HOW DOES HILLARY CLINTON CORRESPOND WITH HER CONFIDANTS?

MUNEEB ALAM

ABSTRACT

In this study I compare Hillary Rodham Clinton's released emails as Secretary of State and some of her comments in interviews for sentiment. I find that she is not detectably different in her State Department correspondence (usually on foreign policy issues) to longtime aides and friends than to lesser-known acquaintances than she is speaking to the public in interviews. This suggests that Clinton does not put on a dramatically different persona in public than in private and therefore that voters should take her public comments at face value.

INTRODUCTION

The political climate in the United States today is one of distrust of the establishment. This is most clearly seen from the strong presidential campaigns of Donald Trump and Bernie Sanders, both of whom regularly criticize people who they see as too corrupted by the system—Jeb Bush and Marco Rubio, for example, on the Republican side, and Hillary Clinton on the Democratic side. Nearly two-thirds of voters distrust either Trump or Clinton, the two frontrunners (Chozick, New York Times).

Former presidential candidate and Trump supporter Dr. Ben Carson said that voters worried about Trump's bigotry should consider that his public persona is an act—it differs from his work persona (Parker, New York Times). While that claim cannot be evaluated given the lack of information about Trump's life outside of the public eye, the investigation into Clinton's use of her private email server while serving as Secretary of State and subsequent email release under the Freedom of Information Act allows us a (selected) look into *her* thoughts and behavior in "private."

Although Clinton does not face the same issue of an awful public persona like Trump, she is commonly accused of being duplicitous and altering her position based on her donors, public opinion, or her audience by Sanders and his supporters; while this analysis cannot make people trust her, it can start to answer, perhaps, to what extent she does change, and therefore how predictive her comments and demeanor so far will be of her performance as the most important office-holder on the planet.

I am unable to find a difference between how Clinton addresses people widely considered to be part of her inner circle (the "ground truth") and people outside of it in terms of frequency of sentiment expressed, suggesting that, in that sense, she does not write to either group all that differently. I am also unable to find a difference between how Clinton addresses people in private emails and in public interviews. This suggests that she may be more genuine than she's given credit for my her opponents.

MOTIVATION

Mainstream Clinton email analysis thus far has mainly been searches for specific topics (made easy by sites like WikiLeaks and the Department of Justice itself). It is not an efficient way to analyze her entire set of released emails.

After Kaggle made some of her emails available, many people did some basic positive/negative sentiment analysis (see "Hillary's Sentiment", for example), noting the trickiness of getting sensible answers. NYC Data Science Academy did one of the best put-together projects, but its interactive visualization is no longer active.

I seek to analyze her entire email corpus for consistency in sentiment, both within the corpus and between it and her interviews. I choose to use Python for this analysis because of my comfort in the language and its speed for analyzing strings of text (for which a language like R can be a little awkward). I make use of the Natural Language Toolkit (NLTK) and pandas packages for processing the data and scikit-learn for machine learning, standard fare for working with language, data, and machine learning in Python.

DATA SOURCES

Sentiment analysis is a highly imperfect subject. Typical short lexicons with a few hundred words grouped into positive and negative sentiments will often identify short emails as neutral due to lack of an identified word, let alone more specific emotions or degrees of sentiment. I chose Saif Mohammad's

"EmoLex" lexicon (Mohammad) for its length (over 14,000 words) and emotional classification, associating words with one or more of eight emotions based on Plutchik's Wheel of Emotions in addition to positive or negative sentiment (Mohammad and Turney). Although it is crowdsourced and as such may suffer a little in quality, I thought its sheer breadth made it worth the cost. It is available as a tab-delimited file (in "long" format) on Mohammad's website. One attempt on Kaggle ("Sentiment Analysis") did attempt to use this lexicon, but the code has bugs and did not give a result. See Appendix A for the Python code to load the dictionary and Appendix D for using it for feature extraction.

Table 1. Summary of 14182 entries in NRC EmoLex		
Sentiment/Emotion	Words	% of total
Joy	689	4.9%
Sadness	1191	8.4%
Anger	1247	8.8%
Fear	1476	10.4%
Trust	1231	8.7%
Disgust	1058	7.5%
Surprise	534	3.8%
Anticipation	839	5.9%
Positive	2312	16.3%
Negative	3243	22.9%
Neutral	8627	60.8%

The Kaggle email dataset does not contain all 30,000-plus emails, nor is it particularly clean, with longtime Clinton confidants like Jake Sullivan and Huma Abedin appearing under a slew of different aliases and email addresses (likely both due to Clinton's own typographical inconsistencies and errors as well as Kaggle's own PDF-to-text conversion). The data as scraped and formatted by the Wall Street Journal and WikiLeaks (hosted by the latter) is far cleaner and was used for this task. See Appendix B for the scraping code and Appendix C for the attribute (e.g. sender, date, subject, text) extraction code.

