Practicum 5 - Sentiment Analysis

DS2001 - Computer Science Practicum
Fall 2021
October 6/7, 2021
Practicum Deadline: October 8th, 2021 at 12:00pm (noon!) Boston time

To complete by 9am 10/13/2021 in addition to the work in class this week:
- Read the final project description (on the course website)
- Complete the Practicum 6 - Pre-Practicum Work item on Canvas
    - (including posting to the linked Final Project Discussion thread)
- This will count towards your Practicum 6 grade.

Based on feedback that we've received from students, we're going to experiment with telling you *how far we expect* you to get at the beginning of practicum. This is contingent on both:
- you agreeing to work diligently for the entirety of your practicum, even if you get past the designated "full credit" task
- Felix reserving the right to adjust which task is the "full credit" task, *especially* in the direction of "you don't have to do as many tasks as we thought you would"

We will note any adjustments in announcements on Canvas. Do **not** count on this pdf being updated! (And please, do continue to give us feedback!)

What we're practicing today:
- while loops (repetition)
- lists! -> indexing, slicing, comprehension
- (for loops)

Handouts:
- While loops (from DS 2000)
- Lists (from DS 2000)

What to do if you miss practicum this week:
- Fill out the "I'm missing class" form ASAP
- Follow up in office hours with Felix (find office hours links on the course website). **Note that this is required.**
- Come to practicum next week if you are well again. **Do not come to class if you are not feeling well.**

How far we expect you to get this week:
- We expect a good attempt at least through **Task 3**, and hope that you at minimum read through Task 4 and think about how you would approach it, even if you don't make an attempt at this task during practicum.

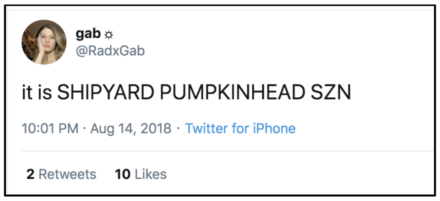
Everyone should create one file for the whole practicum called **p5.py**, and you'll add to it as you go. You'll define **one** `main()` function and add the content from subsequent tasks to it!

Alert! This week we will start making small deductions for submissions that 1) have errors when they run through the autograder and 2) lack answers to group questions (include these as comments in your file!).

Task 0: Write your file comment and define your `main()`
First, write down your name in a comment at the top of your file and a small description of what the file is, like we did last time. This is a good habit to get into for all your programs!

Make sure to write down the names of all your group members as well!

Task 1: Set up (this task has a group question, but no coding involved!)
As humans, we are able to read, say, a tweet, and determine whether it's conveying a generally positive or generally negative thing. For example...


gab ☼
@RadxGab

it is SHIPYARD PUMPKINHEAD SZN

10:01 PM · Aug 14, 2018 · Twitter for iPhone

2 Retweets 10 Likes

...you can tell that this person likes Shipyard Pumpkinhead beer (and they are correct, according to Prof. Laney. Felix has no opinions on this particular matter). You can tell that. I can tell that. But can a computer?

Coming up with a way for a computer to understand the feelings conveyed in text is called *sentiment analysis.* Computers aren't as good as humans at this task; it broadly falls under artificial intelligence (AI) as well as data science.

During spring semester, I was reading through some comments online about the COVID-19 vaccination roll-out here in Massachusetts. I wondered if people were happy/unhappy and whether or not that reaction has changed over time.

For today's practicum, we'll tackle this same problem with Python code. We'll start with a simple approach, but as you can imagine these kinds of algorithms can be tweaked and refined till the cows come home. (In fact, doing an advanced version of this assignment is what my students in CS 4120/6120 - Natural Language Processing are working on next week!)

Group question: come up with two different strategies that you think a computer program might use to measure the sentiment of a given piece of text.

Task 2: Read in the comments

We've provided you with files containing comments to analyse, positive words and negative words:

- `comments.txt`
- `positive_words.txt`
- `negative_words.txt`

Your overall goal is to assign the comments in the comment file a **sentiment score**.

A score can be assigned to one comment (what does one person think?), or to a whole big collection of them (what does everyone think?).

