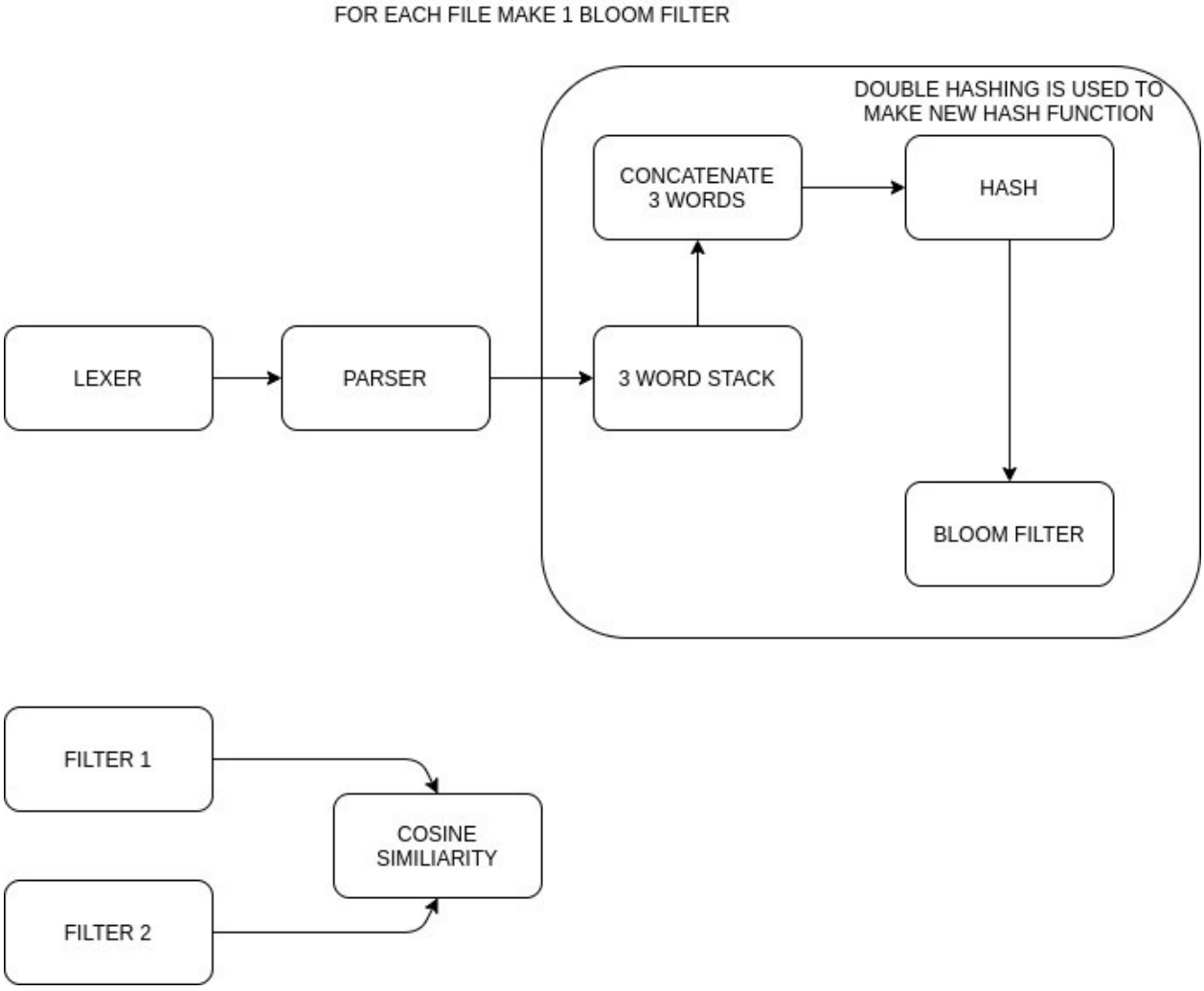
PLAGIARISM CHECKER

COP290
ASSIGNMENT 8 BY SAGAR SHARMA 2018CS10378

Directory Structure:
|-------- src
| |----------plague.l
| |----------plague.y
| |----------bloomFilter.c/bloomFilter.h
| |----------bloomFilter.h
|
|---------exe
| |----------plague
|
|----------corpus files



GENERAL ALGORITHM

1. For each file, form a bloom filter. Do this inserting every consecutive 3 words concatenated into the filter.
2. Compare the two filter by taking cosine similarity between the two arrays of the two bloom filter.