Clinton was the most frequent sender and recipient of emails. She was the primary sender on over 7000 (the majority to Sullivan, Abedin, and Cheryl Mills), and the primary recipient on over 17,000 (from over 300 people as identified by the WSJ). The average email length was 109 words but the standard deviation was 344 words—there are many emails several hundred or even thousands of words long (though the longest ones are more likely due to a PDF-to-text parsing error or reading an entire email thread rather than just the single Clinton email than actually being 20,000 words long).

I needed another dataset in order to make a comparison between her voice to her confidants (via email) and her voice when she communicates with others. While I do not have access to her, say, speeches to Goldman Sachs, Clinton does have an extensive documented public speaking record. For an "apples-to-apples" comparison, I restricted by search to her comments on foreign policy. In particular, I used four of her foreign policy speech and interview transcripts, helpfully compiled by the Council on Foreign Relations. Two were in interviews with CFR itself (one on counterterrorism and another on strategic interests), while another was with the Atlantic on Middle East policy and a fourth was with PBS on trade. This data resulted in 356 observations.

METHODS

Preparing data

I first scraped the email data from WikiLeaks (Appendices B and C). I extracted sentiments from the

emails using the following "bag of words" approach (Appendix D):

1. Replace question marks, exclamation points, and semicolons (i.e. punctuation marks which frequently end independent clauses) with periods, and split the email into its sentences via the period-space two-character combination.
2. For each sentence:
    a. Remove commas, dashes, and hyphens.
    b. Split it by spaces into its constituent words. For each (put-into-lowercase) word:
        i. If it is a negation word, switch the polarity of the sentiment.
        ii. If it is a stop word (identified via NLTK), ignore.
        iii. If it is in EmoLex:
            1. If current polarity is positive, increment the count of each sentiment or emotion associated with the word by one.
            2. If current polarity is negative, increment the count of the opposite sentiment or emotion by one.
        iv. If it is not, but the stem (via NLTK's "english" SnowballStemmer) is, then perform the same process on the stem.
3. Sum over the sentences to get a sentiment and emotion count for that email.

I then extract Clinton's comments from her speeches (Appendix E). I grouped sentiments (identified using the same method as above) by paragraph, which is nothing other than a human—helpfully—grouping sentences by ideas. Each paragraph can then be considered the equivalent of an email.

Identifying confidants and email comments

Next, I try to identify the people closest to Clinton. On a simple level, this could be the people who receive the most emails from her, either a top-N or above an arbitrary limit.

Table 2. People who received over 100 emails from Clinton

Name	Emails received from Clinton
Jake Sullivan	1403
Cheryl Mills	1136
Huma Abedin	872
Lauren Jiloty	717
Robert Russo	522
Lona Valmoro	438
Monica Hanley	434
Philippe Reines	172
Sidney Blumenthal	143
Melanne Verveer	114
Anne-Marie Slaughter	112

Based on this list, cutting off the confidant list at Hanley seems reasonable, although Clinton's ties to Blumenthal (a longtime advisor) and Slaughter (a former top aide) are also well-documented. Furthermore, there's no guarantee the people at the top of the list are confidants—they are all also work colleagues in some sense. On the other hand, as Vanity Fair notes, Clinton's network has been built up over two-plus decades and is, unsurprisingly, quite extensive.

To identify her closest confidants, I try classify emails from selected individuals who most likely are confidants—Sullivan, Mills, Abedin, Reines, Blumenthal, and Maggie Williams — in one group and

those who only received 20 emails from Clinton or fewer as not. (The email sample sizes are 3731 and 553, respectively.) From this data, I use a support vector machine classifier to try to separate confidant and acquaintance emails using the word count (transformed via natural logarithm because word counts vary by three orders of magnitude), sentiment counts per 100 words, and time of day (see Figure 1), then try to use this classifier to find people whose emails largely fall under the "confidant" category.