First off, you'll need to read all three files into three different lists. We did this last week, so we went ahead and put some example code here for you that you can use to get started. We've put two options to match the two most common patterns for reading a file into a list.

```
      # you'll need to define the filename variable for either option!

      # option 1
      ls = []
      file = open(filename, "r")
      while True:
            line = file.readline()
            # stop if we've reached the end of the file
            if line == "":
                  break
            # leave lines as strings, but use .strip() to remove the \n
            ls.append(line.strip())
      file.close()
      print("here's your list:", ls)

      # option 2
      ls = []
      file = open(filename, "r")
      line = file.readline() # read the first line
      while line != "":
            # leave lines as strings, but use .strip() to remove the \n
            ls.append(line.strip())
            line = file.readline() # read the next line
      file.close()
      print("here's your list:", ls)
```

For the comments, once you have them read in so that every element in the list is one comment, you're going to transform the contents of the list so that each comment becomes a **list of lower-cased words**. Use the string `.lower()` function to do this.

Note! This function is a string function, not a list function!

When you've finished this task, you should have:

- a list of positive words
- a list of negative words
- a list of comments

Print out the length of each list and <u>the first 3 elements</u> from each list:

```
Number of positives: 2006
['a+', 'abound', 'abounds']
Number of negatives: 4783
['2-faced', '2-faces', 'abnormal']
Number of comments: 14
['massachusetts gov. baker issues a warning after hearing some people are trying to take
advantage of new companion eligibility rules to get 75 and older residents to bring them
to vaccination sites. http://nbcct.co/otmumj0 #nbcct', 'the idea that a state like
massachusetts can’t get vaccines out faster is inexcusable. the per capita percentage of
people who work in the healthcare industry means that we should be running vaccination
centers from morning till night until everyone who wants one get one.', 'from the vantage
point of a 71-year-old, turning 30 doesn’t mean that life’s options are necessarily
narrowing. at the moment i’m marveling at the rare feeling of being too young for
something even at this age: vaccination in massachusetts.']
```

Group question: what would happen if you did **not** convert the comments to lower case? Do you think that there would be later impacts on your program?

Task 3: Sentiment score of one comment

For this part, we’ll assign a **sentiment score** to a single comment -- what does this one person think?

Write the code to produce a sentiment score for the first comment in your list of comments, with each word its own element, like this:

```
['massachusetts', 'gov.', 'baker', 'issues', 'a', 'warning', 'after', 'hearing',
'some', 'people', 'are', 'trying', 'to', 'take', 'advantage', 'of', 'new',
'companion', 'eligibility', 'rules', 'to', 'get', '75', 'and', 'older',
'residents', 'to', 'bring', 'them', 'to', 'vaccination', 'sites.',
'http://nbcct.co/otmumj0', '#nbcct']
```

You should iterate over the comment, then, if a word in the comment appears in your list of positive words, add 1 to the comment’s score. If a word in the comment appears in your list of negative words, subtract 1 from the comment's score.

Print out the comment and it's overall score when you've finished!

Example:

```
massachusetts gov. baker issues a warning after hearing some people are trying to take
advantage of new companion eligibility rules to get 75 and older residents to bring them
to vaccination sites. http://nbcct.co/otmumj0 #nbcct
score: -1
```

Hint: you might find it helpful to print something out each time the score changes so you can see what word was counted in what direction!

Group question: come up with an example comment that either *looks positive to your program but is actually negative* or vice versa.

Task 4: Sentiment Score of the Group

We don’t just care about the sentiment of one comment (what does one person think?), but a bunch of them collectively -- that tells us what everyone thinks, as a whole. Modify your code from Task 3 so that it computes **one** sentiment score **per comment** for a **whole list** of comments. (this is your list of lists of words!)

Print out each comment and each comment's score.

Task 5: Graphing Sentiment Trends

Next, modify your code so that each score is itself stored in a list of scores.

