INTA 8803 - Big Data Analysis

Elizaveta Gonchar & Ian Helfrich

April 15, 2019

To begin, this project was painful and did not necessarily follow the structure as indicated by the questions we are asked to answer. Our initial premise was to analyze changes in email sentiment to find evidence of changing opinions and subsequent wrongdoing within Enron. This approach was based on the idea that, over time, individuals within the company, particularly those partaking in wrongdoing, will shift the polarity of their emails towards expressing negative or neutral sentiment; moreover, we believed that individuals who engaged in some form of devious behavior were more likely to express deteriorating sentiment towards the Risk Assessment and Control (RAC) department and associated persons within Enron in the months leading up to the company's eventual bankruptcy. The RAC played a role in Enron's wrongdoing by allowing employees to ultimately push inflated valuations and excessively risky deals through the department despite their own protests, without raising alarms (McLean and Elkind, 2013); however, quite often being the gate-keeper meant members of the RAC were seen as "slowing down progress" by other within the company. This is what inspired us to track changes in employee sentiment over time whether employee sentiments transitioned either to be more neutral or negative over time.

During the course of this project, as was ultimately the case at Enron, things did not turn out exactly as we had planned. Through trial and error, we were able to track changes in email sentiment for 150 core Enron executives over time. We did not, however, manage to filter our analysis to only consider sentiments regarding the RAC department and associated staff. This is due both to the steepness of the learning curve associated with running sentiment analysis in Python and to limitations in the structure of the data.

For the purposes of this project, we define a "neutral sentiment" regarding the RAC as one that views the RAC as irrelevant to their personal success, whereas an employee regarding the RAC as a direct detriment to their success in the company is classified as having a negative sentiment. We began our analysis with the priors that neutral sentiments are likely indicative of employees finding ways to "game" the system and feeling that the RAC is not effective and that a negative sentiment is indicative of an

individual feeling that the RAC is blocking career enhancing tactics which are likely to be egregiously unethical or illegal.

To start our journey, we began with an analysis of email sentiment using the core natural language processing unit (CoreNLP) from Stanford University, using the python module[1] *pycorenlp* to compute the sentiment of words and phrases. The goal was to run an aggregate analysis before progressively homing in on the topic in on sentiments regarding the RAC, then expanding the analysis to measure the same trends over time. This unfortunately did not happen as we came across many roadblocks along the way. First, we struggled in deciding which data to use. We initially intended to use the "maildir" files[2]. Researching the situation, it became apparent to use that the act of importing multi-directory files is not a trivial task in python. Second, we struggled to get the CoreNLP server running locally before attempting to analyze multiple strings of data, which also proved more difficult than expected. Third, just we figured out how to run CoreNLP on localhosts, Elizaveta's entire Python setup "broke," due to alterations in the way the core file directories had been reconfigured in the process of trying to figure out the CoreNLP system, which set the project back more than a day. After having fixed the problem with Elizaveta's system, we hit our fourth snag: we could analyze the sentiment in a single email but would still need to decide how to aggregate the sentiments of all of the lines within the email, as well as find a way to loop the analysis over all of the data to incorporate the weighting system we designated. This fourth set of issues, coupled with time constraints and a healthy does of naiveté, led us to abandon CoreNLP[3] in favor of other methods of conducting sentiment analysis.

In the end, we found that the analysis provided by the Textblob module seemed to align more with our interests. The only component it was lacking relative to CoreNLP was whether the sentiments we positive, negative, or neutral. Upon further research, we found that the TextBlob module itself includes an option to attach sentiment indicators in string form to the float polarity score. However, we are not exactly sure how to make the module run in this way. Our attempts to do so seem to also tokenize the email corpus and provide sentiment values for each word in each email individually, rather than on parsed sentences or blocks. Ultimately, we chose to continue with TextBlob over CoreNLP for two principal reasons. First, it computes the sentiments of each email body as a whole, whereas CoreNLP computes sentiments by identified strings within each email. This lack of aggregation in CoreNLP

