Landon Speer

Assignment Four

Programming Assignment Four

7/12/2015

Problem:

In this assignment you will implement a spell checker. Using the randomized dictionary (*random_dictionary.txt*) given in the list of files, read in the dictionary (words in the random_dictionary.txt file) into an array of 26 MyLinkedList objects, one for each letter of the alphabet. Thus the first Linked List would contain only those words starting with letter 'a', while the last would contain only those words starting with the letter 'z'. Notice that all words are in lowercase. Then, when you read in the book (*oliver.txt*), you examine the first character of each word, and traverse one of the 26 Linked Lists. Each word read from the book should be searched in the corresponding Linked List. If it is not found, this word is either misspelled, or not in the dictionary, then count it as a word not found. If the word is found, count it as a word found. You will also need to count the number of string comparisons that are done during search. Maintain two counters for this, one is for the number of comparisons for words that were found in the dictionary, and the other is for the number of comparisons for words that were not found in the dictionary. At the end of the program, compute and display the average number of string comparisons for words that were found and the words that were not found. (ex: average number of comparisons for words found = number of words found / total number of comparisons for words found, compute the average comparisons for words not found similarly).

Algorithms:

An algorithm for loading the dictionary, which takes in an array of MyLinkedLists, was created to take in words from the dictionary file and place them into an index of the array based on the value of the first character of the word. There is another algorithm which is the parser that reads each word from the Oliver text file. Based on the first character of the word it will search for the word in the array at the index associated with that first character. If that LinkedList contains the word then wordsFound will be incremented as well as compsFound will be added as a running total. If the word is not found wordsNotFound is incremented and compsNotFound will be added to. At the end of the method it will compute the averages for wordsFound and wordsNotFound by dividing compsFound by wordsFound and compsNotFound by wordsNotFound. There is also a toDisplay method that will display all of the data to the user at the end of the program.

Design:

The program is designed with three different methods. There is a loadDictionary, parser, and a toDisplay method. The loadDictionary loads the array of MyLinkedLists with words. The parser reads the Oliver file and checks if the words are in the linked lists or not and will increment the attributes respectively. The toDisplay method will display all the data after it has been computed at the end. In the main method the linked lists are instantiated before anything else can be done. Then the loadDictionary method is called, after that the parser is called, then finally the toDisplay is called.

Results/Observations:

The words are read from the files correctly and are placed in the correct linked lists. The program displays the data to the user at the end of the program and gives correct outputs.