# CS 1110 Spring 2022, Assignment 3 optional extension: Try your A3 code on real SOTU data!*

You don't need to submit anything for the activities suggested in this writeup! They are just for your amusement, Python empowerment, and perhaps your interest in doing some computational analysis on real, politically significant human-language data. When we release solutions for A3, we'll tell you what results we got.

Download and unzip the following into the directory in which your working A3 code lies: http://www.cs.cornell.edu/courses/cs1110/2022sp/assignments/assignment3/real_sotus_files.zip

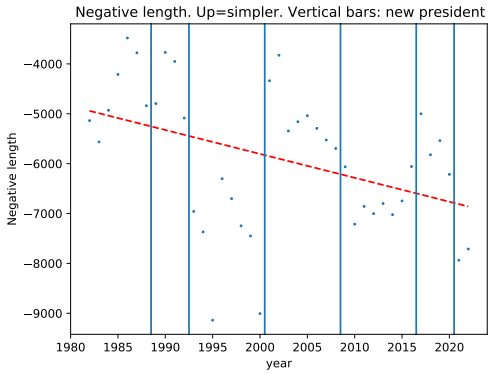
File sotu_dict.py defines a dictionary where the keys are years, in ascending order, and the values are tokenlists for the U.S. Presidential State of the Union addresses (SOTUs) for those years, going all the way back to 1791.[1]

The file analyze_sotus.py runs your A3 analysis code and plots the resultant data. Note that we set variable START_YEAR to 1982 rather than 1791; this is because in years prior to 1982, some SOTUs were written and some were spoken, and we wanted to eliminate that confound.[2]

## 1 Basic run of the analysis code

Navigate in the command shell to the directory your A3 code and the real SOTU data is, and run the script analyze_sotus.py. The code does three analyses:

- Length: you'll get a printout, generated by **your** print_trends() function!, of the negative[3] lengths of each SOTU. Those numbers should match what is in file sotu_len.txt. The code also produces a visual plot of the negative-length data in file sotu_len.pdf; if your print_trends() function is right, your generated file should look like this:




  The red dotted line is a linear fit to the data, So if we use negative length as a measure, it seems like recent SOTUs are getting *less* simple, in contrast to what was asserted by the motivating quote for the main A3.

  Vertical lines demarcate changes in who was the U.S. President. Observe that different presidents[4] seem to have distinct length preferences, for the most part.[5]

*Authors: Lillian Lee

[1] Except that we omitted years where there were two or more SOTUs.

[2] But you can change the START_YEAR variable to be whatever you want!

[3] Negating the lengths means that larger values correspond to greater simplicity, for consistency with the other functions considered in A3.

[4] Or at least their speechwriters.

[5] You may find it interesting to note that our results are quite close to, but not quite the same as, the length statistics produced by the American Presidency project. They describe their (more-careful-than-ours) processing of tokens as follows:

> Length in words is computed by using the word count feature in Microsoft Word. The APP first uses the transcript provided by the White House, and then provides a modified word count about a week later after the official transcript is provided by

- Type-token ratio: the code will next print out and plot your type_token_ratio() function's results. The visual plot will be in file sotu_ttr.pdf. According to this measure, is there a trend towards greater simplicity over time?
- ~~Finally,~~Next, the code then calls your percent_avg_or_shorter() function, and prints/plots the results, with the visual plot in file sotu_percent_short.pdf . What conclusions can you draw?

- For function zipf(), we have the issue of needing to pick a value for parameter `n`. Our code prints/plots the results for the somewhat arbitrary choices of `n` in `[10, 50, 100, 500]`; the visual plot will be in file sotu_zipf.pdf.

## 2 Explore more?/!

Perhaps looking at your plots makes you curious to develop other measures of language complexity or check whether there are confounding effects to the ones we've proposed, or even just look at other years of SOTUs. Use your knowledge of Python and the dictionary we built for you in sotu_dict.py to investigate to your heart's content!!

## 3 Want to apply these ideas to other data, and wonder how to access and process such data?

If you're interested in how we set up the data — in particular, in case you'd like to do similar data exploration on your own — , take a look at file grab_sotus.py. We wrote it to be mostly understandable by CS1110 students, at least by the end of CS1110; and much of it should be understandable, with some effort, to you here in mid-March 2022. Feel free to ask questions about this code on Ed Discussions!

the National Archives in the "Daily Compilation of Presidential Documents". The following elements are removed from the transcript prior to generating the word count: 1) audience reaction notations such as "applause", etc.; 2) Prompts indicating a change in the speaker; 3) All audience comments included in the transcript; 4) All "em dashes" between clauses in sentences; 5) All words in brackets that reflect a White House correction to the word actually spoken; and 6) all HTML or other coding to produce paragraph breaks, among others. The word count is then generated using a single blob of words separated only by spaces and normal punctuation.

For example, our program does not strip out audience interjections, some of which have been considered newsworthy.