

This week’s homework and activity are intended to give a ‘get-acquainted’ experience with a few elemental data tools and concepts: plotting, correlation, linear regression, and quantitative analysis of large amounts of text—all using short programs and modules written in Python. No past programming experience in Python is needed—and the required portion is shorter than last week’s SQL activity. The goals this time are having fun playing around and discussion of data policy issues. The discussion includes what it means to build and train a *predictive model*. This activity needs **Chrome** or **Firefox** or a downloaded Python system; the two browsers seem to work on any platform.

This part belongs to a genre whose most immediately recognizable example may be the mass filtering of communications and webpages for terrorist indications and gauges of threat levels. “Hot words” and combinations of words are assigned numerical *scores* in various categories of threats and implications by the models underlying these filters. Another prominent example is the Google Ngram Viewer, as mentioned in lecture. It counts occurrences of *N*-word phrases in Google’s entire collection of digitized books (searchable through 2008) but does not give scores. A third level is represented by various models of *stylometry* including Signature by Peter Millican of Oxford University.

We will use the recent August 2022 update to the full *version 1* (March 2020) of the Canadian *National Research Council Emotion Intensity Lexicon* (NRCEIL). It has almost 6,000 “intense” words scored in 8 categories of emotion. An earlier version named for “Affect Intensity” used only four categories: *anger*, *fear*, *joy*, and *sadness*. We will add *disgust* from the new version, as that rounds out the five emotions from the the 2015 Pixar movie *Inside Out*, but skip the other three (*anticipation*, *surprise*, and *trust*).

We will use this tool to make simple measures of the “emotional temperature” of various webpages. How to choose a set of measures based on scientific justification rather than personal whim—and to focus *what* is being *modeled*—is for discussion at the end. This also serves as a rudimentary example of “programming the Internet.” Here’s what’s involved:

- **Homework** to complete *individually* before your recitation hour, including showing a score on the game guessthecorrelation.com and running regressions on NFL data.
- Allowing time for troubleshooting Python on individual machines, the *running* part is short.
- Emphasis is on discussion and interpretation, so again please form groups of 3-or-4.
- The project files are those beginning with **heat** in the CSE199 repository for this unit. They are duplicated in my own CSE199 <https://www.cse.buffalo.edu/~regan/cse199/> folder.

## 1 Preparation

The main technical requirement is access to a working Python 3 installation. Directions and options for this are the same as was given for the NFL-data homework.

Then please skim-read the 10-page “Word Affect Intensities” paper. You need not understand all the technical bits but should appreciate the following points:

- Even at professional level this kind of work is in early stages.
- This is almost the first lexicon with numerical rather than binary (i.e. 0-1 or yes/no) data on words for general purposes.

- The numbers were computed (to three decimal places) not by whim but by some kind of regular scientific procedure based on examining large amounts of data and using “cloud and crowd” methods.
- The crowd-sourced numbers were checked against human annotators. They were found to correlate significantly highly with an average of human opinions. The correlation measures mentioned on page 2 are not the same as the “ $R^2$ ” from the NFL part but are related.

Download the project code files from <https://www.cse.buffalo.edu/~regan/cse199/> via web or copy them from `~regan/cse199` on the CSE machines. In full they are:

```
heatlib.py
heatindex.py
NRC-AffectIntensity-Lexicon.txt    ---download or just let code access
https://www.cse.buffalo.edu/~regan/cse199/deepweb/NRC-AffectIntensity-Lexicon.txt
```

Please finally skim-read the Python code. Note especially the lines with `location = in` in `heatindex.py` and with `pageStr` and uses of `re.sub` in the `heatScore` function in `heatlib.py`, where there are alternatives to comment in or out.

## NSFW and Copyright Notice

The NRCEIL begins with the highest scoring words for *anger* and includes slurs as well as compounds of the F-word and its ilk. Well, *you* don’t have to read it—your (or our) computer will. The word *trigger* does not appear while *warning* scores 0.297 in the *fear* category.

The copyright is held by Saif M. Mohammad under the aegis of the National Research Council of Canada. We have written permission from him and his co-worker Pierre Charon to use it in CSE199. In its current v0.5 form it is even freely downloadable at its <http://saifmohammad.com/WebPages/AffectIntensity.htm> home page. However, the Terms-of-Use there include a directive not to *redistribute* it, and at some point it will be covered by the same EULA (please view) as the other NRC Lexicons.

## 2 Activity Directions

The runs are relatively simple. If using <https://trinket.io/python3> (or the alternate Trinket page):

1. *Create a new window* in your browser to go there.
2. Copy-and-paste `heatindex.py` into “main.py” there. Again, no need to change the name `main.py` and no file upload.
3. Then click the ‘+’ to add a new file, name it `heatlib.py`, and copy-and-paste the text from your own download or browser view of `heatlib.py`.
4. Click the triangle to run. You will be prompted to enter a URL. You can either type something like `www.cnn.com` right there or copy and paste the URL of any webpage you like. Again you can widen the output window to see lines whole.

On a home Python3 system or the CSE machines the command options are:

python3 heatindex.py or give a URL argument at the command line, say  
python3 heatindex.py www.cnn.com

Or from within the Python environment, enter `from heatindex import *` on the first run, and if you quit the menu (but not Python) and want to restart, enter `processUserInput(url,heatDict,mulDict)`.

