CSE 231 Fall 2008

Programming Project #7

Assignment Overview

This project focuses more on using functions and file reading. It also introduces dictionaries. It is worth 50 points (5% of your overall grade). It is due Monday, Oct 20th before midnight.

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 Palin-Biden 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 three participants speaks, that line is marked with one of 'IFILL:', 'BIDEN:' or 'PALIN:' . Once encountered, all words are attributed to that speaker until another label occurs. 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:

- 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 (palin's part, biden's part), 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

Program Specs

- Prompt the user for the file to be processed
- print the top 40 counts (word and count pairs) for each candidate
- Generate two html files (palin.html, biden.html) with the to 40 words each used in the debate as a tag cloud

Deliverables

proj07.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. "proj07.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: Palin, Biden or Ifill. Use the file format to help you with this. Remember, once you see one of the speaker tags ('IFILL:', 'PALIN:', 'BIDEN:') all lines/words belong to that speaker until you see another speaker tag.

2. Once you have the words separated, you must remove all stop words. 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.

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.

4. Once you have a dictionary for each candidate, you must extract the 40 most frequently used non-stopwords and their counts.

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 proj07.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. 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. 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 them up to start 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

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. If you are interested, contact Punch for extra credit on dealing with this issue.