Example:
./exe/plague ./corpus_files/catchmeifyoucan.txt ./corpus_files
hal10.txt 28.206024
bef1121.txt 5.110178
edo14.txt 3.465087
sra119.txt 2.033421
ckh80.txt 3.163811
bgt221.txt 2.665508
abf0704.txt 3.871665
sra31.txt 4.563520
hte42.txt 3.082405
erk185.txt 2.444318
edo26.txt 2.642005
sra42.txt 5.872892
esv254.txt 7.806428
bwa248.txt 3.655930
abf70402.txt 2.203731
ecu201.txt 28.326792
ehc229.txt 4.478337
edo20.txt 6.560177
sra126.txt 6.571851
tyc12.txt 48.492893
jrf1109.txt 3.919395
sra107.txt 5.245930
catchmeifyoucan.txt 99.896797
prz100.txt 6.984127
bmu5.txt 5.446723

DETAILS:

TO INSERT 3 CONSECUTIVE WORDS :

1. Recieve words from parser, push them into stack, if stack is full(size for my implementation is 3). then free stack[0] and push new word.
2. After pushing new word concatenate stack[0] + " " stack[1] + " " + stack[2].
3. Insert the new string to bloom filter.

IN BLOOM FILTER:

1. I have provided false positive probability to be 0.3 (Tested with other probabilities no significant change coming in similarities). Expected size of file to be 10000 words. Just to safekeep size limit. These two parameter are enough to calculate size and no of hash functions (hash_count) needed.
2. A string when insert into bloom filter, is hashed hash_count number of times with different seed using double hash scheme (djb2 and sdbm hash functions)

3. The indices recieved are marked as 1.

SIMILARITY:
As the two bloom filter contain arrays containing 1 and 0, I decided to consider cosine similarity as my similarity measure.
I take scalar product of two vector and divide by product of magnitude of two vectors.

Let v1 = a1,a2,a3,a4.....
Let v2 = b1,b2,b3.b4.....

$$s = \frac{(a1*b1 + a2*b2 + a3*b3 \ldots\ldots\ldots\ldots\ldots)}{sqrt(a1^2 + a2^2 ..)*sqrt(b1^2 + b2^2 ...)}$$

TIME and SPACE COMPLEXITY ANALYSIS

Let max file size be n words. Where I expect n<10000.
Let len be the maximum length of 3 consecutive words. Where I expect len<40 letters

Complexity of sdbm hash = O(len)
Complexity of djb2 hash = O(len)

If expected no of elements n and desired false positive probability 0<P<1 is given then size of bloom filter is calculated as

$m = -\frac{n \ln P}{(ln2)^2}$ floor is used here
optimum no of hash functions (k) is given by

$k = \frac{m}{n} ln2$ floor is used here

Space complexity of bloom filter is O(m) = O(n * ln(p))

and hash_count is O(m/n) = O(|lnP|)

time complexity of insert becomes O(k * (O(djb2) + O(sdbm))) = O(|lnP| * len)

time complexity of parsing a document is O(n*len)

time complexity of parsing and making filter= n* (O(len) + O(insert)) = O(n*len + n* |lnP| *len)

time complexity of finding similarity = O(m)

SPACE COMPLEXITY = no of corpus files * O(n* |lnP|)
we eventually free the strings stored, only 3 strings are stored at time in memory

TIME COMPLEXITY = no of corpus files * O(n*len + n*len*|lnP| + m)