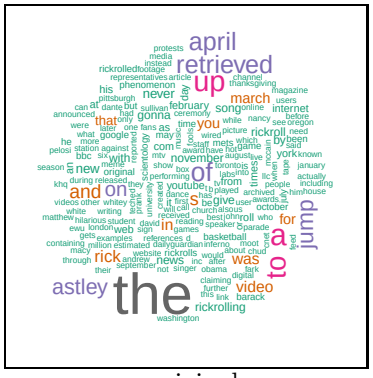
CSC 112 Lab 8
Arnav says "Stemming ain't easy"
Spring 2015

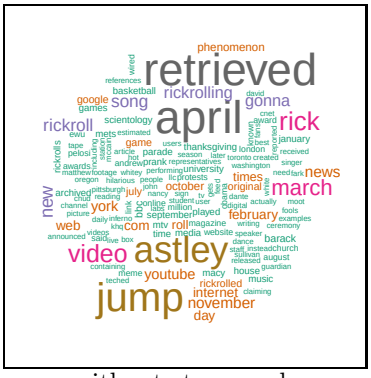
Due
3:00pm Friday, 4/3

1 Introduction

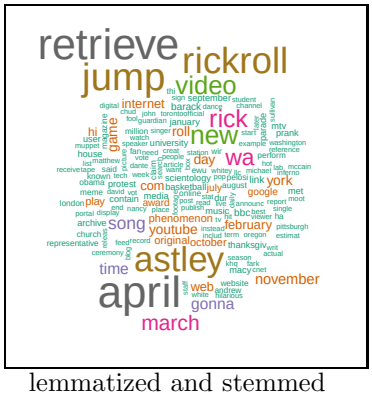
In the previous lab, certain or uninteresting or common words (called *stop words*) were filtered out of a text document to provide a better word cloud representation. In this lab, *stemming* and *lemmatization* processes will be applied to provide an even better word cloud representation of the document. Stemming and lemmatization are processes that group together the different inflected forms of a word so they can be analyzed as the same word. For example after the process "rickroll," "rickrolls", "rickrolled", and "rickrolling" would simply become "rickroll." But as Arnav says, this ain't easy due to the oddity of the English language.



original



without stop words



lemmatized and stemmed

1.1 Word Frequency after Stemming and without Stop Words

Similar to lab 6, we are interested in determining the word frequency (a count of the number of times a word is used) of a text file that has been stemmed and without stop words. The program will accept three command line arguments; the text file name, the stop words file name, and the resulting word frequency file name. If the user does not provide the three arguments, then the program should stop and display how to properly execute the program (explaining the command line arguments and the order). Similarly, if any of the files cannot be opened, stop the program and explain the error. If the arguments are correct, the program should read the text file (first file argument) and process every word. Once the frequency has been determined, print the original number of words found to the screen (not stemmed and with stop words). Afterwards apply the stemming process and display the new number of words in the list. Then process the stop words file (second file argument), remove all the stop words from your list, and redisplay the word count. Finally, write the final list **in alphabetical order** to the frequency file (last argument file) and indicate this on the screen. For example, assume the user wishes to process `roll.txt` as the text file, `stop.txt` contains the stop words, and `roll.frq` is the resulting file. The following would be the result.

screen output

Terminal

```
> ./lab8 roll.txt stop.txt roll.frq
roll.txt has 1237 unique words
-----------------------------------------------
after stemming roll.txt has 1115 unique words
-----------------------------------------------
without stop words (read from stop.txt)
roll.txt has 966 unique words
-----------------------------------------------
Creating roll.frq ...  done!
```

`roll.frq`

```
abc 1
abus 1
academic 1
accord 1
acknowledge 1
act 1
actual 4
ad 1
adam 2
add 2
⋮
```

2 Lemmatization, Stemming, and Google

Lemmatization (or lemmatisation, if you are British like Arnav) in linguistics is the process of grouping together the different inflected forms of a word so they can be analyzed as a single term. Stemming is similar to lemmatization, since the process reduces inflected (or sometimes derived) words to their word stem, base or root form. Algorithms for these processes have been studied in computer science since the 1960s. Interestingly, many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation. We will use a very simple lemmatization and stemming processes to reduce the list of words read from the text file.

2.1 Lemmatization and Stemming Process

For this lab assignment, we are primarily interested in removing plurals and tenses from words in our text. Use the following rules to adjust the words in your list, where rule order does matter.

If the ...
word is at least 6 characters long and ends with "ies" then replace "ies" with "y"
word is at least 5 characters long and ends with "ves" then remove the final "s"
last letter is "s" but the second to the last letter is not "s" or "u" then remove the final "s"
word is at least 5 characters long and ends with "ved" then remove the final "d"
word is at least 5 characters long and ends with "ed" then remove the final "ed"
word is at least 7 characters long and ends with "ly" then remove the final "ly"
word is at least 5 characters long and ends with "ing" then remove the final "ing"

Note, the above rules are not perfect and may result in an incorrect word. For example, "addresses" will become "addresse" and "boxes" will become "boxe," which is acceptable for this assignment.

