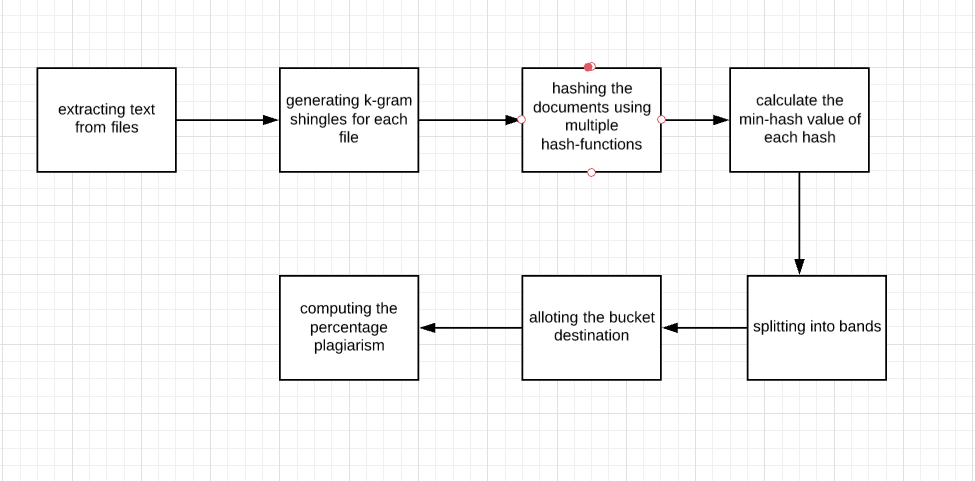
ARCHITECTURE:



DATA STRUCTURES USED:

The major data structures used include :

1. Set and Lists for maintaining shingles and hashed values;
2. hash map to maintain the buckets after hashing

RUNNING TIME:

The running time for LSH is:

O(LDK)+O(LDN/2^K)

where L: no.of hash functions

D:no.of dimensions

K:no.of hyperplanes while hashing

N:no.of points in space or documents

CODE:

#=======================================================================# PLAGIARISM CHECKING TOOL USING LSH TO REDUCE DIMENSIONALITY OF DATA

#=======================================================================

from \_\_future\_\_ import division

import os

import re

import random

import time

import binascii

from bisect import bisect\_right

from heapq import heappop, heappush

a=os.listdir("C:\\Users\\karth\\Desktop\\ir2") #path to the document corpus is given here

k\_gram=3 #shingle length can be adjusted here

#======================================================================= SHINGLING

#=======================================================================

def shingles(string):

return set(string[head:head+k\_gram] for head in range(0,len(string)-k\_gram))

shingles\_list=list()

all\_shingles\_set=set();

for file in a:

f=open(os.path.join("C:\\Users\\karth\\Desktop\\ir2",file),'r')

content=f.read()

shingles\_list.append(shingles(content))

# print(shingles\_list)

for i in range(len(shingles\_list)):

for j in shingles\_list[i]:

all\_shingles\_set.add(j)

# print(all\_shingles\_set)

all\_shingles\_list=list(all\_shingles\_set)

#=======================================================================

DOCUMENT MATRIX

#=======================================================================

matrix=[[0 for i in range(len(a))]for j in range(len(all\_shingles\_list))]

for i in range(len(all\_shingles\_list)):

for j in range(len(a)):

if all\_shingles\_list[i] in shingles\_list[j]:

matrix[i][j]=1

#=======================================================================

MIN-HASHING

#=======================================================================hashes\_count=20

t0=time.time()

maxShingleID=2\*\*32-1

nextprime= 4294967311

def pickRandomCoeffs(k): #function to create 'k' random numbers,thereby creating random hash-functions

# Create a list of 'k' random values.

randList = []

while k > 0:

# Get a random shingle ID.

randIndex = random.randint(0, maxShingleID)

# Ensure that each random number is unique.

while randIndex in randList:

randIndex = random.randint(0, maxShingleID)

# Add the random number to the list.

randList.append(randIndex)

k = k - 1

return randList

coeffA = pickRandomCoeffs(hashes\_count)

coeffB = pickRandomCoeffs(hashes\_count)

k\_index=pickRandomCoeffs(len(all\_shingles\_list))

signature=[[0 for i in range(len(a))]for j in range(hashes\_count)]

for i in range(hashes\_count): #computing minimum value for each hash function

for j in range(len(a)):

minval=2\*\*32-1

for k in range(len(all\_shingles\_list)):

if matrix[k][j]==1:

if minval>(((coeffA[i])\*k+(coeffB[i]))%nextprime):

minval=(((coeffA[i])\*k+(coeffB[i]))%nextprime)

signature[i][j]=minval

print('\n','signature matrix: ','\n')

for i in range(len(signature)):

print(signature[i])

bands=10

rows=2

#===================================================================== LOCALITY SENSITIVE HASHING

#=======================================================================hash\_table\_list=list()

for i in range(bands): #hashing documents into respective buckets

hash\_table=dict()

for j in range(len(a)):

tup=list()

for k in range(rows):

tup.append(signature[i\*rows+k][j])

if hash(tuple(tup)) in hash\_table:

hash\_table[hash(tuple(tup))].append(j)

else:

hash\_table[hash(tuple(tup))]=[j]

hash\_table\_list.append(hash\_table)

print('\n','buckets are: ','\n')

for i in range(len(hash\_table\_list)):

print(hash\_table\_list[i])

doc\_id=0 #you can also take input for the document you wish to compute the candidate pairs for

similar\_docs=set()

for i in range(len(hash\_table\_list)):

for list in hash\_table\_list[i].values():

if doc\_id in list:

for j in list :

if j!=doc\_id:

similar\_docs.add(j)

print('\n','similar documents are: ',similar\_docs)