Natalie Valett, nvalett
CSE13S
Prof. Darrell Long
May 25, 2021

Asgn7 Design: Huffman Coding

The purpose of this assignment is to create a “firewall” which parses any text given and censors the text based on the given words to filter or replace. By adding all badspeak and oldspeak words to a Bloom Filter, we’re able to efficiently query whether the word being screened may be a bad word, or is definitely not a bad word. If the bloom query comes back potentially positive, our filter will then query the hash table, where the badspeak words are definitely stored in linked lists, which can be indexed by their corresponding hash values. If the word being screened is in the hash table and has no corresponding newspeak translation, it is badspeak. This means the user has committed thoughtcrime and should be reprimanded for using badspeak, so we append the badspeak word to an array to return to them. If the bad word does have a newspeak translation, the message output should instruct them on the propper newspeak etiquette. Enter both the oldspeak bad word and translation into arrays to return.

Args:
-h: prints out the program usage. Refer to the reference program in the resources repository for what to print.
-t: size. specifies that the hash table will have size entries (the default will be 10000).
-f: size. specifies that the Bloom filter will have size entries (the default will be 220).
-m: will enable the move-to-front rule. By default, the move-to-front rule isdisabled.
-s: will enable the printing of statistics to stdout. The statistics to calculate are the total number of seeks, the average seek length, the hash table load, and the Bloom filter load.

Node
Each node contains the oldspeak bad word and the corresponding newspeak translation, if it exists. They also contain a next and prev pointer to nodes that precede and succeed them in the linked list they’re nested within. They are used in the linked lists to represent each bad word with the hash value at that index of the hash table,

Linked List
A linked list is a linked data structure which is composed of nodes who indicate which node comes before and after itself. Each linked list has a length, the number of nodes, a head and tail sentinel node, and boolean flag mtf, indicating whether the move to front option was enabled by the user.

Hash Table

A hash table is an array of linked lists. Each bad word will be hashed and stored at the index of the hash table, adding that node to the linked list at that index of the hash table. Each time a word passes the bloom filter check, the hash table will be queried by hashing the word and walking through the linked list at that index, checking each node to see if the badspeak is equal to the word being searched.
If mtf is enabled, each time the node is searched it gets moved to the front of the linked list.

Bloom Filter
A bloom filter acts as a low-cost, low-accuracy first check of whether or not the word in question is possibly a bad word. The BF itself is simply a large bitvector which stores either a 1 or 0 in all its indices. Upon processing the newspeak and oldspeak at the start of the program, each oldspeak word will be hashed three times, and for each hash result, the bit at each of those indices will be set. Therefore when scanning each word later to filter the data, if any of the word's hash values correspond to a BF index that equals 0, we can be certain that the word is not a bad word, and not have to put it through the hash table check.

main function
Psuedocode:
```
bf_create()
ht_create()
// populate ht and bf
For line in badspeak.txt:
    bf_insert(bf, badspeak)
    ht_insert(ht, badspeak, NULL)
For old, new in newspeak.txt:
    bf_insert(bf, old)
    ht_insert(ht, old, new)
// read from infile using given parsing code
While ((word = nextword(infile, &re)) != NULL):
    If (bf_probe(bf, tolower(word))):
        // ht check
        Node *n = ht_lookup(ht, word)
        If n != NULL:
            If n->newspeak == NULL:
                badspeak.append(n->badspeak)
                Thoughcrime = true
            Else:
                oldspeak.append(n->badspeak)
                newspeak.append(n->newspeak)
                Rightspeak = 1
    If (thoughcrime && rightspeak):
        print(message)
        For word in badspeak:
            print(word)
        For i in range(oldspeak):
            print(oldspeak[i] + “ -> “ + newspeak[i]
    Else If (thoughcrime):
        print(message)
        For word in badspeak:
            print(word)
    Else If (rightspeak):
        print(message)
        For i in range(oldspeak):
            print(oldspeak[i] + “ -> “ + newspeak[i]
```

Regular Expressions: (regex)