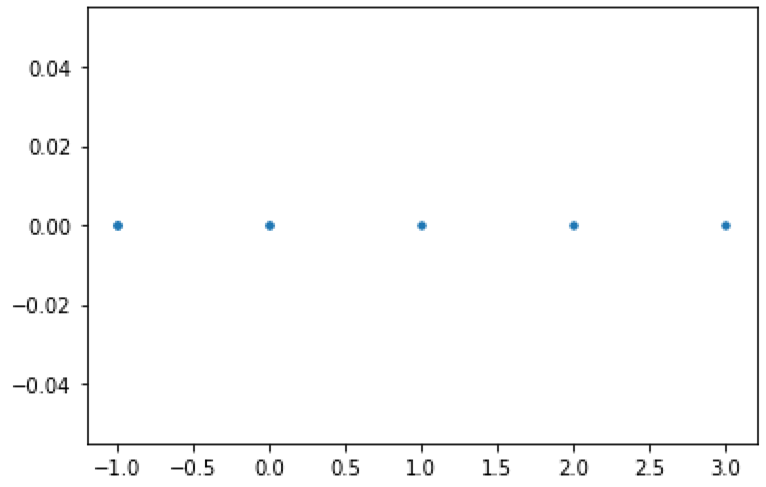
Using **list comprehension**, create a list that has the same number of elements as your list of scores, but where every value is the number 0. You may find it useful to create your list first with a "regular" while loop or for loop, then, once your code is working, change it to be created using list comprehension!

Here is an example of list comprehension, which combines list creation with loops:

```
my_new_list = [i for i in range(5)]
print(my_new_list) # [0, 1, 2, 3, 4]
```

Now, graph your scores, using the scores as the x-values and the 0s as the y-values (or vice-versa!). This time, we'll be using the `plt.plot()` function with parameters of two lists, a list of x values and a list of y values, rather than two integers!

Your resulting graph should look something like this:



One of the issues that we might encounter with this graph is that all comments with a score of, say, 1, are **graphed on top of each other.** A more appropriate kind of graph might be a histogram, whose job is to plot frequencies. For the `plt.hist()` function, we can pass in **one** list of values (like our scores) and matplotlib will count up how many scores fall into each "bin" where a bin has a certain range, say, 1 - 3, then it will display a graph with the bin values on the x-axis and the count (or frequency) on the y-axis. Neat!

Task 6: Curate your own test set & test out your results

Finally, you'll make your own `movie_reviews.txt` file. Put a bunch of reviews of your favorite movie (or anything else, really) in this file. Run your code so that instead of reading from `comments.txt`, it reads from your `movie_reviews.txt` file. What distribution of scores do you get? Did you have to modify anything about your code other than the name of the file? (you shouldn't need to)

Done? Make the following improvements

- This initial version of sentiment analysis is focused only on one word at a time. This can be a problem, because if you have the word *good* in your positive list, and the comment is *this is not good*, then you'll miscategorize the comment.

    - Update your scoring code to try to do a better job with negation!
- add axis labels, titles, and legends to your graphs
- use for loops instead of while loops to read in your example files
- Do your Practicum homework for this week! :)

Remember to submit whatever you have working at the end of the allotted time!

Sources for our positive & negative word lists:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Opinion Lexicon: Negative
;
; This file contains a list of NEGATIVE opinion words (or sentiment words).
;
; This file and the papers can all be downloaded from
;       http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
;
; If you use this list, please cite one of the following two papers:
;
;   Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews."
;       Proceedings of the ACM SIGKDD International Conference on Knowledge
;       Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle,
;       Washington, USA,
;   Bing Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing
;       and Comparing Opinions on the Web." Proceedings of the 14th
;       International World Wide Web conference (WWW-2005), May 10-14,
;       2005, Chiba, Japan.
;
; Notes:
;       1. The appearance of an opinion word in a sentence does not necessarily
;       mean that the sentence expresses a positive or negative opinion.
;       See the paper below:
;
;       Bing Liu. "Sentiment Analysis and Subjectivity." An chapter in
;       Handbook of Natural Language Processing, Second Edition,
;       (editors: N. Indurkhya and F. J. Damerau), 2010.
;
;       2. You will notice many misspelled words in the list. They are not
;       mistakes. They are included as these misspelled words appear
;       frequently in social media content.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```