[1] https://stanfordnlp.github.io/CoreNLP/index.html

[2] http://www.cs.cmu.edu/~enron/

[3] For now, at least. It may be worth revisiting to conquer the beast in the near future.

presents opportunities for bias to be introduced because we would need to decide how to average the sentiment values within an email by determining possible weightings. This bias stems from the fact that the sentiments are computed by selected strings and given that the string selection varies by email, we may inappropriately average sentiments of emails. For example, in Figure 1, we see that there are more neutral sequences than negative. In this case, 38 of the words from the email account for three strings identified to be neutral, whereas 52 words account for two strings identified to be negative. If we were to simply compute the average of the computed sentiments, we would find that the sentiment is -0.4; however, if we were to weight the scores by proportion of email (as in $(-1 \times \frac{52}{90}) + (0 \times \frac{38}{90})$), we would obtain a sentiment score of -0.578. The second important justification for using TextBlob was a modified version of *Occam's Razor*: simpler solutions are more likely to be correct than complex ones, especially when it comes to programming, and in particular when Ian is involved.

Included with this write-up is our attempted code for running the Stanford CoreNLP server at the end of this document in Section 1.2. We were successful in analyzing one email at a time but were not confident on how to run the server on a loop and have the values stored on a single dataframe.

We have provided the code used the analysis in Section 1.1 and provide a brief summary of what it does. We begin by importing the appropriate modules along with the data, then take the data and convert it to a Panda Dataframe format. We clean the observations of the JSON string formatting by removing the new line and tab coding fragments. Next, we clean the data for forwarded emails in order to avoid biasing the sentiments and then compute the aggregate polarity and subjectivity scores of all the emails in the data. We found that the aggregate polarity score is 0.1932 indicating mildly positive sentiments and the aggregate subjectivity is 0.4970 indicating significant subjectivity in emails. We then compute the polarity and subjectivity of each email body. Figure 2 presents the first five rows of the dataframe containing dates, emails, polarity measures, and subjectivity measures and Figure 3 presents the trends in the results. In addition to the aggregated results, we were also able to distill averaged monthly email polarity and objectivity measures to track changes over time. These results are displayed in Figure 4 below. We see that there seems to be negligible change in the polarity and subjectivity of emails by month. We are not sure how to rationalize this result without having more information. We are limited by the fact that we cannot re-weight the results by incorporating number of employees sending the emails in each month or even the number of emails sent in a given month, which we did not have time to compute given that the provided code in the course server did not work. At the moment,

we are unable to provide a complete interpretation because there are many factors that would need to be considered beforehand.

We hit a significant setback in our analysis fairly late in the process where we realized that the data from the Amazon AWS file only contains the dates and content of the emails with no information regarding senders or recipients. We tried to find ready-for-use data online or to find solutions on how to import the exceptionally large file containing the comprehensive email directories; unfortunately, we were not successful in this endeavor. We were able to find a potential solution[4] but, given time constraints, attempting this solution was no longer a priority.

Elizaveta's Thoughts

Given more time, I believe our method has the potential to be successful but we were unable to properly implement it within the time constraints because we had to learn a lot of Python along the way. I think we were successful in learning programming in Python which was the ultimate goal for us. I lost a fair bit of time trying to get Python and Anaconda running after upgrading from Python 2.7 to Python 3.7. I think the biggest setback we faced along the way, besides the computer issues, was the lack of comprehensive data. We lost a great deal of time trying to get the full directory imported and trying to get the subsample data in a dataframe format. The latter issue was particularly frustrating because we seemed to know what the data contained but lost a great deal of time formatting, which turned out to be a remarkably simple process; honestly, that sequence of events summarizes the whole experience: we banged our heads on the walls at the most trivial things. This has allowed us to have a deeper understanding of what the modules do and how Python reads the code. In terms of actual analysis, we did not really get close to achieving the goal we set for ourselves (analyzing sentiments of RAC over time) but we did lay the foundation for pursuing this task in the future.

