Jessica Sutantio

Software Design Spring 2015

Text Mining Mini Project

The Loudest Rants

Using the mailing list archives of Therapy, I aimed to find the most frequently discussed topics in the mailing list. Originally, I hoped to find the most frequent words and audibly play the most popular words of varying volumes depending on how frequently the words were mentioned in the emails. I hoped to learn how to rip information off the internet and convey the analyzed information in an unconventional matter.

Implementation

Despite my desire to learn how to scrape or crawl through the internet through python, I chose download the individual monthly archived emails off the mailing list website. This was so that I could appropriately dedicate more time to analyzing the data, rather than dealing with the tedious process of crawling through password enabled websites. I was able to filter the email content while combining the multiple files into one file. The most difficult part of the email filtration process was that there were several nested conversations and content that I did not want to analyze. I wanted solely the body of the email, once. I found no other solution, but to hardcode the exclusion of email IDs, headers of duplicated messages, and subject lines. As I was checking for the excluded content, the files were compiled line by line.

Once I achieved a clean file with only the email body content, I simply created a dictionary of all the words with their word counts as their respective values. Because natural text contains several common words, I had to cut off approximate 200 of the most popularly used words. This was done after ordering the dictionary based on the descending order of the dictionary values. The words were then put in a list, and I simply created a new list starting from the 230th element of the previous list. Choosing a place to cut off the common words was quite arbitrary after the first 150 words; therefore I chose to start on a more interesting group of words. Another way of excluding the most common words such as “the”, “a”, “to”, etc., was getting a simple text file of the most common words from the internet instead of eliminating approximately the first 200 frequent words in my file. This would ensure that some of the common words would not appear later in the ordered list, nor will the most frequent, uncommon word be eliminated. I chose not to go this route due to expedience and simplicity, as well as most files on the internet required restructuring or payment.

In order to get an audio sampling of the “loudest rants”, I used a simple text to speech api from online by using the pattern.web module. The output of the application is an .mp3 file of the words I cycled through.

