Patrick Kispert

Natural Language Processing

**What’s the assignment?**

Run text analysis on a twitter feed or another data source. In my case, I chose to run an analysis on 1000 of Sarah Palin’s e-mails, trying to run 2000 emails was too much for my computer to handle.

The first step is to clean the data.

We were then asked to create word clouds, of the unigram, bigram, and trigram variety.

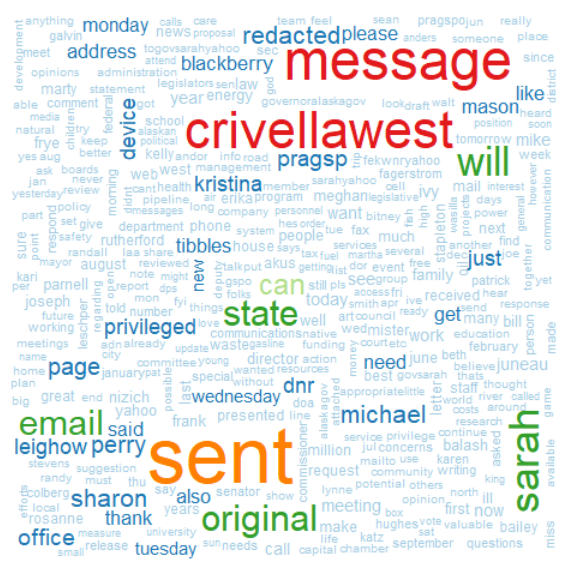
Then we were to create a TF-IDF word cloud.

We will have to run comparison/contrast word clouds based on whether the emails were positive or negative.

Finally, we needed to run an emotional analysis on the e-mails. Then, we needed to

**The Pre-Processing Stage:**

Naturally, the first step that needs to be taken is to load emails. Then, we needed to convert the email list into a vcorpus. Vcorpus stands for Volatile Corpus, which is a list of text files that is stored in the RAM. Next, we needed to clean the text files, which includes deleting numbers, punctuation, and stop words, which are common words that wouldn’t add to the analysis that was required of us. I chose the stop words by running the cloud analysis’s and seeing what common words show up that we would expect in most emails. Below is an example of how I chose stop words.



In this case, the words sent, message, email, original, etc. were deleted, because these are words that we would expect of find in an email. After cleaning the data, we needed to convert the texts to a term document matrix, which is a matrix where each row is an email, each word is a column and the elements are the number of words in each row. By using the tidy function, we gain a data frame that has the summary of the model’s statistical findings.

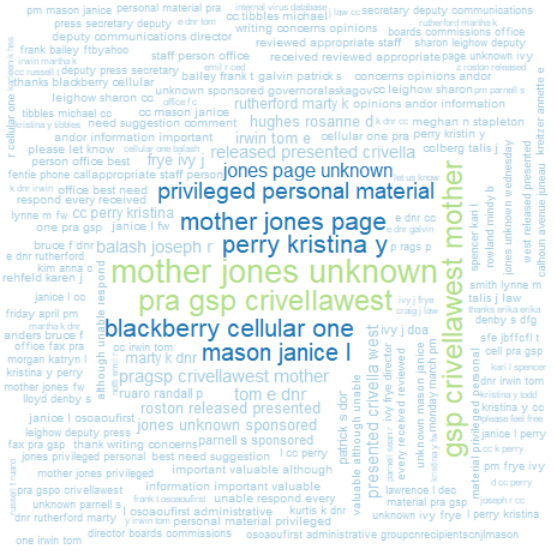
**Term Frequency Word Clouds:**

The first step for the Unigram word cloud is to convert the data frame to a matrix, add all the rows together and sort them in descending order. This gives us the number of times each term (in this case a word) is used in every email. Then convert the matrix back to a data frame and use the wordcloud function. With word clouds, the larger the word is the more times it (they) are mentioned.



With Sarah Palin’s emails, Crivellawest is the largest word. Crivella West is an analytics company.

With bigrams and trigrams, they follow how unigrams are made, except for some code that is created/ran before the unigram’s code. The first step is to create a tokenizer, which (in this case) pairs off words in twos (for bigrams) and threes (for trigrams) as terms. Then we run the corpus through the term document matrix with a new parameter, the tokenizer. After that, the code is the same as the unigram code, but with the updated variable names. Below will be the bigram on the left, and the trigram on the right.