Ian's Thoughts

This project was a battle from the very start. I believe Elizaveta and I were very successful in accomplishing the learning objectives for this project, even though the deliverable may fall short of our expectations. Not only did we get to play in Python, we learned a great deal about file management, formatting, and the difficulties associated with having messy directories. Given access to the full $500,000+$

[4] Wordpress blog post on "Mining Mailboxes with Elasticsearch and Kibana"

emails in the general Enron corpus and if we had data that had not be scrubbed of some of the necessary identification information from each email such as the "to" and "from" entries, it would be possible to conduct the intended RAC-centered analysis. I came into this project without significant computer programming experience. Relative to Elizaveta, I am a total novice. However, this changed over the duration of this assignment. Part of the joy of being new to programming, I assume, is that I tried multiple different approaches to solving each problem we faced along the way. Consequently, my contributions to the project were often the intermediate steps, troubleshooting, and in the analysis. While Elizaveta's computer was down during the CoreNLP phase of our project, I was able to run segements of stanfordcorenlp and managed to construct a working interactive sentiment analysis tool using Terminal, which for me was a very proud moment. Ultimately, I give the credit for this project to Elizaveta. She helped form the core structure of our analysis and wrote most of the final iteration of code presented here. I count myself fortunate to work with her and am very glad to have had this opportunity to learn!

References

McLean, Bethany and Peter Elkind (2013) *The Smartest Guys in the Room: The Amazing Rise and Scandalous Fall of Enron*: Penguin.