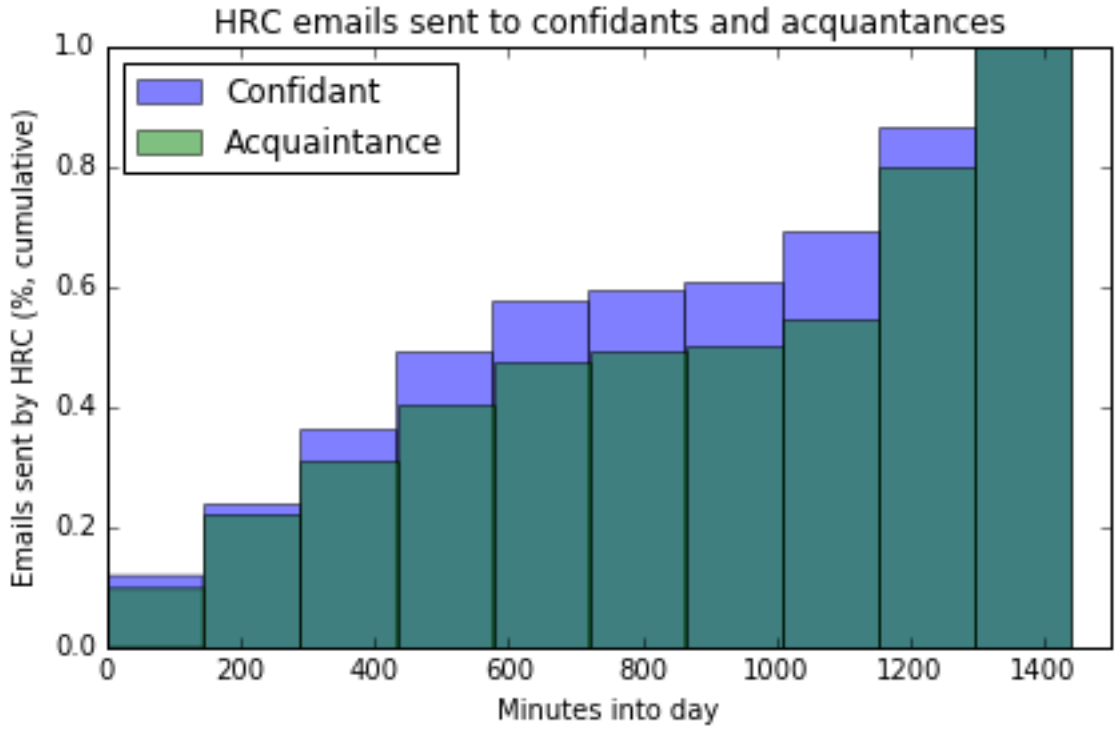

Figure 1. Emails sent by Clinton by time of day to her top few confidants and her lesser-contacted acquaintances. She generally contacts her confidants relatively more in the mornings. Time of day could then be a useful separator between confidant emails and acquaintance emails.

I next compare the confidant emails data with the speeches data and train a classifier to separate between the two. If there is a difference, it suggests that Clinton is different when discussing matters with her inner circle than in the public space—expected, perhaps, but difficult for voters to account for given that all they usually know of are her public comments.

RESULTS

Table 3 lists the results of the attempt at training an SVM classifier to separate confidant emails from acquaintance emails.

Table 3. SVM Classifier Confusion Matrix (training)		
	Expected confidant emails	Expected acquaintance emails
Actual confidant email	1844 (100%)	0 (0%)
Actual acquaintance email	114 (39.7%)	173 (60.3%)

It avoids false negatives entirely, but has more trouble with false positives. Nearly 40% of acquaintance

emails were flagged as confidant emails in the training data.

Table 4. SVM Confusion Matrix (test)		
	Expected confidant emails	Expected acquaintance emails
Actual confidant email	1885 (99.9%)	2 (0.1%)
Actual acquaintance email	257 (98.5%)	4 (1.5%)

The classifier does a very poor job handling true acquaintance emails, flagging almost all of them as positive. About 88% of test data are confidant emails, but the classifier predicts nearly 100% of the observations to be confidant emails. (In fact, it predicted that 99.9% of the unclassified emails were from confidants.) It is clearly unreliable.

Next, I trained a random forest classifier to use instead. It worked much better than the SVM, although it was still far from satisfactory at classifying acquaintance emails.

Table 5. Random Forest Classifier Confusion Matrix (training)		
	Expected confidant emails	Expected acquaintance emails
Actual confidant email	1840 (99.8%)	4 (0.2%)
Actual acquaintance email	19 (6.6%)	268 (93.4%)

Table 6. Random Forest Confusion Matrix (test)		
	Expected confidant emails	Expected acquaintance emails
Actual confidant email	1760 (93.3%)	127 (6.7%)
Actual acquaintance email	219 (83.9%)	42 (16.1%)

The random forest classifier also predicts that the vast majority of new data (92%, to be specific) will be from confidants.

