**Implement Word Count using Map Reduce via Python+Hadoop**

**Step 1:-** Create folder in desktop named “**reduce”**

**Step 2:-** Create a text file with the name “***word\_count\_data.txt*** “ and add some data to it. Create in **reduce** folder

**Step3:-** In terminal

cd Desktop/reduce

touch mapper.py

**Step 4:-** Open the reduce folder and open the newly created mapper.py.

Add the following code

#!/usr/bin/env python

# import sys because we need to read and write data to STDIN and STDOUT

import sys

# reading entire line from STDIN (standard input)

for line in sys.stdin:

# to remove leading and trailing whitespace

line = line.strip()

# split the line into words

words = line.split()

# we are looping over the words array and printing the word

# with the count of 1 to the STDOUT

for word in words:

# write the results to STDOUT (standard output);

# what we output here will be the input for the

# Reduce step, i.e. the input for reducer.py

print ('%s\t%s' % (word, 1) )

**Step 5:-** In terminal

cd Desktop/reduce

cat word\_count\_data.txt | python3 mapper.py

**Step 6:-** In terminal

touch reducer.py

**Step 7:-** Open the reduce folder and open the newly created reducer.py.

Add the following code

#!/usr/bin/env python

from operator import itemgetter

import sys

current\_word = None

current\_count = 0

word = None

# read the entire line from STDIN

for line in sys.stdin:

# remove leading and trailing whitespace

line = line.strip()

# splitting the data on the basis of tab we have provided in mapper.py

word, count = line.split('\t', 1)

# convert count (currently a string) to int

try:

count = int(count)

except ValueError:

# count was not a number, so silently

# ignore/discard this line

continue

# this IF-switch only works because Hadoop sorts map output

# by key (here: word) before it is passed to the reducer

if current\_word == word:

current\_count += count

else:

if current\_word:

# write result to STDOUT

print ('%s\t%s' % (current\_word, current\_count))

current\_count = count

current\_word = word

# do not forget to output the last word if needed!

if current\_word == word:

print ('%s\t%s' % (current\_word, current\_count) )

**Step 8:-** In Terminal

cd Desktop/reduce

cat word\_count\_data.txt | python3 mapper.py | sort -k1,1 | python3 reducer.py

Step 9: Now let’s start all our Hadoop servers with the below command.   
In Terminal

cd  
start-dfs.sh

start-yarn.sh

**Step 10:-** In terminal

hdfs dfs -mkdir /word\_count\_in\_python

**Step 11:-** In Terminal

hdfs dfs -copyFromLocal /home/aiml/Desktop/reduce/word\_count\_data.txt /word\_count\_in\_python

**Step 12:-** Now our data file has been sent to HDFS successfully. we can check whether it sends or not by using the below command or by manually visiting our HDFS. In Terminal

hdfs dfs -ls /

hdfs dfs -ls /word\_count\_in\_python

**Step 13:-** Let’s give executable permission to our mapper.py and reducer.py with the help of below command. In Terminal

cd Desktop/reduce

chmod 777 mapper.py reducer.py

**Step 14:-** Download Jar File**, Extract the jar and store it in reduce folder**

<https://jar-download.com/artifacts/org.apache.hadoop/hadoop-streaming>

choose hadoop-streaming from group org.apache.hadoop (version 3.3.6)

**Step 15:-** In Terminal (*don’t copy entire code, go line by line*)

hadoop jar /home/aiml/Desktop/reduce/hadoop-streaming-3.3.6.jar \

-input /word\_count\_in\_python/word\_count\_data.txt \

-output /word\_count\_in\_python/output \

-mapper "python3 /home/aiml/Desktop/reduce/mapper.py" \

-reducer "python3 /home/aiml/Desktop/reduce/reducer.py"

**Step 16:-** In Terminal

hdfs dfs -cat /word\_count\_in\_python/output/part-00000