Related Figures

Figure 1: Example of Sentiment Analysis using Standford CoreNLP

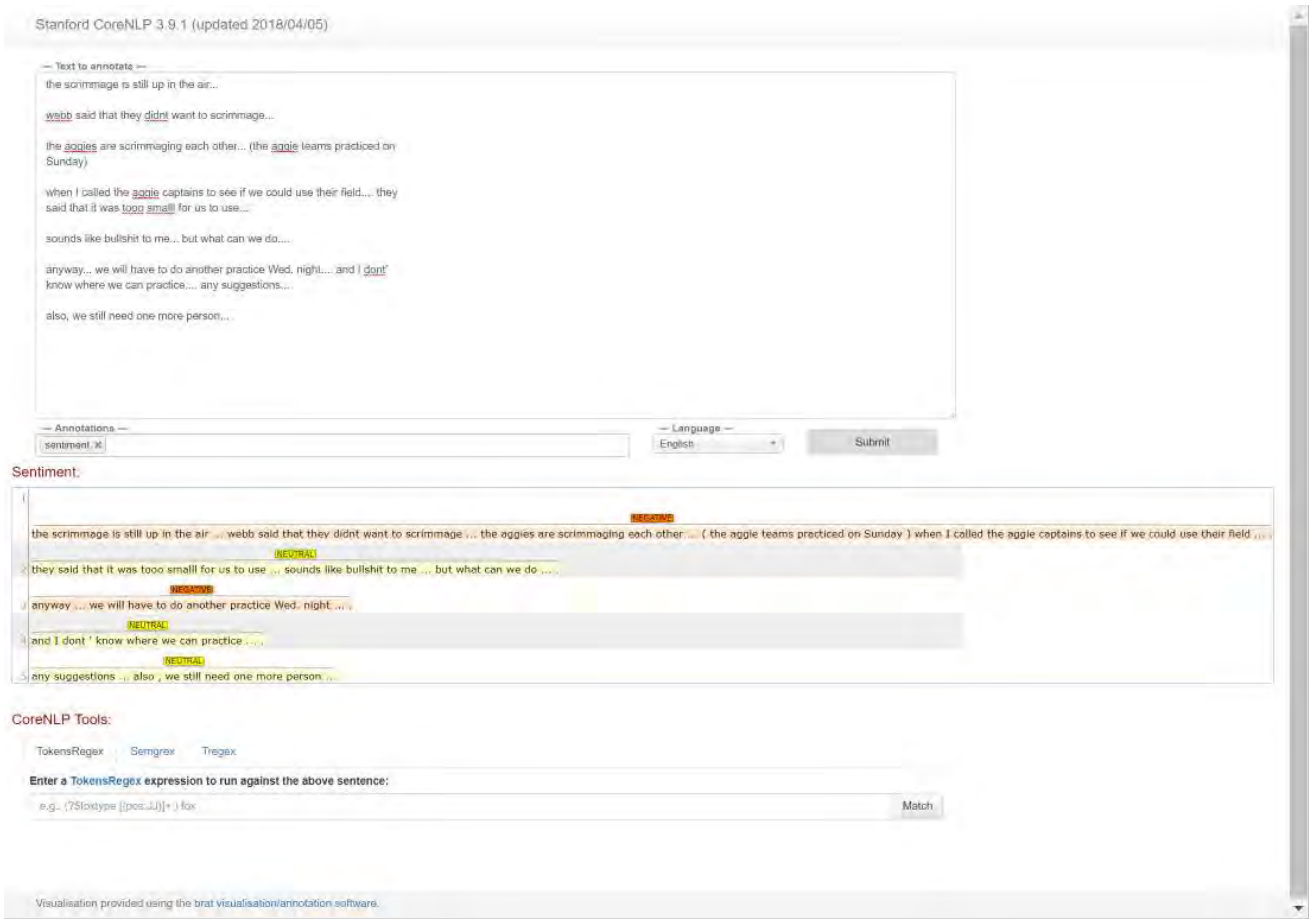


Figure 2: Example Results of TextBlob Analysis

Date	Email Body	pol_sub	Polarity	Subjectivity
2000-11-14 08:22:00-08:00	the scrimmage is still up in the air...webb sa...	(0.1875, 0.4375)	0.187500	0.437500
2000-11-14 07:37:00-08:00	suggestion... make a playbook that just shows ...	(0.13, 0.53)	0.130000	0.530000
2000-11-14 07:25:00-08:00	I'm ready, are you? Did you get my message ab...	(0.39166666666666666, 0.75)	0.391667	0.750000
2000-11-14 08:32:00-08:00	I've had lots of folks ask for more details on...	(0.10378747795414461, 0.49533068783068784)	0.103787	0.495331
2000-11-14 03:11:00-08:00	[IMAGE]Airlines[IMAGE][IMAGE][IMAGE][IMAGE][IM...	(0.28163797061524326, 0.4940469631378721)	0.281638	0.494047

