**Task 3:**

Map-Reduce Phases in Relative frequencies in word-pairs problem

Pseudo Code in terms of Map Reduce Phases

1. Input of Input File
2. Mapper1 :- map (word1,1)
3. Reducer1:-reducebykey(word1,count1)
4. Output1
5. Input of Input File
6. Mapper2:- map(word1,word2,1) similarly map(word2,word3,1) **//called bigrams**
7. Reducer2:-reducebykey(word1, word2,count2) where key:- (word1,word2)
8. Output2
9. Input of Output1 , Output2 with key as word1,word2
10. Reducer3 where based on key count2/count1 with **Output:(Word1 , Word1,Word2, count2/count1)**

Now , these phases are implemented in PySpark

import pyspark

from pyspark import SparkContext

from operator import add

sc = SparkContext('local[\*]')

f=sc.textFile('sample\_12.txt')

**Steps 2, 3 of Mapper1 and Reducer1 are performed here**

data = f.flatMap(lambda line: line.split ()).map(lambda x : (x,1)).reduceByKey(add)

**Steps 6, 7 of Mapper2 and Reducer3 are performed here**

data2 = f.flatMap(lambda line: line.split(".")).map(lambda line: line.strip().split(" ")).flatMap(lambda xs: (tuple(x) for x in zip(xs, xs[1:]))).map(lambda x : (x,1)).reduceByKey(add)

**Step 10 of reducing based on the key is performed here by joining by key and then calculating relative frequency**

yahoo=data.cartesian(data2).filter(lambda x : x[0][0]==x[1][0][0]).map(lambda x : (x[0][0],x[1][0],x[1][1]/x[0][1])).collect()

**Output**

print(yahoo)