CPrE/SE 419: Software Tools for Large-Scale Data Analysis, Spring 2023: Lab 2

GroupMembers:

1. Jyothi Prasanth Durairaj Rajeswari
2. Eranda Sooriyarachchi

Exp 1:

```
cpre419@cpre419-VirtualBox:~/Downloads$ hadoop fs -cat hdfs://localhost:9000/lab2/output/* | head -10
2023-03-01 18:46:43,878 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
!       10526
!'By    1
!'twas  1
!,      1
!As     1
!Ay     1
!Come   1
!Give't 1
!Handkerchief   1
!Hear   1
cat: Unable to write to output stream.
2023-03-01 18:46:44,215 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
cat: Unable to write to output stream.
cpre419@cpre419-VirtualBox:~/Downloads$
```

Exp 2:

For Shakespeare Data

```
cpre419@cpre419-VirtualBox:~/Downloads$ hadoop fs -cat hdfs://localhost:9000/user/cpre419/lab2/output/ex
p2/*
2023-03-01 23:42:11,199 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
to the  1512
it is   1083
i am    1855
i ll    1783
to be   971
in the  1580
my lord 1699
i will  1571
i have  1620
of the  1374
cpre419@cpre419-VirtualBox:~/Downloads$
```

For GutenBurg Data

```
cpre419@cpre419-VirtualBox:~/Downloads$ hadoop fs -cat hdfs://localhost:9000/user/cpre419/lab2/output/ex
p2/*
2023-03-02 00:15:11,778 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
to the  636851
in a    272811
on the  410838
and the 453613
of a    303399
it was  370330
at the  304649
to be   388179
in the  927220
of the  1575063
cpre419@cpre419-VirtualBox:~/Downloads$
```

3) How you might be able to get around the fact that bigrams might span lines of input. Briefly describe how you might deal with that situation? (5 points)

One way we can handle such a situation is to assign multiple lines of texts per node and perform the mapreduce task as in the code above. However that may not be efficient. So a better approach would be to perform a first round of mapreduce and do some further processing in subsequent rounds of mapreduce.

Mapreduce works such that the input file is split and each line of the file is split into a node depending on the size of the distributed system. Then the map function generates key, value pairs. These key value pairs get sorted/shuffled and then the reduce function sums up the values for each key to give the output. In the case of the bigram, we identify the two consecutive words after punctuations are removed. If a single word is more than one line, there would not be any punctuation. After one round of mapreduce, we would have key value pairs. Now we need to run mapreduce on this output of the reduce function of the previous round. This will again form bigrams of those key, value pairs. This should be able to capture bigrams that span multiple lines.