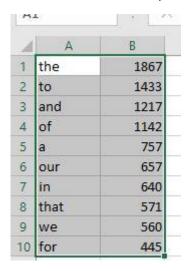
Part 1: Simply running Wordcount1.java

										V			Ŷ.
2017.02.18 17:47:32 UTC	2017.02.18 17:47:38 UTC	2017.02.18 17:48:23 UTC	job_1487438491589_0002	word count	vagrant	default	SUCCEEDED	7	7		1	1	

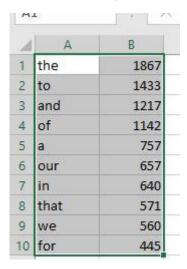
Screenshot of the top 10 words



Part 2: Simply running Wordcount2.java

2017.02.18 18:01:47 UTC	2017.02.18 18:01:55 UTC	2017.02.18 18:02:39 UTC	job_1487440869693_0001	word count	vagrant	default	SUCCEEDED	7	7	1	1
-------------------------------	-------------------------------	-------------------------------	------------------------	---------------	---------	---------	-----------	---	---	---	---

Screenshot of the top 10 words



Part 3: Running Wordcount1.java to produce words which occur more than 4 times

2017.02.18 18:08:00	2017.02.18 18:08:05	2017.02.18	job_1487440869693_0002		vagrant	default	SUCCEEDED	7	7	1	1
UTC	UTC	UTC		count							

Screenshot of the top 10 words

м.	+					
À	А	В				
1	the	1867				
2	to	1433				
3	and	1217				
4	of	1142				
5	а	757				
6	our	657				
7	in	640				
8	that	571				
9	we	560				
10	for	445				

A SMALL ANALYSIS

The interesting thing to note is that the top 10 words of all the 3 outputs IS THE SAME. This is because in step 1 and 2, we are simply running the wordcount.java codes. In the step 3, we skipped all words who occurred less than 4 times. Even in that case, our output list was only trimmed, it did not change.

Part 4: Running Wordcount2.java and skipping prepositions and punctuations

01:54:01	01:54:07	01:54:53	job_1487440869693_0006	word count	vagrant	default	SUCCEEDED	7	7	1	1	
UTC	UTC	UTC										
0047 00 40	0047 00 40	0047 00 40	1-6 4407440000000 000F			4 - 5 15	EAUED	0	0	0	^	

Screenshot of the top 10 words

A:	1	*		×			
A	А		В				
1	the		18	68			
2	and		12	22			
3	а		7	59			
4	our		657				
5	th		5	82			
6	we		5	62			
7	is		3	99			
8	will		3	98			
9	Ì		2	69			
10	this		2	65			

In this case, the output was a little different as compared to the other 3 steps.