ConfigObject bMatrix = new ConfigObject() //multidimensional map

ConfigObject termCMatrix = new ConfigObject() //multidimensional map

ConfigObject tf = new ConfigObject()

ConfigObject w = new ConfigObject()

def df = [:]

def idf = [:]

//assert co.one.two.equals('something')

import java.lang.\*;

def doc1 = ['this','is','not','document','one','one']

def doc2 = ['that','is','a','document','doc3']

def query = ['this', 'is', 'a', 'test']

def j = i = 0

def union = doc1.plus(doc2).plus(query).unique { a, b -> a <=> b }

def N = 3 //no of document

union.each{

if(doc1.contains(it))

bMatrix['doc1'][it] = 1

else

bMatrix['doc1'][it] = 0

if(doc2.contains(it))

bMatrix['doc2'][it] = 1

else

bMatrix['doc2'][it] = 0

if(query.contains(it))

bMatrix['query'][it] = 1

else

bMatrix['query'][it] = 0

}

union.each{

termCMatrix['doc1'][it] = doc1.count(it)

termCMatrix['doc2'][it] = doc2.count(it)

termCMatrix['query'][it] = query.count(it)

}

maxDoc1 = termCMatrix['doc1'].max{it.value}.value

maxDoc2 = termCMatrix['doc2'].max{it.value}.value

maxQuery = termCMatrix['query'].max{it.value}.value

union.each{

tf['doc1'][it] = termCMatrix['doc1'][it]/maxDoc1

tf['doc2'][it] = termCMatrix['doc2'][it]/maxDoc2

tf['query'][it] = termCMatrix['query'][it]/maxQuery

}

union.each{

df[it] = bMatrix['doc1'][it] + bMatrix['doc2'][it] + bMatrix['query'][it]

}

union.each{

idf[it] = Math.log(N/df[it])

}

union.each{

w['doc1'][it] = tf['doc1'][it] \* idf[it]

w['doc2'][it] = tf['doc2'][it] \* idf[it]

w['query'][it] = tf['query'][it] \* idf[it]

}

def cosine(w1,w2,union){

def temp = 0.0

union.each{

temp = w1[it]\*w2[it]+temp

}

return temp

}

def cosine1 = cosine(w['doc1'],w['query'],union)

def cosine2 = cosine(w['doc2'],w['query'],union)

println cosine1

println cosine2