Rather than poor model specification or feature selection, this lopsided result could also be explained if the *a priori*–chosen confidants I selected do not, in fact, receive emails from Clinton that are fairly different from the rest—not even Sidney Blumenthal, who stands out in certain respects from the others.

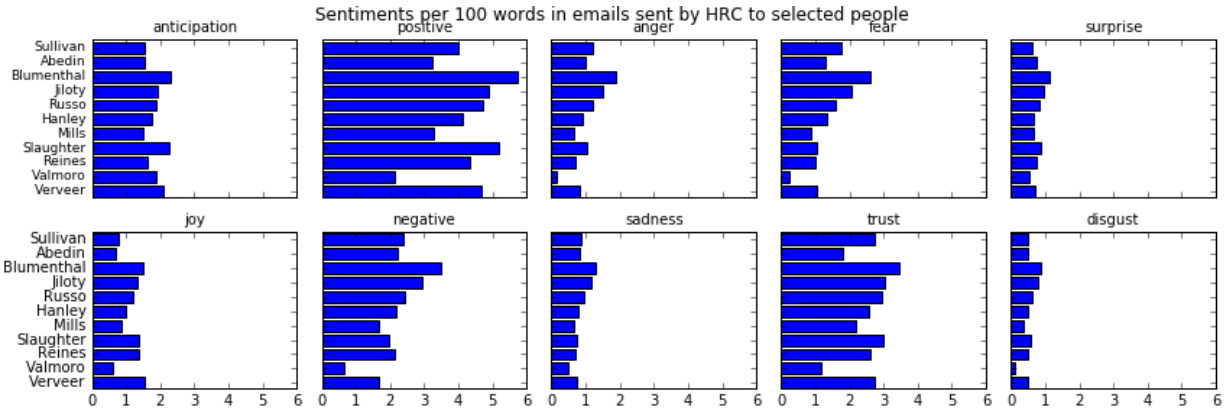

Figure 2. Sidney Blumenthal often received more sentiments per 100 words from Hillary Clinton than others, but still not consistently enough to separate himself from the rest of her email recipients.

In other words, the more personal emails Clinton sends them are swamped by the mundane ones.

Table 7. Random Forest Prediction for New Data	
Category	Count
Predicted confidant	3040 (92.4%)
Predicted acquaintance	249 (7.6%)

So while particular confidants may have certain notable characteristics, on the whole, the confidant group is not. It may be best to consider the entire set of emails, then, as going to "confidants"—after all, none of them were intended for release in the first place.

Identifying speeches

Next, I try to separate speeches from emails. I combine the data and separate into training and test, but the classifiers are again not able to identify a good decision rule.

Table 8. Random forest Confusion Matrix (training)		
	Expected emails	Expected speech
Actual email	3768 (99.9%)	5 (0.1%)
Actual speech	121 (68.4%)	56 (31.6%)

Table 9. Random forest Confusion Matrix (test)		
	Expected emails	Expected speech
Actual email	3774 (99.4%)	21 (0.6%)
Actual speech	176 (98.3%)	3 (1.7%)

I also attempt to train a logistic classifier, but it simply flags every observation as positive. This was also the case training the classifier on both the linear and quadratic forms of the variables (that is, 20 predictors). Again, the observations in the two groups are not similar enough intra-group and are too similar inter-group.

