Q1. Write an in-mapper combiner algorithm modifying Co-occurrence Matrix (pairs approach) algorithm.

class Mapper

Method initialize()

H = new AssociativeArray()

Method Map (docid a, doc d)

For all term u in record r do

For all term v in Window(u) do

H{(u,v)} = H{(u,v)} + 1

Method close()

Emit((u,v), H)

Q2. Write an in-mapper combiner **algorithm** modifying Co-occurrence Matrix (stripe approach) algorithm.

Class Mapper

Method initialize()

A = new AssociativeArray()

Method Map (docid a, doc d)

For all term u in record r do

H = new AssociativeArray()

For all term v in Window(u) do

H{v} = H{v} + 1

A{u} = A{u} + H

Method close()

For all term u in A do

Emit(u, A)

Q3. Assume that there are two input spits and two reducers. Note that Mapper 1 and Reducer 1 run on the same machine. Mapper 2 and Reducer 2 run on the same machine.

Further, let the partitioner assign all words less than letter ‘k’ to Reducer 1 and everything else to Reducer 2.

Input Split 1 : [ {cat mat rat, cat}, {cat bat cat pat},{cat bat rat bat}] (Note : 3 records)

Input Split 2 : [{cat rat bat rat}, {bat mat pat bat}, {pat cat bat mat}] (Note: 3 records)

**Let the window of X, W(X) be set of all term after X and before the next X.**

**Pair approach:**

Mapper 1 output :

W(cat) = {mat, rat} => ((cat, mat), 1) ((cat, rat), 1)

W(mat) = {rat, cat} => ((mat, rat), 1) ((mat, cat), 1)

W(rat) = {cat} => ((rat, cat), 1)

W(cat) = {bat} => ((cat, bat), 1)

W(bat) = {cat, pat} => ((bat, cat), 1) ((bat, pat), 1)

W(cat) = {pat} => ((cat, pat), 1)

W(cat) = {bat, rat, bat} => ((cat, bat), 1) ((cat, rat), 1) ((cat, bat), 1)

W(bat) = {rat} => ((bat, rat), 1)

W(rat) = {bat} => ((rat, bat), 1)

Mapper 2 output :

W(cat) = {rat, bat, rat} => ((cat, rat), 1) ((cat, bat), 1) ((cat, rat), 1)

W(rat) = {bat} => ((rat, bat), 1)

W(bat) = {rat} => ((bat, rat), 1)

W(bat) = {mat, pat} => ((bat, mat), 1) ((bat, pat), 1)

W(mat) = {pat, bat} => ((mat, pat), 1) ((mat, bat), 1)

W(pat) = {bat} => ((pat, bat), 1)

W(pat) = {cat, bat, mat} => ((pat, cat), 1) ((pat, bat), 1) ((pat, mat), 1)

W(cat) = {bat, mat} => ((cat, bat), 1) ((cat, mat), 1)

W(bat) = {mat} => ((bat, mat), 1)

Reducer 1 input :

Bat

((bat, cat), [1])

((bat, pat), [1, 1])

((bat, rat), [1, 1])

((bat, mat), [1, 1])

Cat

((cat, mat), [1, 1])

((cat, rat), [1, 1, 1, 1])

((cat, bat), [1, 1, 1, 1, 1])

Reducer 2 input :

Mat

((mat, rat), [1])

((mat, cat), [1])

((mat, pat), [1])

((mat, bat), [1])

Pat

((pat, bat), [1, 1])

((pat, cat), [1])

((pat, mat), [1])

Rat

((rat, bat), [1, 1])

Reducer 1 output :

Bat

((bat, cat), 1)

((bat, pat), 2)

((bat, rat), 2)

((bat, mat), 2)

Cat

((cat, mat), 2)

((cat, rat), 4)

((cat, bat), 5)

Reducer 2 output :

Mat

((mat, rat), 1)

((mat, cat), 1

((mat, pat), 1)

((mat, bat), 1)

Pat

((pat, bat), 2

((pat, cat), 1)

((pat, mat), 1)

Rat

((rat, bat), 2)

**In-Mapper Combining Version of the Pair approach.**

Mapper 1 output :

((bat, cat), 2)

((bat, pat), 1)

((cat, mat), 1)

((cat, rat), 2)

((cat, bat), 3)

((cat, pat), 1)

((mat, rat), 1)

((mat, cat), 1)

((rat, cat), 1)

((rat, bat), 1)

Mapper 2 output :

((cat, rat), 2)

((cat, bat), 2)

((cat, mat), 1)

((bat, rat), 1)

((bat, mat), 2)

((bat, pat), 1)

((mat, pat), 1)

((mat, bat), 1)

((pat, bat), 2)

((pat, cat), 1)

((pat, mat), 1)

((rat, bat), 1)

Reducer 1 input :

Bat

((bat, cat), [2])

((bat, pat), [1, 1])

((bat, rat), [1])

((bat, mat), [2])

Cat

((cat, mat), [1, 1])

((cat, rat), [2, 2])

((cat, bat), [3, 2])

((cat, pat), [1])

Reducer 2 input :

Mat

((mat, rat), [1])

((mat, cat), [1])

