 5000 keyvalue
c) *5000 keyvalue Pairs
d) 500 (key list pairs)
}20
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
500
Shuffle & Sort