1. Play around with a few webpages. You can copy and paste URLs from the browser window. Twitter pages will give you the person's last yea-many Tweets.
2. Can you find a webpage with a negative score? Negative means more 'joy'—
3. Stop—! We need a “coach's huddle” before we get carried away...

The most important thing to know about the numerical results at the very end is that they are **not designed by Dr. Mohammad** but rather by me, KWR. This project design performs a “reduction” on his work. The final number comes from adding up the ‘anger,’ ‘fear,’ and ‘sadness’ scores and *subtracting* the total ‘joy’ score. Then my code divides by the total number of words read and—totally arbitrarily for better “eye candy”—multiplies that by 1,000. In spot-tests this puts the final numbers roughly on a familiar 0-to-10 scale, concentrated more near zero, and possibly negative—when ‘joy’ outweighs the three other categories. So it has some street sense. But it's still doing “push a button and get a number”—which is what everybody wants in order to make their jobs simpler but which comes with blinkers and dangers.

Put another way, I have slapped a layer of “multipliers” onto his model. The default multipliers pass the “Occam's Razor” test of being simple: +1 each for *anger*, *disgust*, *fear*, and *sadness*, and –1 for *joy*. The code allows you to change them at will, but they are still *arbitrary*—that is, not justified by theoretical considerations and/or empirical testing. They also represent some presumptions about *what* we want to model:

- They presume we are trying to model ‘light’-versus-‘dark’ in some way.
- If we want to model *intensity* of any kind, we should use +1 for *joy* too.
- At least the +1 multipliers are leaving Dr. Mohammad's carefully-crafted numbers alone...
- But adding them up and dividing by the total number of words is still “reducing” his work. What is that ratio supposed to represent?
- If we append to the bottom of a page an equal amount of completely neutral text, we will *halve* the score. Does this make sense—shouldn't we give more weight to text at the top?
- Perhaps ‘anger’ should be regarded as generating more “heat” than ‘fear’ (which is more passive) and certainly ‘sadness.’ (Indeed, those who have seen *Inside Out* may recall its upshot about sadness.)
- Try multipliers of +10 for anger, +6 for fear, +2 for sadness and (really “winging it” now) –5 for joy. Repeat some earlier URLs. Do any scores shift notably? (Note: Just doubling the multipliers will not double the score, because the code has a further “normalizing” division by the sum of the absolute values of the multipliers.)
- Try other multipliers too. Each combo constitutes a different *model* of “webpage intensity.” Then reflect on the most important question:

From the myriad parameter combinations, how can we determine **one** that is *correct*, *best*, or at least *neutrally justifiable*? In particular, how could we *train* the model?

Answering this question first requires determining what we want to model and what the purpose is. Suppose we adopt the “light-versus-dark” purpose. The first thing we might try is to identify a corpus of pages that represent the “neutral” middle. Then we want to define our scale and train our multipliers so that those pages get a score of 0. This still does only a quarter of the work we need—in technical jargon, it leaves three “degrees of freedom” in the four parameters we are fitting.

Further progress needs asking, what are we trying to *predict*? Here we might come full circle to the “threat levels” application mentioned at the outset. Suppose we have a corpus of pages that were reliably associated with the same threat level in past history. Some pages might have more ‘anger,’ others more ‘fear’ and so on. We can train our multipliers to equalize those pages.

A general methodology emerges from the idea that how morose or bubbly a webpage written today is can predict how morose or bubbly a page written by the same person(s) tomorrow will be. Or we can apply this prediction idea from sentence to sentence or phrase to phrase. Now we are in the world of those  $N$ -grams. Predicting a probability range on the nature of the  $(N + 1)$ st item from the previous  $N$  in a continuing series involves building a so-called *Markov model* (or *Markov chain*, named for the Russian mathematician Andrey Andreyevich Markov). The Markov model  $M$  can use the principle that words continuing the score trend of the previous  $N$  are most likely. Its results thus depend on the scores and hence on the multipliers we choose. The likelihood principle in this context yields the training policy:

Find the combination of parameters that maximizes the probability that  $M$  projects for the record of what actually happened.

This sounds like “making yourself look good by predicting the past” but the power of this principle carries into the future. It still, however, takes quite a bit of work to build  $M$  and then some computer muscle to carry out the *maximum likelihood estimation* (MLE).

### 3 Discussion Points (including the NFL data too)

If you think of Data Science as ‘sexy’ then it has been my purpose to spend a lot of time metaphorically on safety and STDs—and even literally on matters of personal consent and privacy. Instead of brilliant results with clear correlations, we’ve seen:

- Inconclusive results from small data (when that is all the data that is);
- How to “cheat” with Cincinnati—or rather, how sensitive results can be to one data point;
- How quantities for which there is obvious and immediate demand are currently *inchoate*;
- How the desire for simplicity induces arbitrary choices that can be fraught with biases;
- How fixing these issues requires lots of attentive hard work.

Scoring is 3 for participation, 3 for completing the runs, and 3 for discussion. This may be re-scaled from being out of 9 to being out of 3.