Figure 3: Aggregate Trends in Polarity and Subjectivity

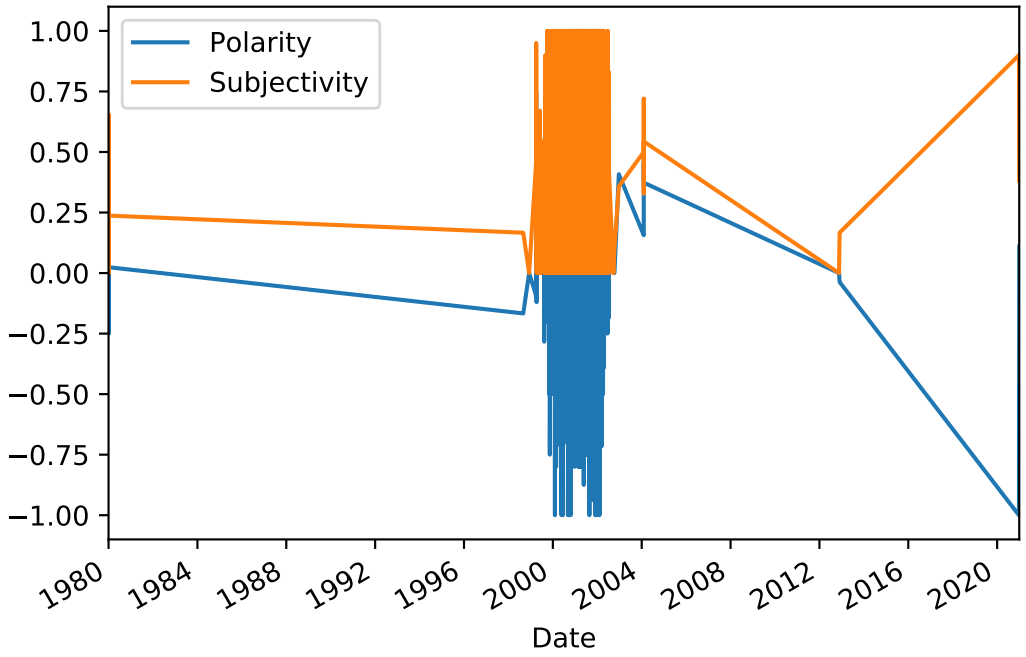
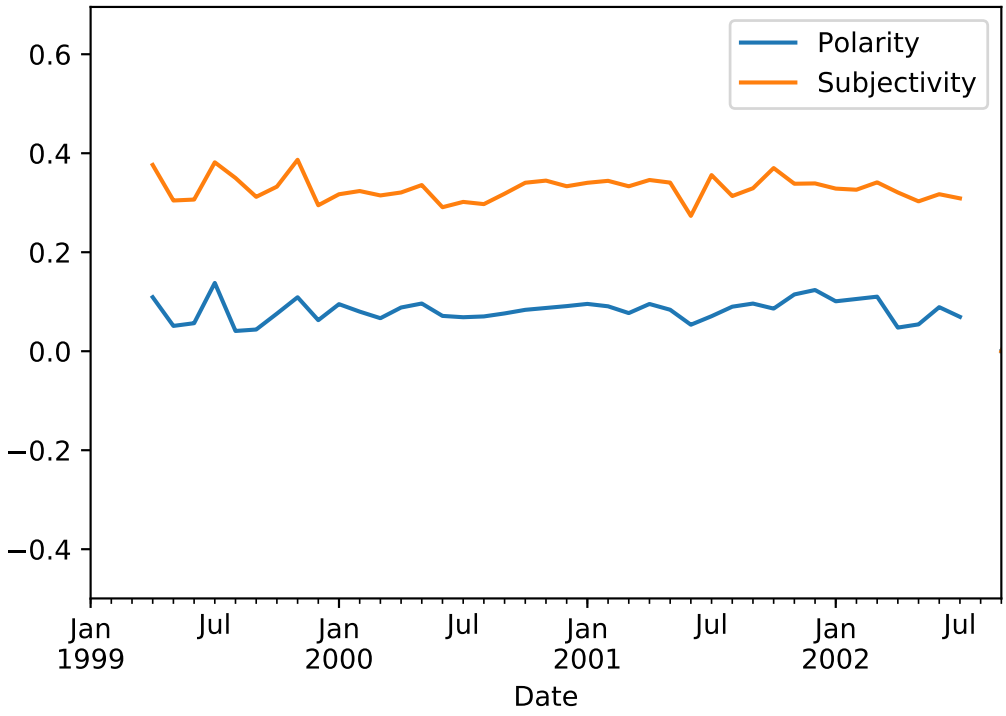