- pattern matching

+ : one or more   . : matches any character

* : zero or more

? : zero or one (optional)

| : or

(...) : precedence

[chars] : match chars

{#} : # of times str is matched

Ex: a+ : matches strings w/ one or more a's

a* : matches strings w/ zero or more a's

"a" matches w/ a+

" " not matched by a+

(0|1)+ : matches strings w/ @ least 1 "0" or "1"

"0110" matched by (0|1)+

[a-z] : matches any symbol btw a-z inclusive

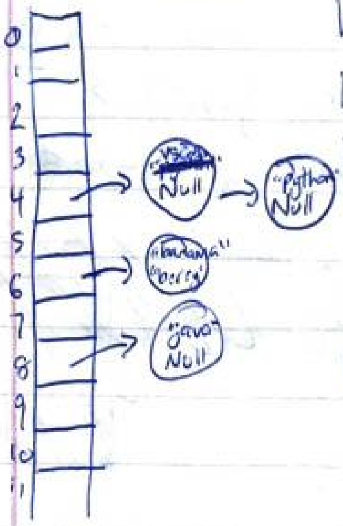
[a-z]+ : matches strings w/ 1 or more chars a-z (inclu

"([a-z]+)-([a-z]+)" : matches hyphenated words

"([a-z]+)'([a-z]+)" : matches contractions (w/ apostrophe) i.e. (don'

"([a-z]+)-([a-z0-9]+)" : matches hyphenated words w/ #s @ end

Hash Table



• Null if no newspeak translation exists

hash(ht->salt, "java") -> 8

hash(ht->salt, "python") -> 4 } hash collission

hash(ht->salt, "vscode") -> 4

• insert el @ front of linked list @ that index

hash(ht->salt, "bananna") -> 6

Ex: is "computer" a bad word?

① Bloom check:

```
    hash(pri, "computer") -> 2
    hash(sec, "comp...") -> 4
    hash(ter, "comp...") -> 9
if all 3 bits == 1:
    computer's probably been added // prob a bad word
```

② Check hash table

```
    hash(ht->salt, "computer") -> 4
    • Go to index 4 of Hash table &
      walk the linked list
    for each node:
        if node->oldspeak == 'computer':
            bad word
        else:
            not bad word // bloom had false pos
```

- If bloom filter returns potentially positive, Check hash table to confirm
- If word found in hash:

```
    If word has newspeak translation
        oldspeak
    else:
        badspeak
```

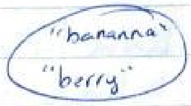
Phase 1)

```
    Add words from badspeak.txt
        to BF & hashtable
    Add words from newspeak.txt to
        BF & HT
```

Phase 2) (firewall)

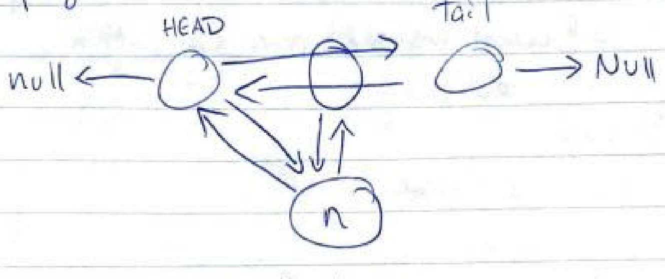
```
    read in user input
        for each word, query bloom filter
            if in bloom:
                check HT
            else:
                pass // not bad word
```

Linked Lists:



• insert in head

Empty



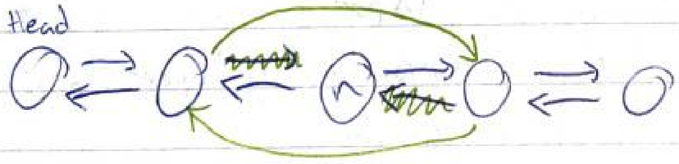
n->next = head->next

n->prev = head

head->next = n

n->next->prev = n

• Move to front enabled should move node to front of linked list whenever queried in HT



1) Bridge over n (disconnect)

n->prev->next = n->next

n->next->prev = n->prev

n->next = head->next

n->prev = head

head->next = n

n->next->prev = n

result: