Jason Gallavin

Assignment 5

Oliver.txt with BST

11/08/2015

Abstract

The text file oliver.txt has many spelling errors in it. The writer needs a quick way to find out how many words he spelled correctly and how many words he spelled incorrectly. He has tasked us to build a fast program to spell check his document. Because the writer really wants the spellchecker to be fast, he also would like us to output the average comparisons of correctly spelled words and average comparisons of incorrectly spelled words.

The program first creates 26 binary search trees, one for each letter. It will then read the dictionary file into the trees by their first letter. All of the input words are capitalized so that the word is not case sensitive. After the dictionary is loaded into memory, the program reads oliver.txt. The parser reads the character by character. If the character is a letter, it is added to the current word. If the character is not a letter then the current word is searched in the dictionary. The search find the binary search tree that corresponds to the first letter searches. The binary search tree keeps track of the comparisons it makes so that when the word is found the comparisons are added to the total comparisons for words found and vice versa. Once the program finished reading the file, average comparisons are calculated by dividing the total comparisons by the words found or the words not found.

The average comparisons in general are much less than Assignment 2 and 4. The binary search tree is optimized so that you will not have do as many string comparisons. Because there is much less comparisons to be made the time that it takes to check the dictionary is much less. This run took on average 750 milliseconds when the linked list Assignment took about a minute. And the assignment 2 took even longer. The reason the binary search tree has a lot less comparisons is the way the structure is set up. A linked list is just a linked chain that you have to traverse all the way through up until you find the word. If the word is not found, you had to traverse the entire chain or all the words in the dictionary. A binary search tree works differently. The tree has levels. The tree starts with a word as the root and then has a word to the left and right. Left is lesser and right is greater. With this structure you do not have to traverse the entire list to find a word.

Outputs

```
Words Found: 939674

Words Not Found: 52466

Average comparisons of Words Found: 16.0

Average comparisons of Words Not Found: 9.0

comparisons of Words Found: 15284398

comparisons of Words Not Found: 515894
```