Programming Project 6

Assignment Overview
This assignment will give you some experience with dictionaries, as well as some more practice with file I/O. This assignment is worth 40 points (4.0% of the course grade) and must be **completed and turned in before 11:59 on Monday, October 22.**

The Problem
A common element seen on web pages these days are tag clouds (http://en.wikipedia.org/wiki/Tag_cloud). A tag cloud is a visual representation of frequency of words, where more frequent words are represented in larger font. One can also use colors and placement.

We are going to analyze the presidental debate transcript and create a tag cloud for each candidate of the words they used, where the frequency of the words indicates the size of the font in the cloud.

Background
We provide a number of elements to help with this task:

- a transcript of the debate
- a list of stopwords
- some functions

Transcript
The transcript is in **debate.txt**. It has a particular format. Each time one of the candidates speaks, that line is marked, e.g 'PRESIDENT OBAMA:'. Once encountered, all words are attributed to that speaker until another label occurs (sometimes it is a question from the moderator so you have to ignore those). Take a look at the file.

Stopwords
Not all words are worth counting. 'a', 'the', 'was', etc. are just junk. A list of such words is provided as **stopWords.txt** Each line has a single word. No word in the stop word list should be counted in the tag cloud. This is the MySQL 4.0.20 list with additions by me (mostly just duplication of contractions. That is both "can't" and "cant" are now in the list).

Functions
Three functions and an example are provided in **htmlFunctions.py**. Use them in your program. That file contains:

- `make_HTML_word (word, count, high, low)`: This function takes a `word` and wraps it in a font tag with a specific size—returning that string whose `fontsize` is between `htmlBig` and `htmlLittle` (two local vars in the function that you can change to be whatever you like). The parameters are:
    - `word` (string), the word to be wrapped,
    - `count`, how many times it occurred in the document (for each candidate),
    - `high`, the highest word count and

- `low`, the lowest word count of words being processed for this candidate

The function returns the word as a formatted string.

- `make_HTML_box(body)`. This function takes a single string named `body` that contains all the font-wrapped words from `make_HTML_word` and places them in an HTML box to be displayed. It returns a string which is the HTML code for the box.
- `print_HTML_file(body,title)`. Takes the `body` (string) returned from `make_HTML_box`, wraps a standard HTML web page around it, and creates a file. The string `title` is used in the HTML. The title is also used as the file name with an '.html' suffix. The file is created and written.

Program Specs

- Prompt the user for the transcript file to be processed.
- Print the top 40 counts (word and count pairs) for each candidate. Sort the words by count for printing. (For the next step you will sort them alphabetically for the HTML files.)
- Generate two HTML files (obama.html, romney.html) using the provided functions to generate a tag-cloud file with the top 40 words each used in the debate. For the tag cloud sort the top 40 words alphabetically—the tag cloud looks more interesting that way.You can view the files in your browser.

Deliverables

proj06.py -- your source code solution (remember to include your section, the date, project number and comments).

1. Please be sure to use the specified file name, i.e. "proj06.py"
2. Save a copy of your file in your CS account disk space (H drive on CS computers).
3. Electronically submit a copy of the file.

Assignment Notes:

There are a couple of problems here. Think about each one before you start to program.

1. Parsing the debate file. You have to read in the file and separate the lines according to who said them: Obama, Romney, etc. Use the file format to help you with this. Remember, once you see one of the speakers' tags ('MR ROMNEY:') all lines/words belong to that speaker until you see another speaker's tag.

2. Once you have the words separated, you must remove all stop words (using the provided file "stopWords.txt". You can do this while you are collecting the words as well, but you must remember that you have to read in the stop words from the file and remove any stop word from a candidate's words. Also, remember to remove punctuation from words, just because a word comes at the end of a sentence and has a period at the end of it doesn't make it a different word. (Importing `string` and using `string.punctuation` is a useful way to specify punctuation.)

3. You must then count the word frequency in the candidate's words. Use a dictionary, where the key is the word and the value is the count. Section 9.2 (page 391) of the text will be very helpful for this part.

