CSE199 Activity: Textual Sentiment Analysis Fall 2024

This week's activity is a 'get-acquainted' experience with: using Python, data-mining the web, sentiment analysis, and numerical interpretation of results. No past programming experience in Python is needed—the program is given and runs within (the free level of) a web app called Python 3 Trinkets. Besides having fun, the goal is to raise discussion of data policy issues pointing toward the third week's mini-project. 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.

1 Introduction

Sentiment analysis means evaluating the emotions and possible intents of the authors of textual material. Here are two mainstream applications:

- Filtering communications and webpages for terrorist indications and gauges of threat levels.
- Evaluating customer (dis-)satisfaction with your company's products—and possibly diagnosing faults.

The presence of emotion distinguishes this from other textual modeling, such as the Signature *stylometry* system by which Peter Millican and co-workers identified the pseudonym Robert Galbraith as belonging to J.K. Rowling. The particular emotions are five personified in the 2015 Pixar movie *Inside Out*: *joy*, *anger*, *fear*, *disgust*, and *sadness.* [1]

We will use this tool to make simple measures of the "emotional temperature" of various webpages. One component of our measure is scientific. It is a human-scored (crowdsourced) *rating* of a lexicon of words for *intensity* in one or more of the five categories. For instance, the word "surprise" scores **0.172** for *fear* and **0.562** for *joy.* It scores zero for the other three categories. The plural "surprises" scores only for *joy*: **0.606**. The intensity is always less than 1; the most intense word for *anger* is "outraged": **0.964**. There are numerous NSFW terms in the lexicon—well, this activity is structured so that you won't see them unless you load a webpage that has them.

The lexicon is copyrighted by Dr. Saif Mohammed and the Canadian National Research Council (NRC). We are using an older version of his lexicon with permission from him and his co-worker Pierre Charron. Although the current version is freely downloadable from the project page, it carries terms of use (bottom of page) that forbid *re-distributing* it and were in effect for our version as well.

2 To Do Before Recitation

For background, please *skim-read* Dr. Mohammed's 10-page paper "Word Affect Intensities" (2018). You need not understand all the technical bits but should appreciate the following points:

- Even at professional level this kind of work is (and in 2024, still is) in early stages.

[1] This year's sequel Inside Out 2 features five new emotions: *anxiety*, *envy*, *embarrassment*, *ennui*, and *nostalgia.* Meanwhile, the creators of our tool and its training lexicon went in a different direction with categories of *anticipation*, *trust*, and *surprise.* We will stay with the basic five for simplicity.

- This was 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 linear-model examples in lecture but are related.

Then finally please skim-read the Python code files `heatindex.py` and `heatlib.py`, which can be found in my https://www.cse.buffalo.edu/˜regan/cse199/ folder. Note especially the lines with `location =` 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. They include an upgrade to version 3 of the Python URL library `urllib3` and a switch in the order of lines in the lexicon from word-score-category to word-category-score. These work in the 2024 current Python 3 Trinkets server under the directions given below. You are optionally welcome to run your code files in your own Python 3 environment. The Trinkets server has behaved well the past two years, but in case of glitches, the commented-out lines might possibly be of use.

In brief, what the code does is tally all the lexicon words that appear in a loaded webpage. The score total of words in the *joy* category is multiplied by -1. The other categories each have multipliers of $+1$, which means their positive scores are just summed together. Adding the scores gives a *negative* result if there are more—and stronger—*joy* words than others; else a positive score. It may seem strange to you that joy is being modeled by a negative number, but ask yourself, to what emotions does *intensity* more often apply?

3 Setup Directions (15 minutes including my intro video)

The runs are relatively simple and will be demo'ed in the week 2 Wednesday lecture. 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. There is 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 right-pointing triangle (standard "play" icon) to run. You will be prompted to enter a URL. With Trinket it is best to *copy-and-paste* the URL from a webpage and hit Enter.
5. Output may take some seconds to appear—even 15 seconds in some 2024 runs. The output dumps all the found words in their categories and gives the "heat score" at the end. You can slide the bar between the code and output windows to see more of the final output lines.
6. The program prompts for another URL, but on Trinket this glitches over half the time, so please instead restart everything by clicking the "play" icon again.

The Trinket app also often freezes if it is left untouched for some length of time. These glitches are not observed in a native Python environment (?), so it is hard to tell if improving `heatindex.py` and `heatlib.py` themselves would matter on Trinket.

Some webpages cause errors of three kinds:

- The webpage refuses the “GET” request in a way that causes Trinket to report an error (the output pane turns pink and shows code trackbacks). Restart and try another webpage.
- The webpage reports a decoding error—also in pink. This will happen if you try to read a PDF or Word file or anything not in UTF-8 plain-HTML format. Restart and try another webpage.
- The webpage gives no output, or nondescript output. Twitter/X pages now (2024) seem to return only the string `x.com`. The *Wall Street Journal* currently gives a boiler-plate site index dump—which gives a score of `-2.031` favoring `joy` in the activity below but doesn't count. In previous years, some pages returned the text of an error saying “Something went wrong. Don't fret—let's give it another shot” and elsewhere, “Terms of service” and “Access control disabled.” This matched the NRC lexicon on words including “wrong,” “fret,” “shot,” “service,” and “disabled,” which at first seemed thematic for a news story covering a war. If you suspect this, comment-in the line `print(Ï read ;pageStr)` (on or around around line 180 of `heatlib.py`) to get a dump of the entire text received. Or just restart and try another webpage.

