CSC112 – Laboratory #6*

David John

Spring 2013

1 Overview

Given a certain statement it is an interesting to determine its relevance in relation to a number of texts. A statement such as

> The rate of change of a continuous function is known as the first derivative

is highly relevant to Stewart's *Single Variable Calculus*, but is not so for Larsson's *The Girl with the Dragon Tattoo*. In this lab we will develop one tool to assist in this type of relevance analysis.

The assembly of this tool will happen in three stages. In the **first stage** you will develop a class for a node, and a class to manage a singly linked list of these nodes. Each node will contain a string (in a normal form) and an associated frequency. There must be a method that allows the addition of a string to the linked list, either by incrementing the frequency when the word is found, or inserting a new node with an initial frequency of 1. In the **second stage**, you must develop a main program. The main program must interactively control the addition of text from a file into a linked list. In the **third stage**, a robust program will be delivered that can accept one phrase of target text, and up to ten texts used for comparison.

I will also provide several files of text that you may use for testing. Shortly, *telltaleheart.txt* and *lottery.txt* will be available from *sakai* resources.

*Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this document, and I am aware of a trademark claim, the designations have been printed in initial caps, all caps or italics.

2 Phase One

1. ListNode class

   Modify your current ListNode class to contain two class variables. One to hold a word of text, a second that will contain the frequency of occurrence associated with that word.

2. List class

   Modify your current List class to contain another class variable that will hold the total number of all words in the list. Unless you feel there is a need for the current *rear* pointer, you may omit it. Add a List method that will return the address of a specific word in the list or *null*. Include a method which, given a string, updates the frequency of that string in the list. The words stored in the list must be distinct and must be stored in alphabetical order.

3 Phase Two

1. Main Program

   Add to the main program a command such as

   add myfile.txt

   which will result in all the words in *myfile.txt* being inserted into the list.

2. List class

   Add a *value(string)* method that will (1) find *string* in the list, and (2) compute the entropy of the string relative to list. The entropy of *string* is $-prob(string)\log_2(prob(string))$, where $prob(string)$ is the probability of *string* occurring in the list. There is an interesting case that we will need to discuss, and you will need to resolve. You also will need to remember that $\log_2(x) = \frac{\log(x)}{\log(2)}$. The natural logarithm function, $log(x)$, is part of *cmath* (you will need to include *cmath*).

3. Main program

   Add to the main program a command such as

   value phrase

which will compute and return the sum of all the entropies of the strings in the phrase. We will talk about how to get this done in class.

4 Phase Three

In this phase you will first put all this together as a working program that uses up to ten different texts. Once this works you want to use up to ten different *named* texts, this will give the program a really good feel to the user.

To implement up to ten different texts you should create an array of physical size 10 in your main program where each cell has type *List*.

You will need to modify your *add* interactive command to be something like:

add fname.txt

which would cause the contents of *fname.txt* to be inserted. You will need to modify your *value* interactive command to print all the information scores for the *phrase* for each of the lists present. The interactive command *delete* should be appropriately modified.

5 Some specifics

The basic assignment is worth a maximum of 100 points.

You must appropriately comment your program. You must use a style that makes your code easy to read.

This is a *pledged work* assignment. Outside of our designated lab period, you may discuss your work only with me.

This assignment is due on Friday, April 21, 2013.