4. Once you have a dictionary for each candidate, you must extract the 40 most frequently used non-stopwords and their counts. Since a dictionary is unordered you need to create (count,word) tuples, put them in a list, and then sort the list. Put count first in the tuple because sorting (using either `sort` or `sorted`) will sort on the first item.

5. Use the provided functions to turn the words and counts into an HTML page (see the **htmlFunctions.py** example in the file).

Getting Started

1. Do the normal startup stuff (create proj06.py, back it up to your H: drive, etc)
2. Write a high-level outline of what you want to do. Think before you code.
3. Read in the stopword list and store it. That should be easy (maybe use a function to do it ????)
4. Start by trying to parse the file into its three parts: Obama, Romney, everything else (junk). Make it easy on yourself, create a little file from the top of the whole file (just enough with lines from each person twice) so you can see what's going on (or make a small test file). Once done, try it out on the big file.
5. Try to do a count of one candidate's words in a dictionary. Again, use a small file so you can look at the results. Can a function help here too?
6. Try to extract the top counts in a dictionary.
7. Take a list of words and counts (make some up to start with if you like) and make a web page using the provided functions. Now take words and counts from a candidate dictionary and do the same.

Stuff to think about

1. Some problems arise because our program cannot tell the difference between 'american' and 'americans'. This is a stemming problem. There are ways to address this issue, but it is not required for this assignment.

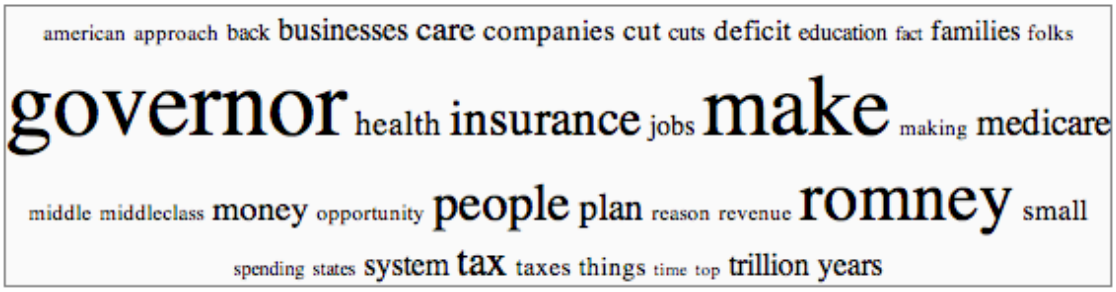
2. Once you have your tag clouds, take a look at them. What do they tell you about the candidates?

Sample Ouput:

1. First is what is printed to the shell (note: sorted by count)
2. Next are what the .html files look like in a browser (note: sorted alphabetically)

```
++++++++++++
Obama : words in frequency order as count:word pairs
 48:governor       44:make           35:romney         26:people
 25:insurance      21:tax            19:plan           19:medicare
 18:money          18:care           17:health         16:years
 16:trillion       16:system         16:businesses     15:small
 15:deficit        15:cut            15:companies      14:jobs
 14:families       13:things         13:taxes          12:education
 12:cuts           12:back           11:revenue        11:reason
 11:opportunity    11:middleclass    11:middle         11:making
 11:folks          11:approach       11:american       10:states
 10:spending       10:fact            9:top             9:time
  9:seniors
++++++++++++
Romney : words in frequency order as count:word pairs
 73:people         39:tax            36:president      35:plan
 33:government     32:cut            31:number         31:medicare
 23:taxes          22:state          22:percent        22:jobs
 22:care           22:america        20:work           19:make
 18:years          18:put            18:billion        17:health
 17:businesses     16:small          15:lower          15:cost
 15:bring          14:business       14:back           13:year
 13:time           13:rates          13:place          13:million
 13:idea           13:federal        13:economy        12:regulation
 12:rate           12:jim            12:insurance      11:trillion
 11:schools
```

Obama



Romney