((mat, pat), [1])

((mat, bat), [1])

Pat

((pat, bat), [2])

((pat, cat), [1])

((pat, mat), [1])

Rat

((rat, cat), [1])

((rat, bat), [1,1])

Reducer 1 output :

Bat

((bat, cat), 2)

((bat, pat), 2)

((bat, rat), 1)

((bat, mat), 2)

Cat

((cat, mat), 2)

((cat, rat), 4)

((cat, bat), 5)

((cat, pat), 1)

Reducer 2 output :

Mat

((mat, rat), 1)

((mat, cat), 1)

((mat, pat), 1)

((mat, bat), 1)

Pat

((pat, bat), 2)

((pat, cat), 1)

((pat, mat), 1)

Rat

((rat, cat), 1)

((rat, bat), 2)

**Illustrate Stripe approach**

Input Split 1 : [ {cat mat rat, cat}, {cat bat cat pat},{cat bat rat bat}] (Note : 3 records)

Input Split 2 : [{cat rat bat rat}, {bat mat pat bat}, {pat cat bat mat}] (Note: 3 records)

Mapper 1 output :

(cat, [(mat, 1), (rat, 1)])

(mat, [(rat, 1), (cat, 1)])

(rat, [(cat, 1)])

(cat, [(bat, 1)])

(bat, [(cat, 1), (pat, 1)])

(cat, [(pat, 1)])

(cat, [(bat, 2), (rat, 1)])

(bat, [(rat, 1)])

(rat, [(bat, 1)])

Mapper 2 output :

(cat, [(rat, 2), (bat, 1)])

(rat, [(bat, 1)])

(bat, [(rat, 1)])

(bat, [(mat, 1), (pat, 1)])

(mat, [(pat, 1), (bat, 1)])

(pat, [(bat, 1)])

(pat, [(cat, 1), (bat, 1), (mat, 1)])

(cat, [(bat, 1), (mat, 1)])

(bat, [(mat, 1)])

Reducer 1 input:

Bat

(bat, [(cat, 1), (pat, 1), (rat, 1), (rat, 1), (mat, 1), (pat, 1), (mat, 1)])

Cat

(cat, [(mat, 1), (rat, 1), (bat, 1), (pat, 1), (bat, 2), (rat, 1), (rat, 2), (bat, 1), (bat, 1), (mat, 1)])

Reducer 2 input

Mat

(mat, [(rat, 1), (cat, 1), (pat, 1), (bat, 1)])

Pat

(pat, [(bat, 1), (cat, 1), (bat, 1), (mat, 1)])

Rat

(rat, [(cat, 1), (bat, 1), (bat, 1)])

Reducer 1 output :

Bat

(bat, [(cat, 1), (pat, 2), (rat, 2), (mat, 2)])

Cat

(cat, [(mat, 2), (rat, 4), (bat, 5), (pat, 1)])

Reducer 2 output :

Mat

(mat, [(rat, 1), (cat, 1), (pat, 1), (bat, 1)])

Pat

(pat, [(cat, 1), (bat, 2), (mat, 1)])

Rat

(rat, [(cat, 1), (bat, 2)])

**Illustrate In-Mapper Combining Version of the Stripe approach**

Input Split 1 : [ {cat mat rat, cat}, {cat bat cat pat},{cat bat rat bat}] (Note : 3 records)

Input Split 2 : [{cat rat bat rat}, {bat mat pat bat}, {pat cat bat mat}] (Note: 3 records)

Mapper 1 output :

(bat, [(cat, 1), (pat, 1), (rat, 1)])

(cat, [(mat, 1), (rat, 2), (bat, 3), (pat, 1)])

(mat, [(rat, 1), (cat, 1)])

(rat, [(cat, 1), (bat, 1)])

Mapper 2 output :

(bat, [(rat, 1), (mat, 2), (pat, 1)])

(cat, [(rat, 2), (bat, 2), (mat, 1)])

(mat, [(pat, 1), (bat, 1)])

(pat, [(cat, 1), (bat, 2), (mat, 1)])

(rat, [(bat, 1)])

Reducer 1 input :

Bat

(bat, [(cat, 1), (pat, 1), (rat, 1), (rat, 1), (mat, 2), (pat, 1)])

Cat

(cat, [(mat, 1), (rat, 2), (bat, 3), (pat, 1), (rat, 2), (bat, 2), (mat, 1)])

Reducer 2 input :

Mat

(mat, [(rat, 1), (cat, 1), (pat, 1), (bat, 1)])

Pat

(pat, [(cat, 1), (bat, 2), (mat, 1)])

Rat

(rat, [(cat, 1), (bat, 1), (bat, 1)])

Reducer 1 output :

Bat

(bat, [(cat, 1), (pat, 2), (rat, 2), (mat, 2)])

Cat

(cat, [(mat, 2), (rat, 4), (bat, 5), (pat, 1)])

Reducer 2 output :

Mat

(mat, [(rat, 1), (cat, 1), (pat, 1), (bat, 1)])

Pat

(pat, [(cat, 1), (bat, 2), (mat, 1)])

Rat

(rat, [(cat, 1), (bat, 2)])