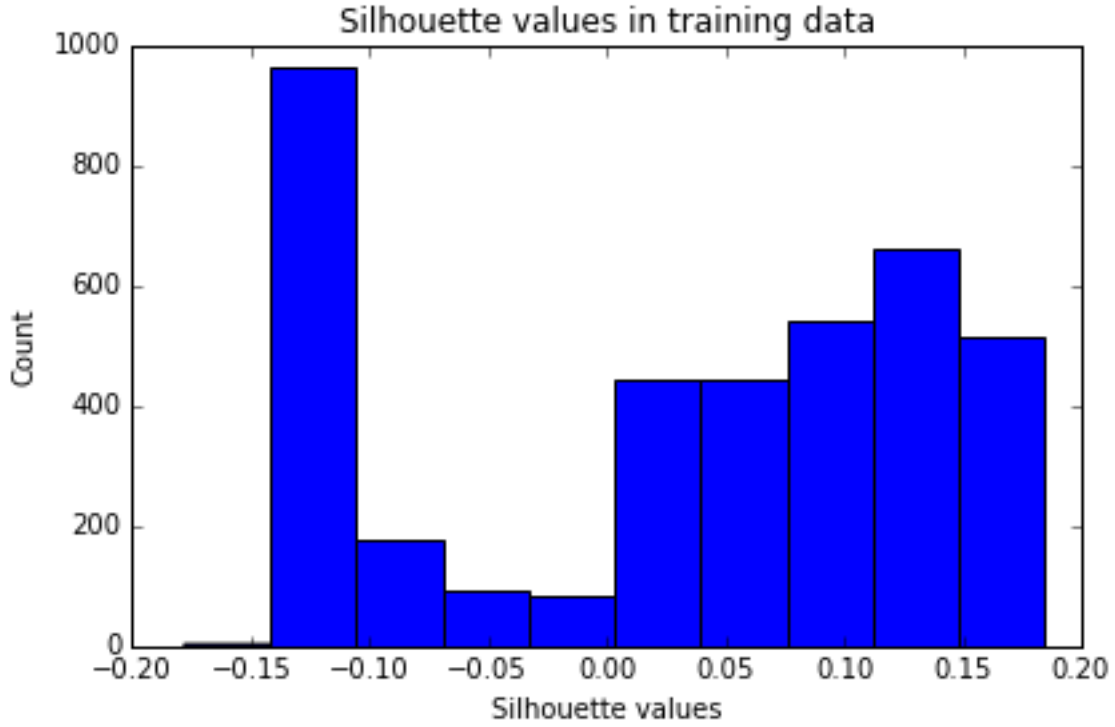

Figure 3. Silhouette values in the email-speech training data. Values near 1 indicate similarity and values near -1 indicate dissimilarity.

CONCLUSIONS

The data as they are suggest there is no significant emotional difference between Hillary Clinton in writing privately to confidants, Clinton writing to acquaintances, and Clinton speaking publicly. Assuming Clinton truly does not communicate differently with her confidants and with her acquaintances (quantity of emails aside), the result suggests one or more of a few possibilities:

1. They are all her confidants to varying degrees. This is possible given the Clintons' large network and the sample bias here in that Clinton finds all these people important enough to contact herself rather than through her staff
2. None of them are her confidants. This is also possible given that she would not be releasing truly personal correspondence and that, presumably, she has a life outside of her work in government.
3. She communicates with her confidants in other ways, or communicates differently in unreleased emails.

The implication of the similarity between emails and speech is that she might be somewhat genuine—maybe surprising given how distrusted she us, but perhaps not given her (at times) awkwardness in public relative to consistent crowd-pleasers like Trump, Sanders, or even husband Bill Clinton. Thus, while in 1999 perhaps one could not truly know Clinton through her public persona only (as Williams notes in the Vanity Fair piece), one might know her a little better via only her public comments in 2016.

Given the imperfections in machine learning methods for language (used and available), none of these results can be accepted confidently. The sentiment analysis here was basic—consideration of subject matter (like through latent Dirichlet allocation), consideration of other topics in the same email thread,

and models other than “bag of words,” to name a few—and it is possible that a classifier would have more success with more sophisticated features. Future studies should expand the speech dataset, find a better way to compare them to emails (which are fundamentally different in nature from speech), try to find and train on a relevant corpus, consider sentiments on specific subjects instead of in general (see e.g. Zaidkhan), and consider other public figures who have had emails released (like Sarah Palin in 2011) to see whether failure to reject the null might simply be “normal.”

REFERENCES

Chozick, Amy. "Emails Add to Hillary Clinton's Central Problem: Voters Just Don't Trust Her." *New York Times.* May 25, 2016. Accessed May 28, 2016. http://www.nytimes.com/2016/05/26/us/politics/hillary-clinton-emails-campaign-trust.html

Cillizza, Chris. "Did Bernie Sanders deliver an ultimatum that Debbie Wasserman Schultz should be fired?" *Washington Post.* May 26, 2016. Accessed May 28, 2016. https://www.washingtonpost.com/news/the-fix/wp/2016/05/26/it-sure-sounds-like-bernie-sanders-is-making-an-ultimatum-about-debbie-wasserman-schultz/

Council on Foreign Relations. "Hillary Rodham Clinton on Strategic Interests, Values, and Hard Choices." Accessed May 28, 2016. http://www.cfr.org/united-states/hillary-rodham-clinton-strategic-interests-values-hard-choices/p35672

Council on Foreign Relations. "Hillary Clinton on National Security and the Islamic State." Accessed May 28, 2016. http://www.cfr.org/radicalization-and-extremism/hillary-clinton-national-security-islamic-state/p37266

Council on Foreign Relations. "The Presidential Candidates: In Their Own Words." Accessed May 28, 2016. http://www.cfr.org/elections/presidential-candidates-their-own-words/p36633

