University of New Brunswick, Faculty of Computer Science

CS 2263 Assignment 3 Winter 2018

Due Date: Friday, Feb. 16, 11:59 pm

Purpose: Introduction to file I/O and string tokenization

Movie Review Sentiment Analysis

Sentiment Analysis is a Big Data problem that seeks to determine the general attitude of a writer given some text they have written. For instance, we would like to have a program that could look at the text "The film was a breath of fresh air" and realize that it was a positive statement while "It made me want to poke out my eye balls" is negative.

One algorithm that we can use for this is to assign a numeric value to any given word based on how positive or negative that word is and then score the statement based on the values of the words. But, how do we come up with our word scores in the first place?

That's the problem that we'll solve in this assignment. You are going to search through a file containing movie reviews from the Rotten Tomatoes website which have both a numeric score as well as text. You'll use this to learn which words are positive and which are negative. Each review starts with a number 0 through 4 with the following meaning:

0. Negative
1. Somehat negative
2. Neutral
3. Somewhat positive
4. Positive

The data has been formatted to make it easy to identify each word, with each review given on one line, which begins with the score:

```
1 A series of escapades demonstrating the adage that what is good for the goose is also good
4 This quiet , introspective and entertaining independent is worth seeking .
...
```

This assignment will be done in two stages. First you will compute the average score for a given word, and then you will determine the score for a given movie review.

1. Word Score

Write a C program to ask the user to enter a word, and search every movie review for that word. Each time you find it, add the score for that review to the word's running score total. You also will need to keep track of how many reviews contain the word so that you can report the average score of reviews containing that word back to the user. The movie reviews are stored in a file whose name is passed as a command line argument.

Here are two example runs:

```
$ ./sentiment1 movieReviews.txt
Enter a word: rain
rain appeared 4 times
The average score for reviews containing rain is 1.500000
$ ./sentiment1 movieReviews.txt
Enter a word: awful
awful appeared 23 times
The average score for reviews containing awful is 1.086957
$ ./sentiment1 movieReviews.txt
Enter a word: fantastic
fantastic appeared 13 times
The average score for reviews containing fantastic is 2.846154
```

The movie review file is provided in D2L. Your program must include the following function:

```
void getWordStats(char *word, FILE *f, int *sum, int *num);
```

which searches for the string `word` in the file `f` (a pointer returned by a call to `fopen` in `main`), and computes the total score of reviews containing the word and the total number of reviews that contain the word. These two values are returned by reference (`sum` and `num`).

2. Movie Review Sentiment Analysis

Write a C program that reads a review entered by the user on a single line, and estimates the sentiment of the review. The sentiment is determined by computing the average score of the words in the review. The review is labelled as positive if the average score is greater or equal to 2.0 and negative otherwise. Here are some sample program runs:

```
$ ./sentiment2 movieReviews.txt
Enter a review:
programming is the most exciting job
The review has an average value of 2.332416
Positive Sentiment
```

```
FCS-OSX:A3 gdueck$ the rain fell on the dead killer
-bash: the: command not found
$ ./sentiment2 movieReviews.txt
Enter a review:
the rain fell on the dead killer
The review has an average value of 1.603154
Negative Sentiment
```

You can reuse much of your code from question 1, including the `getWordStats` function. After you search for the first word, you will need to rewind the file before searching for each following word. This can be done using `fseek(f, 0, SEEK_SET)`, where `f` is the file pointer (see class notes on `make`, as well as the textbook). In order to parse the words in the review, you can use the `strtok` function in `string.h`:

```
char *strtok(char *restrict str, const char *restrict sep);
```

On the first call, the first parameter is the string to be tokenized, and the second parameter is a string of characters that can be used to separate tokens; for example for comma-separated values this second string would be `","`. On subsequent calls to get further tokens of the same string a null pointer should be passed instead as the first parameter. From the `man` page (do `man strtok` in the terminal): `strtok returns a pointer to the beginning of each subsequent token in the string, after replacing the token itself with a \0 character. When no more tokens remain, a null pointer is returned.`

Create the following files:
- Word_Stats.h
- Word_Stats.c
- mainq1.c
- mainq2.c
- Makefile

The make commands must be supported:
- **make sentiment1** create the executable for question 1
- **make sentiment2** create the executable for question 2
- **make clean**

To pass in the assignment: Create a single pdf document with your code listings and terminal session for both questions, and a zip file with source files for both questions. Submit these files to the Desire2Learn dropbox. Name your documents `LastName_FirstName_As3.pdf` and `LastName_FirstName_As3.zip` (`LastName` and `FirstName` are of course substituted with your last and first name).