Intro to Computer Science

PA10

Creating Word Tags/Clouds of a Speech

Code due by Wednesday, April 23rd at 11:00 AM

Paperwork due the same day at the start of class

Assignment Overview

The goal of this project is to 1) gain experience using pieces of code written by someone other than yourself, 2) test your new top-down design skills in breaking a large problem down into steps and helper functions, and 3) gain more practice with file I/O, dictionaries, lists, tuples, and functions.

Background

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 a presidential debate transcript and create a tag cloud for each candidate based on the words they used, where the frequency of the words indicates the size of the font in the cloud.

To help you with this assignment you are provided with the following documents:

- A transcript of a debate (take your pick)
  - The October 3, 2012 Obama-Romney Presidential debate. (first.txt)
  - The October 16, 2012 Obama-Romney Presidential debate. (second.txt)
  - The October 22, 2012 Obama-Romney Presidential debate. (third.txt)
  - For testing purposes only, a partial transcript of the first debate that only contains the first five minutes or so of the debate is given. This is still big, but it might give you a chance to print some stuff to the screen without flooding the system when you are testing your code. (partial.txt)
- A list of stop words (stopSQL.txt)

- Some helper functions (htmlFunctions.py)

Each of these files is explained below:

- **Transcript** - Open up the file for the debate you picked. Each transcript is in a particular format. Each time one of the three participants speaks, that line is marked with the speaker's name followed by a colon ('OBAMA:', 'ROMNEY:' or 'LEHRER:' for the first of the debates). Once encountered, all words are attributed to that speaker until another label occurs. Notice that this may not be for several lines.

- **Stopwords** - Not all words are worth counting. In the context of speeches, 'a', 'the', 'was', etc. are just junk. A list of such words is provided as stopSQL.txt Each line has a single word. No word in the stop word list should be counted in the tag cloud. This is the list distributed with MySQL 4.0.20 list with a few additions (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:
    - makeHTMLword (word, cnt, high, low)

      This function takes a word and wraps it in a font tag with a specific size. The function takes the word to be wrapped, how many times it occurred in the document, the highest word count and the lowest word count of words being processed (the highest count we are considering for this tag and the lowest). It returns a string that is the word and fontSize between htmlBig and htmlLittle (two local vars in the function. You can change them to be whatever you like)

    - makeHTMLbox(body)

      This function takes a single string of all the font-wrapped words from makeHTMLword and places them in an html box to be displayed. It returns a string which is the html code for the box.

    - printHTMLfile(body,title)

Takes the body returned from makeHTMLbox and wraps a standard html web page around it. The string title is used in the html. The title is also the file name with an '.html' suffix

Play with this file for a few minutes until you see how it works. You do not need to understand all of the details, but you need to understand what each function does and how they work together.

Project Specifications

In a file called pa10.py, you will need to write a group of helper functions called by a mater function (named **main()**) which will read through one of the debate files, create a dictionary of words spoken by a specific speaker, remove stop words, identify the 40 most frequently used words by that speaker, and use that information to call the helper functions provided to in the file htmlFunctions.py to create the word cloud. Your code should include **no less than 3 new helper functions** (other than main()) to illustrate that you understand how to break large problems into small, managable steps. Of course, you can create more than 3 new helper functions if you like.

The master function main() will take in two parameters:

- A string representing the name of the debate text file
- A string representing the speaker that you wish to analyze. This string should be the exact name/label used in the debate transcript (without the colon. See below)

For example, if I invoke:

main("first.txt","ROMNEY")

my code should produce a file called ROMNEY.html which is the word cloud spoken by that candidate in the first 2012 debate. BTW, that file should look like this one (ROMNEY.html)

If I invoke:

main("first.txt","OBAMA")

my code should produce a file called OBAMA.html which is the word cloud spoken by that candidate in the first 2012 debate. BTW, that file should look like this one (OBAMA.html)

CHECKING YOUR WORK. If you run the commands above on your finished code and the html files look different from mine, then something isn't quite right for one of us. It COULD be me who is wrong, but you should ask questions.

Helpful Hints:

1. **Create your design diagrams and documents first! Break down all the steps that need to be performed, and figure out which functions they go into.** (Note: You do not have to turn in a design tree, but it will probably be useful to you to create one.)
2. Parsing the debate file. You have to read in the file and separate the lines according to who said them: OBAMA, ROMNEY or LEHRER. Use the file format to help you with this. Remember, once you see one of the speaker tags all lines/words belong to that speaker until you see another speaker tag.
3. You have to remove the stop words. You have two choices. Either do it as you are reading the debate or go back and remove them once you are done reading the entire debate. Each has advantages. Pick one and go with it.
4. 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. Also, you don't need to count the audience words inside of parenthesis like (APPLAUSE) and (LAUGHTER). You can tell audience words apart from regular speech because they occur on separate lines and are surrounded by parenthesis.
5. Capital letters are a potential problem. If we aren't careful the word "Economic" at the start of a sentence will be counted separately from the word "economic" inside of a sentence. For simplicity I will allow you to treat all words as lower case letters (having said that, be careful **when**you decide to convert to lowercase letters since speakers are labeled with all caps ("OBAMA:") and you may want/need to use this information.
6. Count the word frequency in the candidate's words. Use a dictionary, where the key is the word and the value is the count.

7. Once you have dictionary for the candidate in question, you need to extract the 40 most frequently used non-stopwords and their counts. This should be familiar. You will need to use lists of tuples as we have done before.
8. We also need to extract the biggest count and the smallest count from the words in this top-40 as that information is needed by makeHTMLword ()
9. Once you have the forty most frequent words (and their counts) we want to alphabetize that list. That sounds like another set of tuples in a list.
10. Finally, use the code presented to you in htmlFunctions.py to generate the html file with the appropriate name.

Final Submission

To upload your homework for grading, log on to eLearning, select this class, and navigate to the "Assignment Submissions" area. Click on the "Programming Assignment 10" folder and upload the python file in its designated location.

In addition to this, you should print paper copies of your code and design document. Please submit these stapled printouts in the following order in class the day the assignment is due:

- design document
- pa10.py