Ellison, Sarah. "How Hillary Clinton's Loyal Confidants Could Cost Her The Election." *Vanity Fair*. November 2015. Accessed May 28, 2016. http://www.vanityfair.com/news/2015/10/hillary-clinton-inside-circle-huma-abedin

Goldberg, Jeffrey. "Hillary Clinton: 'Failure' to Help Syrian Rebels Led to the Rise of ISIS." *The Atlantic*. August 10, 2015. Accessed May 28, 2016. http://www.theatlantic.com/international/archive/2014/08/hillary-clinton-failure-to-help-syrian-rebels-led-to-the-rise-of-isis/375832/

Kaggle. "Hillary's Sentiment About Countries." Accessed May 28, 2016. https://www.kaggle.com/operdeck/d/kaggle/hillary-clinton-emails/hillary-s-sentiment-about-countries/code

Kaggle. "Sentiment Analysis: 3 Different Methods." Accessed May 31, 2016. https://www.kaggle.com/apasupat/d/kaggle/hillary-clinton-emails/sentiment-analysis-3-different-methods/run/144754

Natural Language Toolkit. http://www.nltk.org/

Parker, Ashlee. "Two Donald Trumps' Emerge, the Public One and the Private One." *New York Times.* March 11, 2016. Accessed May 28, 2016. http://www.nytimes.com/politics/first-draft/2016/03/11/two-donald-trumps-emerge-the-public-one-and-the-private-one/

PBS Newshour. "Full Interview: Hillary Clinton on trade pact doubts, dealing with Putin." Accessed May 28, 2016. http://www.pbs.org/newshour/bb/full-interview-hillary-clinton-trade-pact-doubts/

Saif Mohammad. "NRC Word-Emotion Association Lexicon." Accessed May 28, 2016. http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

Mohammad, Saif M., and Turney, Peter D. (paper presented at Proceedings of the NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, Los Angeles, California, June 2010).

NYC Data Science Academy. "The Hillary Clinton Email Explorer." Last updated December 5, 2015. Accessed May 28, 2016. http://blog.nycdatascience.com/students-work/the-hillary-clinton-email-explorer/

WikiLeaks. "Hillary Clinton Email Archive." Accessed May 28, 2016. https://www.wikileaks.org/clinton-emails/

Zaidkhan. "Analyzing Hillary Clinton's Emails." Last updated October 12, 2015. Accessed May 28, 2016. http://zaidkhan.me/2015/hillary/

APPENDIX A.

Loading EmoLex into a Python dictionary using Python 3.4.

```
def read_nrc():
    #returns {word: (positive/negative, {emotions})}
    r = open(' /NRC-Emotion-Lexicon/NRC-emotion-lexicon-wordlevel-alphabetized-v0.92.txt')
    data = r.read().strip()
    r.close()

    firstword = 'aback'
    data = data[data.index(firstword):].split('\n')
    lex = {}
    generalset = {'positive', 'negative'}
    for line in data:
        word, emot, posneg = line.split('\t')
        posneg = int(posneg)
        if word not in lex:
            lex[word] = [None, set()]
        if posneg == 1:
            if emot in generalset:
                lex[word][0] = emot
            else:
                lex[word][1].add(emot)
    return lex
sent_dct = read_nrc()
```

APPENDIX B.

Scraping WikiLeaks for Clinton's emails.

```
import aiohttp
import asyncio
from os.path import exists
LIMIT = 30322
req_lim = 100
urlbase = 'https://www.wikileaks.org/clinton-emails/emailid/'

@asyncio.coroutine
def get(*args, **kwargs):
    response = yield from aiohttp.request('GET', *args, **kwargs)
    return (yield from response.text())

@asyncio.coroutine
def extract_text(url, sem):
    with (yield from sem):
        page = yield from get(url)
        w = open(get_save_file(url=url), 'w')
        w.write(page)
        w.close()
        emailid = int(url[url.rfind('/')+1:])
        if emailid % 1000 == 0:
            print('Writing', emailid)

def generate_links(start):
    return [urlbase+str(x) for x in range(start, LIMIT+1) if not exists(get_save_file(x))]

def get_save_file(emailid=None, url=None):
    if emailid is None:
        emailid = int(url[url.rfind('/')+1:])
    return str(emailid) + '.txt'

@asyncio.coroutine
def run(links):
    sem = asyncio.Semaphore(req_lim)
    fetchers = [extract_text(link, sem) for link in links]
    return [(yield from f) for f in asyncio.as_completed(fetchers)]

loop = asyncio.get_event_loop()
x = loop.run_until_complete(run(generate_links(1)))
```

APPENDIX C.