As you may have noticed, as the number of words being used to create a term increases, the fewer number of times they show up.

**TF-IDF Word Clouds:**

TF-IDF stands for Term Frequency – Inverse Document Frequency. TF-IDF word clouds are used to determine how statistically important each word is in the email. When coding the TF-IDF word cloud, it is very similar TF word clouds. The only real difference is when you put the text through the term document matrix, you add a control parameter to the mix.

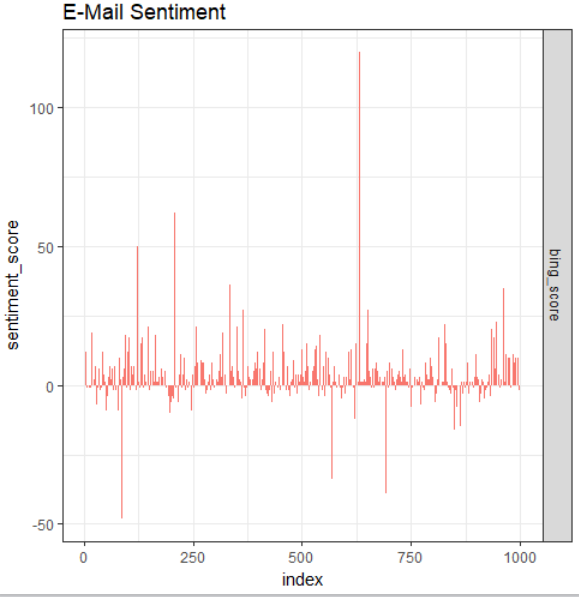


And once again, the size of the word means that the word was the most important to the largest number of emails. One interesting thing to note is that there are quite a few names in the word cloud.

**Sentiment Analysis:**

Within each text file, sentiment analysis takes each term (in this case, once again, is the individual word), finds the words sentiment score, and adds them all up.

In programming the sentiment analysis, we needed to decide on a lexicon (a specific set of vocabulary), which will give certain weights to each word, and this case I chose to use the bing lexicon. We then needed to join the bing lexicon with the files\_tidy, which is a list off all the words used in the emails. We need to give the data frame a new column that is a 1 if the word’s a positive, and -1 if the word is negative. We then multiplied the sentiment value by how many times the word’s in a text, then aggregate (subset the data, get a summary on each subset, and return the results) the data. Now we have a list that has two columns, an index (a number for each text), and the given index’s sentiment score. If there are any na’s in the list than we replace it with a 0. Once this is all said and done, we are able to create a graph using ggplot.



As one can see, the majority of the text files fall within a sentiment score of -30 and 30. There are a few emails that have an extreme value. One such e-mail (15953) has a sentiment score of 148, and this is due to the length of the e-mail, which has over 4000 words in it. We have taken a look at several documents that have a large sentiment score, and it would appear that they contain a longer length then the average documents.

**Commonality/Contrast Word Clouds:**

We have been asked to commonality/contrast the positive and negative emails. The method that we found to do this was to first merge a couple of the data frames so that we could get the email text along with its associated sentiment score. Then we split the text into those that have a positive sentiment score and those that have a negative sentiment score. After that, we unlisted all of the text so that the text would show up as one large text file, rather than many smaller text files. After collapsing the text, we merge the two lists back together, which is then run through vcorpus, clean\_corpus, and termdocumentmatrix functions. After converted to a matrix, we can run the positive and negative emails through commonality.cloud and comparison.cloud. Below, the commonality cloud is on the left, and the comparison cloud is on the right.



The size of the words in the commonality cloud are relatively smaller, since the positive and negative emails probably share a large portion of the vocabulary. It’s not too unexpected that the positive side of the comparison cloud has large words, since there are more positive emails in the sample that we used.

**Emotion Analysis:**

An emotion analysis breaks down the expression of the text, in this case e-mails, into one of eight emotion categories: anger, anticipation, disgust, fear, love, sadness, surprise, and trust, and graphs their total count. Just like the sentiment analysis, a lexicon needs to be chosen, this time I went with the NRC lexicon, since using the Bing lexicon gave me a graph that only had positive and negative in it, not the eight emotions that are in the following map.



According to the graph, there is a large number of trust and anticipation emotions in the text, with there being relatively few expressions in the other 6 emotions.