Figure 4: Monthly Trend of Polarity and Subjectivity



1 Project Code

1.1 Code Implemented for Analysis

```
import os
cwd = os.getcwd()
print(cwd)

import platform
print(platform.python_version())

import json
import io
import numpy as np
import pandas as pd
import dateutil
import parser
import modulefinder
import nltk
import requests
import sys
import re
import pycorenlp
import numpy
from pandas.io.json import json_normalize
import matplotlib.pyplot as plt
import time
import datetime
from datetime import datetime
import textblob
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer

# Check that modules loaded properly
'parser' in sys.modules
're' in sys.modules

print(sys.path)

# Import data (either small set of 10,000 emails or larger set of 100,000 emails
```

```
#enron=requests.get('https://s3.amazonaws.com/inta.gatech/mails.txt').json()
enron=requests.get('https://s3.amazonaws.com/inta.gatech/mail_large.txt').json()
print(enron[0:3])

# Convert to dataframe format
data = pd.DataFrame([a['body'] for a in enron], index =pd.to_datetime([i['date']
    for i in enron]), columns = ['Email Body'])
data.head()

# Clean up data
data = data.replace(regex= r'\n', value='')
data = data.replace(regex= r'\t', value='')
data = data.replace(regex= r'\r', value='')
data = data.replace(regex= r'-----Original Message-----.*$', value='')
data = data.replace(regex= r'---------------------- Forwarded by.*$', value='')
data = data.replace(r'^\s*$', np.nan, regex=True)

# Remove any rows with empty emails after cleaning
data = data.replace(' ', np.nan)
data= data.dropna()

# Look at the data
data.index.names = ['Date']
data.head()
data.tail()
numpy.shape(data)

# CONDUCT SENTIMENT ANALYSIS USING TEXTBLOB

tixt = str(data['Email Body'])
text = TextBlob(tixt)

#### The next code will generate an aggregate sentiment/polarity and objectivity
    score in (polarity_score,objectivity_score) format.

# Aggregate sentiment
text.sentiment

# Conduct sentiment analysis on each email body
```

```
def sentiment_func(blah):
    try:
        return TextBlob(blah).sentiment
    except:
        return None

data['pol_sub'] = data['Email Body'].apply(sentiment_func)

data.head()
data.tail()

# Create column for polarity and subjectivity, respectively
data['pol_sub'][0][0]
data['Polarity'] = data['pol_sub'].apply(lambda x: x[0])
data['Subjectivity'] = data['pol_sub'].apply(lambda x: x[1])

data.head()

# Plot the trends
%pylab inline
senti = data.drop(['Email Body', 'pol_sub'], axis=1)
aggregate = senti.plot.line()
plt.savefig('aggregate.pdf')

senti_month = senti.resample('M').mean()
aggregate2 = senti_month.plot.line()
aggregate2.set_xlim(pd.Timestamp('1999-01-01'), pd.Timestamp('2002-09-01'))
plt.savefig('aggregate2.pdf')
```

1.2 CoreNLP Attempted Code

```
# Start the server
    # cd stanford-corenlp-full-2018-10-05
    # java -mx6g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer -port
        9000 -timeout 5000

import pycorenlp
pycorenlp in sys.modules

nlp = StanfordCoreNLP('http://localhost:9000')

# Conduct sentiment analysis on one email body
enron_sample = str(mail_bodies[0:1])
res = nlp.annotate(enron_sample,
                   properties={
                       'annotators': 'sentiment',
                       'outputFormat': 'json',
                       'timeout': 1000,
                   })
for s in res["sentences"]:
    print("%d: '%s': %s %s" % (
        s["index"],
        " ".join([t["word"] for t in s["tokens"]]),
        s["sentimentValue"], s["sentiment"]))

# One of the attempts to run a loop on the CoreNLP server
sentiment_df = pd.Panel(columns=['index', "sentimentValue","sentiment"])

counter = 0
for email in mail_bodies:
    counter += 1
    a=counter -1
    b=counter
    email = nlp.annotate(mail_bodies[a:b],
                         properties={
                             'annotators': 'sentiment',
                             'outputFormat': 'json',
                             'timeout': 1000,
                         })
```

```
    for s in res["sentences"]:
        print("%d: '%s': %s %s" % (
        s["index"],
        " ".join([t["word"] for t in s["tokens"]]),
        s["sentimentValue"], s["sentiment"]))
    sentiment_df = sentiment_df.append({'index': s['index'], "sentimentValue": s[
       "sentimentValue"], "sentiment": s["sentiment"]}, ignore_index=True)

print(sentiment_df)
```