Assignment 6 Final Design

James Contini

2022 November 20

General information

Citations

None.

Description

Program takes a stdin file, a list of badspeak words, a text file of oldspeak and newspeak pairs. The badspeak and oldspeak should be added to the bloom filter. Oldspeak and newspeak should be added to the hash table.

From stdin read a stream of words to test. For each word check if its in the bloom filter. If there is a negative do nothing. If *bf_probe*() determines it is in the bloom filter, search the hash table for the word. If the word doesn't have a newspeak translation, then the citizen is guilty of *thoughtcrime*. Insert this word in the list of evidence of badspeak words for this citizen. If the word turns out to just be oldspeak not bad speak add it to a list of evidence on evidence for being sent to a joy camp for rightspeak. If the hash list does not the old/badspeak then the bloom filter issues a false positive, do nothing. We can use any type of data strcture to hold this evidence.

Depending on these combinations, only badspeak, only oldspeak, a mix, they get different messages.

1 Bit Vector

The bitvector module I implemented works very simply. A bit vector is comprised of a pointer to blocks of uint64t type. To set a bit in the vector *or* we find the block by floor dividing the bit number by sixty four. Then to find which bit inside we take the remainder of the bit number modulo sixty four. Then we create a mask of a one at that number and zeros everywhere else and then *or* that mask with the original bit vector. To delete the mask becomes ones and a zero and the operator *and* and to inspect we *or* the vector with a of zeroes except for the set bit. If the result of that operation is equivalent to the original mask we can conclude that there was already a one there.

2 Bloom Filter

The bloom filter uses the bitvector to store its data. To insert a long string of words split by newlines is inputted. For every word denoted by a newline it is hashed 5 times with different salts and the hash result is the bit the program sets in the bitvector. To probe the vector we hash a string and see if it has at least one bit not set, if a bit is not set then it is define not in the hash table. But if all bits are set, there is a good chance it is.

3 Node/Hash table

A node is comprised of a pointer to the node ahead of it and the node behind it, a pointer to a char* of oldspeak, and a pointer to some chars of newspeak. A node is a sentinel node if the oldspeak and newspeak point to NULL. The hash table with be made of an array of linked lists. Hash an oldspeak value to find out which bucket/linked list its in and then iterate over the linked list to see if the oldspeak value is in there. To insert a node into the hash table hash the oldspeak and and then ll_insert into the appropriate bucket. Statistics are kept by iterating the corresponding field of the object to track how many times something occurred.

4 Linked list

A doubley linked list has twice the overhead as a singularly linked list. In my implementation of a linked list there is a previous field and a forward field holding the address of the node ahead or behind the current node. To lookup a value, start at the head and traverse the next node values until the tail is reached. If an oldspeak field matches the desired oldspeak, then the oldspeak is in there. To insert a node put it at the head of the list and move the previous and next pointers so that they point to the inserted node.

5 Parser

Parser is a struct containing a buffer and a pointer to a file as well as a line offset. To find the "next word" in the file, the function will iterate over the characters in the buffer until it finds an alphanumeric or ' or - and from there it will copy that character into another buffer and any more acceptable characters afterward in sequence until a bad character, null termination, a space, a newline, or a the end of the buffer. If it is a bad character or a space, the parser field "line_offset gets changed to match where the end of the word is. If the word ends because it is a newline or a null terminator, the buffer is exhausted and needs to be replaced. If fgets doesn't return NULL then there are more words in the file to be moved into the buffer, if the buffer is NULL then that was the last word in the file and the function returns false, otherwise, it returns true.

6 Banhammer

First, my program uses geptopt to parse options from the command line and a uint8_t type integer to function as a set to keep track of my command options. If the unsigned integer is greater than 2 there was a bad option and returns one as well printing help to standard error. If the number is greater than one that means the help options were inputted so zero is returned and the help message is printed to standard error. If the number just equals one then statistics are printed to standard output.

After all of these checks have been made, I initialize two Parser objects, one for badspeak and one for the oldspeak dictionary. Then using the next word function, the program inserts all old and badspeak from both parser instances into the hash table and the bloom filter. Then the newspeak is inputted into the hash table with the oldspeak. Then a third parser instance is made to read from standard input. For each acceptable word in standard input, the bloom filter is probed, if it returns true then it is searched for in the hashtable. If the search returns a NULL value, the bloom filter was incorrect if there is a hit then its checked for a newspeak pair, if there is no newspeak it's badspeak. Depending on if it's oldspeak or badspeak the word gets added to a corresponding list to track which type of words the citizen shamelessly wrote. After *next_word* has returned false and there is no more to read from standard input the program checks to see if the badspeak and the oldspeak list have values in them. If just the badspeak is populated, then the badspeak message is printed along with the list. If just oldspeak is populated then the oldspeak message is printed along with the list. If both are populated the mixed speak message is printed and then badspeak and then oldspeak and translations. If neither list is populated then the program returns zero and prints nothing.