3 Program Design

As done in lab 7, managing the word frequency list **must be dynamically allocated** such that there is no wasted space (logical and physical size are always equal). In addition, your program **must** adhere to the following program design requirements.

3.1 Operator Overloads for `WordFreq`

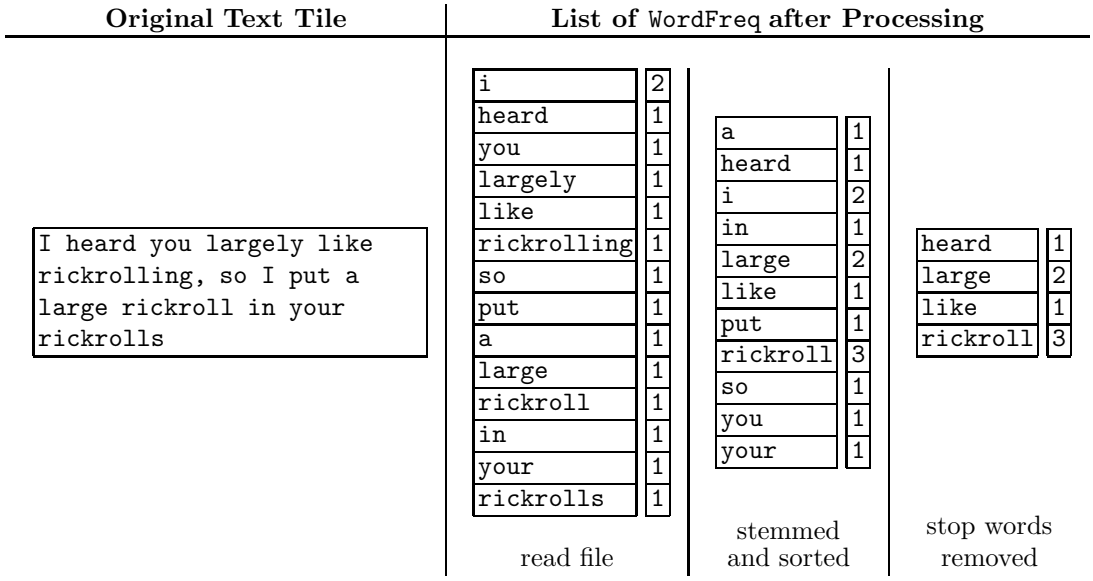
This lab assignment will continue to use the `WordFreq` `struct` from lab 6 to store a word and the frequency. However, you **must** overload and use the following operators.

- `operator<<` should print the word and frequency separated by a single space
- `operator==` should return `true` if the lefthand and righthand `WordFreq`s (`word` data member) are the same, return `false` otherwise
- `operator>` should return `true` if the lefthand `WordFreq` (`word` data member) would appear after the righthand `WordFreq` given a lexicographical comparison, return `false` otherwise.

3.2 Operations for Processing the File

There are 4 list processing steps your program will perform before writing the final frequency file. First, your program will read words from the file then store the unique words and the corresponding frequencies. Next your program should stem (using the rules described in the previous section) and combine any equal words. As seen in the diagram, the example file contains the words "rickroll," "rickrolling, " and "rickrolls." Each of the words occurs once, but after stemming these words become "rickroll" and the count is 3. Once

stemming is complete, sort the list alphabetically. Finally, remove all the stop words from the list and print the resulting list to the frequency file. The processing steps are depicted in the following diagram.



3.3 Multiple Files and `makefile`

Copy the `words.h` and `words.cpp` files from lab 7 into your lab 8 directory. You will update these files with the `WordFreq` operator overloads and alphabetical sort. You will create 4 new files for this lab assignment.

- `main.cpp` contains the main function.
- `words.h` contains the updated word function prototypes (declarations).
- `words.cpp` contains the updates word function definitions.
- `stemming.h` contains the stemming functions function prototypes (declarations).
- `stemming.cpp` contains the stemming function definitions.
- `makefile` will `make` the project (`lab8` executable) and will include a `make clean` option.

4 Programming Points

You **must** adhere to all of the following points to receive credit for this lab assignment.

1. Create a directory `Lab8` off of your `CSC112` directory to store your program files
2. The assignment will consist of 5 files described above.
3. Your program must be modular in design.
4. Your `main` function can only consist of variable declarations, function calls, and control structures (no input or output in the `main` function).
5. Your program must compile cleanly, no errors or warnings are allowed.
6. Your program must adhere to documentation style and standards. Don't forget function headers and variable declarations.
7. **Turn-in** (copy to your `Grade/Lab8` directory) a word cloud png (image file) of the `wakebaseball.twt` text file stemmed with the stop words removed.
8. **Turn-in** a print-out of your program source code. In addition, copy your program source code to your `Grade/Lab8` directory.