Email attribute extraction from the WikiLeaks-hosted Clinton email pages.

```
from datetime import datetime
def parse_file(emailid):
    r = open(get_save_file(emailid))
    data = r.read()
    r.close()

    sent = 'n/a'
    released = 'n/a'
    sender = 'n/a'
    recipient = 'n/a'
    text = 'n/a'
    subject = 'n/a'

    data = data[data.index('Share document'):]

    if 'Original Message' in data:
        data = data[:data.index('Original Message')]
    else:
        data = data[:data.index('</div>')]

    i = data.index('From:')
    data = data[i:]
    sender = data[data.index('>')+1:data.index('</span>')].strip()

    i = data.index('To:')
    data = data[i:]
    recipient = data[data.index('>')+1:data.index('</span>')].strip()

    i = data.index('Date:')
    data = data[i:]
    sent = data[5:data.index('Subject')].strip()
    sent = sent[:sent.index(' ')]
    y, m, d = [int(x.strip()) for x in sent.split('-')]
    sent = datetime(y, m, d)

    i = data.index('Subject:')
    data = data[i:]
    subject = data[8:data.index('</header>')].strip()

    if emailid == 7632:
        i = data.index('D ate:')
    elif emailid == 9700:
        i = data.index('D6a t e') + 2
    elif emailid == 14704:
        i = data.index('Date.')
    elif emailid == 23805:
        i = data.index('D a te :') + 3
    else:
        i = data.index('Date:')
    data = data[i:]
    released = data[5:data.index('</span>')].strip().replace(' ', '')
    if released[-1] == '-':
```

```
        released = released[:-1]
    if released[0] == ':':
        released = released[1:]
    if len(released) > 10:
        released = released[:10]
    m, d, y = [int(x.strip()) for x in released.split('/')]
    released = datetime(y, m, d)

    data = data.split('</span>')[-1]
    text = data.replace('<span class="inlinemeta">', '').strip()

    return sent, released, sender, recipient, subject, text

data = []
skip = {15406}
for i in range(LIMIT):
    if (i+1) not in skip:
        try:
            data.append(parse_file(i+1))
        except Exception as e:
            print(i+1, e, e.args)
            break
    if (i+1) % 1000 == 0:
        print('Done extracting through', i+1)
df = pd.DataFrame.from_records(data)
df.columns = ['Sent', 'Released', 'Sender', 'Recipient', 'Subject', 'Text']
```

APPENDIX D.

