CS273 Module Project 5

Using associative containers for a computational linguistics project

# Objectives:

By the end of this project you should:

1. Have familiarized yourself with the non-sequential (associative) data types available in the STL
2. Utilized your knowledge to build a non-trivial application: doing statistical analysis on a Gutenberg E-Book!

# Due dates:

November 14

# Instructions:

**Step 1:** Using the slides, the textbook and a good reference to the STL, familiarize yourself with the associative containers, especially **sets**, **multisets**, **maps**, and **unordered\_maps**.

**Step 2:** Design a data structure / application that uses one of the associative containers from step 1. Your application should:

* Read in a plain text (ASCII) file, splitting up that file into its constituent words. (You may simplify this by treating a word as any sequence of characters buffered by whitespace.)
* Store these words in an STL associative data structure of your choice.
* Keep tracks of how frequently each word occurs.
* Enable queries to your data-structure / application, see below.

**Step 3:** It’s time to learn something about language! Go to the Project Gutenberg website (<http://www.gutenberg.org/>) and find an interesting ebook. Download that book in plain text form (UTF-8). Process the book so that all its words are associated with the number of times they occur. Then, go and find all the words that occur: once, twice, three times,…,N times. (I recommend that you parameterize your application so that its human user can pick N, but that when YOU use your application you stop at some small but still interesting number, like 10.) Display a table showing your words by frequency. Write a paragraph or two describing a hypothesis explaining how your author’s choice of words changes as their frequency goes up, then submit your application together with your report.

**A big hint:** For efficiency and ease of use, you might find it easier actually to make and use TWO associative structures. Create your first structure by passing through your e-book. This structure stores each word with its frequency. Then make a second pass by iterating over that data structure to create your table, i.e. gather up and store in one sub-collection all the words that occur once, all the words that occur twice, etc.

**More hints:**

* I could imagine some use to encapsulating lots of your functionality within a class. For example, you could write a class that represents a book internally as one or two containers (that may or may not be initialized) and an ifstream (to hold open a plain text file).
* This code may not be extensive at all! If things start getting too long, something about your approach might be off.

## Deliverables

* Your completed application as source code
* Documentation….:
  + Illustrating your development process including UML (if you developed a class, see “More Hints” above), pseudocode for all procedures, and use cases.
  + Unit tests for crucial units.
  + A README showing how to use your application on a fresh e-book.
  + A report showing a table of word classes, e.g. all words that occur once, twice, etc.,….up to N times. You should also write two or three paragraphs of observations about these words including a hypothesis as to how the words change as they become more frequent.

## Extra credit opportunity!

For projects that work correctly, an extra +15% can be obtained by researching and making decent use of exception handling in this project.

**Good luck and have fun!**