Basically the catchall for troubleshooting is “just restart and try another webpage.”

4 Activity Directions

Get into your groups. The first part is to be done **individually** by everyone—for practice—but discussed briefly in groups. It also furthers the discussion about caveats to look for when loading webpages. The activity has a practice stage, then a game-playing stage, and a final “debrief.”

Practice (10 minutes)

1. First, try my webpage https://cse.buffalo.edu/~regan/ Note that just a few words match the lexicon and the net score is only slightly positive. The words “rating inflation” are a chess term, but their presence could well make readers think of the economy, so their “affect” in dimensions of anger, fear, and sadness still applies.
2. Then try the webpage https://www.kimberlymarshall.com/ of Kimberly Marshall. The score should be in the vicinity of **-10**. Look at the words in the categories. It is interesting that “music” counts for both *joy* and *sadness*, and “musical” counts further for *anger*. But they score much higher for *joy*, so they augment the overall preponderance of “joy” terms.
3. Next try the platform page https://www.donaldjtrump.com/platform of the Trump campaign. It has 20 bullet points, and if you comment-in the print line (on or near line 180 of `heatlib.py`), you'll see they are most of the text read. Note the score.
4. Now try the “Issues” page https://kamalaharris.com/issues/ of the Harris campaign. This was in the news when it was created on Sunday, September 8, because it originally had extensive *metadata* showing it had been copied from Biden's issue page. The Python code filters such header and footer data out, but it does not fix another caveat. The Harris page initially shows

19 "arrow points" in blue with short text like the Trump page has, but when you click the arrows, you get more text. Moreover, there are four red arrows giving (*their statements of*) "Trump's Project 2025" agenda for contrast. *All this text too is fetched by the Python code.* Note the score. *For group discussion:* Is it positive? Would you call it significantly or only moderately different from the Trump platform score?

The main caveat is that if we just want to compare the campaign platforms, we should remove the "Project 2025" material. Maybe we should also just use the 19 originally-visible titles of the Harris points, without the underlying text—but it is easier to read than the Trump page's saying to click on another file which is typeset (so not decodable as plain text), so we'll keep it. I have made plaintext files with-and-without the red-arrow text, and a third file containing only that text. So please now do the following:

- Load https://cse.buffalo.edu/˜regan/cse199/HarrisIssuesWithP2025.txt Verify that it gives the same score as the Harris webpage directly. (Both forms mis-decode possessive apostrophes, but that doesn't matter.)
- Load https://cse.buffalo.edu/˜regan/cse199/HarrisIssuesMinusP2025.txt This is meant to be the pure Harris platform. *Briefly discuss*: would you call the difference in score to Trump's (briefer-stated) platform significant now?
- Finally load https://cse.buffalo.edu/ regan/cse199/HarrisIssuesOnlyP2025.txt and note the score. Mind you, this is *the Harris statement of* the project 2025 points.

Game-Playing Stage (10–15 minutes, including time to compare groups' results)

Your objective is to find *webpages with negative scores*—besides the one above. The group with the most-negative score wins brownie points. A second point can go to the group that finds the most webpages with negative scores—or the most diversity in such webpages. Think of kinds of webpages that would have incentive to generate an uplifted impression on their readers.

The following materials are to be logged and submitted by each group:

1. URLs and brief descriptions of webpages you try.
2. For at least two pages with negative scores, a discussion of why the page would be interested in promoting *joy.*
3. A brief reaction to the following "skeptical debrief"—which can be put into your individual final feedback rather than a group submission (and so won't be graded as part of the activity itself).

The grading rubric is: **1 pt.** for participation, **1 pt.** for showing webpages you tried, and **1 pt.** for finding negative scores and discussing why. Especially the final one-point can be divided as **0.25** for each page with a negative score and **0.25** for the accompanying discussion.

5 Skeptical Debrief (as time allows, or offline: the third-week assignments will pick up on some of this)

I hope the webpage scoring is fun, maybe even addictive... But before we get carried away, we need a "coach's huddle" to address the general question of **whether this is responsible data science**.

The most important thing to know about the numerical results given 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,' 'disgust', 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 other four 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 Dr. Mohammad's model. The default multipliers pass the "Occam's Razor" test of being simple: +1 for each *anger*, *disgust*, *fear*, and *sadness* word, 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.

If the debrief gets to this point in the recitation session, that's fine. If there is further time, then we can go into the following—or if not, invite these further considerations offline. They discuss *what* we want to model, and whether the "multipliers" bolted onto Dr. Mohammed's lexicon and the concept of a single "heat score" make some undue presumptions:

- 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 that the "negative" emotions have positive roles.)
- Try multipliers of +10 for anger, +6 for fear, +2 for sadness and (really "winging it" now) −5 for joy (disgust may be left at +1). 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 responsibly *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).

In any event, all of this work—and more—would be needed to bring this "heat scoring" idea anywhere near the level of depth and responsibility and vettability that has gone into my chess predictive model. The fact that the results are plausible even in this simplistic formulation does not confer a scientific imprimatur. It is if anything more prone to appeal to your incoming biases than to reflect the application of professional standards that we seek in the coming years to confer on you at UB.