Mike Vanger
Text Analytics
Assignment 2

I began by reading in the file. This gave me one long string.

I noticed that each bio was separated by something like "04 ---" or "----", so I replaced those separators with a whitespace character.

I did the same for carriage return and newline characters.

I also made the string lowercase.

Next, I needed to split the string into individual sentences. To do this I made the assumption that a sentence ends with ". ", "! ", or "? ".

This gave me an array where each entry was, given the assumptions, an individual sentence. There are possible exceptions/mistakes, for instance if someone forgot to put a space after a period. The space is necessary, however, because there could be a decimal number within a sentence. The array has 662 entries.

The next step was to think about which time points should be captured. Perhaps the most obvious one was the year. I made the assumption that the year would be of the form [1-2][0-9]{3}. This gives every year between 1000 and 2999. Given what I know about the text, it made more sense to limit it to 1900-2015. The regex I used captures those years, as long as the characters immediately before and after the 4 digit year are not a digit. Unfortunately, something like "cars were parked in 2000 spaces" would also be captured as a year. I do not have a good way to avoid this pitfall. Regex = /[^0-9](19\d{2}|2[0-1]{2}[0-5])[^0-9]/

This returned 9 sentences. Since this was a small set I took a look at the results. The only questionable results were "1990's" and "class of 2013". The first one, which represents a decade, I felt was reasonable. If you wanted to remove it, you would just need to check that the character after the fourth digit was not an apostrophe. The second result should probably not count as a timepoint. I cannot think of a good way to filter those. Checking that the year is not preceded by "class of " would work in this case, but there are many similar cases (for instance "msia 2013").

Next I decided to look for months. I limited my search to the 12 full correctly spelled months and their 3 letter abbreviations (ie jan, feb, oct, et cetera). I checked that the preceding and following characters were not alphabetic or that the preceding character did not exist (ie the month was the beginning of the string). Regex = /([^a-z]|^)((january|february|march|april|may|june|july|august|september|october|november|december)|(jan|feb|mar|apr|may|jun|jul|aug|sep|oct|nov|dec))[^a-z]/

This gave 7 results. The only problematic ones included the word "may" which is a common word. I do not have a good solution for this. Something like "in may I will..." would work, but checking that "may" is preceded by something like "in", "on", or "during" does not necessarily work. For instance, you could have sentences such as "may: i graduated" and "may he rest in peace."

Next I looked for days of the week. Similar to months, I looked at their complete spellings as well as what I deemed the most common abbreviations. It would be possible to extend this to more abbreviations, but I stuck here to using only one rather than going on indefinitely. /([^a-z]|^)((monday|tuesday|wednesday|thursday|friday|saturday|sunday)|(mon|tues|wed|thurs|fri|sat|sun))[^a-z]/

This only returned one result. Unfortunately, it was a sentence that used "sat" as the past tense of "to sit." This is the same problem as the month of May. One possible solution would be to avoid downcasing the string. Either solution presents potential problems (for instance, we cannot guarantee that the writer correctly capitalizes their months or days).

Next I wanted to look at dates of the form mm/dd/yy. Unfortunately this has many permutations. For this assignment, I kept it to between 1-12 for months, 1-31 for days, and 00-99 or 1900-2015 for years. I probably missed some with this regex. I also assumed an American ordering of mm/dd/yy rather than dd/mm/yy. Regex = /([0-9]|1[0-2])\/([0-2]{0,1}[0-9]{0,1}|30|31)\/((19\d{2}|2[0-1]{2}[0-5])|[0-9]{2}(\D|$))/

There were no results for this, which I think is okay. Other possibilities I did not check for include mm-dd-yy and variations.

At this point I thought of some relative timepoints, and decided to check for "today", "yesterday", and "tomorrow". The regex is straightforward.

This gave three results, but all of which used "today" in a general sense (ie "today's data scientist"). I decided against including this regex.

Next I tried combinations of "this" and "year|month|week"

That only gave one result, which was not a time point, so I rejected this as well.

Then I tried combinations of "last" and "year|month|week"

This gave seven results, most of which seemed reasonable.

I did not attempt any other relative times.

I next attempted to look for times. The most basic was something like 12:00. I used military time. I don't know if 24:00 is a time or if it is 00:00. I used 00:00 Regex = /(1[0-9]|2[0-3]|(^|\D)[0-9]):([0-5][0-9]){1}/

There were no matches.

The only other time I could think of was something like "4 o'clock" or "4 am" Regex = /(^|\D)([1-9]|10|11|12)\so'{0,1}clock/

There were no matches.

For "4 am", the regex I used was /(^|\D)([1-9]|10|11|12)\s(am|pm)($|\W)/

Again, there were no matches.

These were all I have thought of so far. I did not consider something like October 9, 2014, since it is included already. I combined them all and output the results. There are 22 lines. The only questionable lines have already been addressed above (for instance, "class of 2013" and "he may mention"). I used Ruby for this project.

For a visualization, I found every matching string and counted the frequency of its occurence. I included the results for "today" as well. Then I displayed each match (not the whole sentence) for a period of time proportional to its frequency. It can be viewed at http://mike-text-viz.herokuapp.com/