Results

Remember

Years

Can’t

Money

Here’s a sample of the first four words generated by running the code. I’ve embedded the most frequent word: “remember”. The others are located within the Rants folder within the GitHub repository. In total, I was able to count about 40,000 discrete words. The first 400 are printed here:

```
['to', 'the', 'i', 'a', 'and', 'that', 'of', 'you', 'is', 'it', 'in', 'for', 'this', 'not', 'on', 'my', 'people', 'have', 'be', '', 'at', 'are', 'but', 'or', 'about', 'was', 'with', 'if',
'like', 'we', 'do', "it's", 'they', 'me', 'who', "don't", 'as', 'so', 'what', 'just', "i'm", 'think', 'would', 'because', 'can', 'an', 'all', 'olin', 'your', 'want', 'there',
'when', 'know', 'from', 'out', 'up', 'their', 'more', 'get', 'by', 'one', 'how', 'them', 'some', 'something', 'had', 'no', 'should', 'fucking', 'than', 'really',
'even', 'why', 'has', 'our', 'good', 'someone', 'feel', 'also', 'day', 'time', 'here', 'students', 'things', 'make', 'which', 'other', 'being', 'will', "you're",
'am', 'person', 'better', 'were', 'been', 'anonymous', 'email', 'say', 'only', 'therapy', 'much', 'now', 'going', 'names', 'fuck', "didn't", 'take', 'off',
'school', 'way', 'list', 'then', 'naming', 'where', 'us', 'lot', 'anything', 'every', 'any', "that's", 'never', 'said', 'down', 'did', 'year', 'thing', 'probably',
'need', 'shit', 'change', 'anyone', 'these', 'go', 'carpe', '1', 'could', 'tell', 'still', 'might', 'most', 'please', "i'd", 'care', 'different', 'does', 'too', 'life',
'right', 'talk', 'design', 'part', 'wrote', 'fact', "i've", 'over', 'bad', "there's", 'sure', 'around', 'everyone', 'into', 'rants', 'come', 'those', 'see', 'raves',
"we're", 'he', 'got', 'happen', 'hall', 'very', "they're", 'believe', 'emails', 'thought', 'actually', "doesn't", 'campus', 'mailing', 'dining', 'enough',
'while', 'little', 'though', 'best', 'name', 'idea', 'least', 'pretty', 'use', 'candidates', 'well', 'made', 'babson', 'work', 'before', 'hear', 'maybe', 'group',
'open', 'myself', 'help', 'through', 'always', 'hard', 'last', 'challenge', 'osl', 'love', 'information', 'many', 'meal', 'cause', 'own', 'wrong', 'place', 'first',
'college', 'especially', 'general', 'put', 'less', 'else', 'back', 'makes', 'stop', 'getting', 'class', 'after', 'remember', 'years', "can't", 'money', 'again',
'two', "isn't", 'long', 'let', 'eat', 'mind', 'without', 'mean', 'trying', 'public', 'food', 'nothing', 'community', 'having', 'sent', 'instead', 'student',
'problem', 'may', 'seriously', 'same', 'seems', 'real', 'op', 'event', 'story', 'fire', 'look', 'snow', 'world', 'issues', 'send', 'few', 'high', 'able', 'already',
'opinions', "let's", 'today', 'making', 'technology', 'read', 'alarm', 'health', 'particularly', 'start', 'both', 'whatever', 'kids', 'answer', 'wanted', 'days',
'talking', 'away', 'yes', 'specific', 'night', 'everything', 'cool', 'live', 'between', 'situation', 'totally', 'full', 'issue', 'done', 'b', "wouldn't", 'point', 'try',
'uncomfortable', 'honestly', 'doing', 'she', 'friends', 'give', 'end', 'message', 'nobody', 'came', 'happy', 'heard', "shouldn't", 'yourself', '2', '3',
'case', 'mom', 'hey', 'etc', 'rather', 'its', 'reason', 'alone', 'swe', 'pm', 'appropriate', 'meals', 'next', 'discussion', 'him', 'dad', 'keep', 'means',
'home', 'whether', 'hate', 'sorry', 'saying', 'however', 'yeah', '10', 'senior', 'outside', 'either', 'worse', 'family', 'okay', 'since', 'understand', 'rules',
'parents', 'shitty', 'question', 'once', 'telling', 'soup', 'others', 'conversation', 'clearly', 'unless', "won't", 'job', 'almost', 'political', 'charities',
'space', 'tired', 'needed', 'matter', 'o', 'weekend', 'died', 'wait', 'angry', 'awesome', 'sounds', 'later', 'wants', 'doctor', 'week', 'comes', 'agree',
"i'll", 'amount', 'hell', 'miss', 'supposed', 'sometimes', 'find', 'used', 'big', 'google', 'often', 'per']
```

Reflection

As expected, the word count dictionary was the most familiar and fastest part to implement. In retrospect, this project (to the degree that it was completed) was very simple. However, it was extremely difficult for me to familiarize myself with the tools that were used, as well as understanding the email structure and format. What was also interesting was that none of the tools or modules suggested in the assignment was used, however inspired the use of emails and some interactive online component.

Because the code did not require any user input, most tests were performed as the code underwent several increments of iterations. I first started off with pseudo code, so that I can structure the function and determine the tools required. Then after implementation, I would gradually test after each part of the loop was implemented.

Should I continue to revise this project, I would do the following:

1. Vary the volume of the .mp3 files by lowering the volume by pydub
   Due to time constraints, I wasn't able to get pydub successfully installed. This user-generated module enables the modification of mp3 files. By taking the length of the final ordered list of frequent words, I am able to scale the volume to correspond with the word's usage. Based on what I know so far about pydub, I would have to lower the volume by units of decibels, rather than percentages or ratios.
2. Combine all the audio files into one with pydub
   To avoid opening more than one audio file, I would then use pydub to combine all the scaled volume audio files.
3. Crawl or scrape through lists.olin.edu to get to the Therapy archives (assuming credentials are given)
   I was very eager to write code that was flexible and adaptable. By scraping through the website for the files, I would be able to access any emails that were written to the list to this date. The current code is limited to the database of files that I manually downloaded from the website, which limited the content to be within 2 years of emails, rather than 8 years and more.
4. Check for all the cases for email bounces, repeats, and replies
   In order to make the code more robust, I would like to scour the text for all the cases in which the text is not new or useful. This may involve hardcoding more exclusion cases, or crawling through the website and downloading emails based on bodies of text or by thread.
5. Limit to subject lines.
   Because subject lines are more descriptive and contain fewer words, the text is a far better candidate for achieving key words for relevant and popular topics. The path was not originally chose, simply because the mailing list is not old enough to have a large database or subjects.