Extracting sentiments from the emails.

```
opposites = {'anger': 'fear', 'anticipation': 'surprise', 'disgust': 'trust', 'fear': 'anger',
          'joy': 'sadness', 'sadness': 'joy', 'surprise': 'anticipation', 'trust': 'disgust',
          'positive': 'negative', 'negative': 'positive'}
negation_words = {'not'}
from nltk.corpus import stopwords
stop_word_lst = set(stopwords.words('english'))
from nltk.stem.snowball import SnowballStemmer

def split_by_sentences(text):
    return text.replace('?', '.').replace('!', '.').replace(';', '.').split('. ')

def split_by_words(text):
    return [x.strip() for x in text.replace(',', '').replace('-', ' ')
            .replace(':', '').replace('—', '').split(' ')]

def findemot(text):
    emots = {'anger': 0, 'anticipation': 0, 'disgust': 0, 'fear': 0,
        'joy': 0, 'sadness': 0, 'surprise': 0, 'trust': 0}
    posnegneu = {'positive': 0, 'negative': 0, 'neutral': 0}
    wordcount = 0
    #https://en.wikipedia.org/wiki/Contrasting_and_categorization_of_emotions#Plutchik.27s_wheel_of_emotions
    #emotions have opposites
   try:
        sentences = split_by_sentences(text)
        if len(sentences) == 1:
            negation = False
            words = split_by_words(text)
            for word in words:
                wordcount += 1
                if word in negation_words:
                    negation = not negation
                if word in stop_word_lst:
                    pass
                else:
                    if word in sent_dct:
                        if sent_dct[word][0] is None:
                            posnegneu['neutral'] += 1
                        else:
                            if negation:
                                posnegneu[opposites[sent_dct[word][0]]] += 1
                            else:
                                posnegneu[sent_dct[word][0]] += 1
                        for em in sent_dct[word][1]:
                            if negation:
                                emots[opposites[em]] += 1
                            else:
                                emots[em] += 1
                    else:
                        word2 = SnowballStemmer("english").stem(word)
                        if word2 in sent_dct:
                            if sent_dct[word2][0] is None:
                                posnegneu['neutral'] += 1
```

```
                    else:
                        if negation:
                            posnegneu[opposites[sent_dct[word2][0]]] += 1
                        else:
                            posnegneu[sent_dct[word2][0]] += 1
                    for em in sent_dct[word2][1]:
                        if negation:
                            emots[opposites[em]] += 1
                        else:
                            emots[em] += 1
        else:
            sents = [findemot(sen) for sen in sentences]
            for sen_pnn, sen_emot, wc in sents:
                wordcount += wc
                for sen in sen_pnn:
                    posnegneu[sen] += sen_pnn[sen]
                for sen in sen_emot:
                    emots[sen] += sen_emot[sen]
    except AttributeError:
        pass
    return posnegneu, emots, wordcount

def extract_anger(dcts):
    return dcts[1]['anger']
def extract_anticipation(dcts):
    return dcts[1]['anticipation']
def extract_disgust(dcts):
    return dcts[1]['disgust']
def extract_fear(dcts):
    return dcts[1]['fear']
def extract_joy(dcts):
    return dcts[1]['joy']
def extract_sadness(dcts):
    return dcts[1]['sadness']
def extract_surprise(dcts):
    return dcts[1]['surprise']
def extract_trust(dcts):
    return dcts[1]['trust']
def extract_positive(dcts):
    return dcts[0]['positive']
def extract_negative(dcts):
    return dcts[0]['negative']
def extract_neutral(dcts):
    return dcts[0]['neutral']
def extract_wordcount(dcts):
    return dcts[2]

emots = {'anger': extract_anger, 'anticipation': extract_anticipation,
         'disgust': extract_disgust, 'fear': extract_fear,
         'joy': extract_joy, 'sadness': extract_sadness,
         'surprise': extract_surprise, 'trust': extract_trust}
posnegneu = {'positive': extract_positive, 'negative': extract_negative,
             'neutral': extract_neutral}
df['sentinfo'] = df['Text'].apply(findemot)
for e in emots:
    df[e] = df['sentinfo'].apply(emots[e])
```

```
for e in posnegneu:
    df[e] = df['sentinfo'].apply(posnegneu[e])
df['wordcount'] = df['sentinfo'].apply(extract_wordcount)

df.drop('sentinfo', axis=1, inplace=True)
df.to_csv('wikiemails.csv')
```

APPENDIX E.

Finding Clinton confidants.

```
confidants = {'Jake Sullivan', 'Cheryl Mills', 'Huma Abedin', 'Philippe Reines', 'Sidney Blumenthal', 'Maggie
Williams'}
acq_lim = 20

n = 0
for c in confidants:
    n += len(df[(df['Sender'] == 'Hillary Clinton') & (df['Recipient'] == c)])

vc = df[df['Sender'] == 'Hillary Clinton']['Recipient'].value_counts()
kv = [(k, v) for k, v in zip(vc.keys(), vc)]
n = 0
acq = set()
for k, v in kv:
    if v <= acq_lim:
        n += v
        acq.add(k)

def acq_or_conf(s):
    return s in acq or s in confidants
def acquaintance(s):
    return (s in acq)
def conf(s):
    return (s in confidants)
def get_min_of_day(s):
    s2 = s.split(':')
    hour = int(s2[0][-2:])
    minute = int(s2[1])
    return 60*hour + minute
def train_or_test(s):
    return int(random.random()+0.5)
df['Acq'] = df['Recipient'].apply(acquaintance)
df['Conf'] = df['Recipient'].apply(conf)
df['TrainAcqConf'] = df['Recipient'].apply(acq_or_conf)
df['MinOfDay'] = df['Sent'].apply(get_min_of_day)
df['trainingdata'] = df['Sent'].apply(train_or_test)
training = df[(df['Sender'] == 'Hillary Clinton') & (df['TrainAcqConf'] == True)
            & (df['trainingdata'] == 1)]
test = df[(df['Sender'] == 'Hillary Clinton') & (df['TrainAcqConf'] == True)
            & (df['trainingdata'] == 0)]
newsample = df[(df['Sender'] == 'Hillary Clinton') & (df['TrainAcqConf'] == False)]

labs = ['wordcount', 'minofday']
X = [[x for x in np.log(training['wordcount'])], [x for x in training['MinOfDay']]]
for e in emots:
    X.append([x for x in training[e + 'rate']])
    labs.append(e)
y = [1 if x else 0 for x in training['Conf']]
labs.append('confidant')
"""
w = open(folder2 + 'conf classifier.csv', 'w')
w.write(','.join(labs) + ',confidant')
for i in range(len(X[0])):
```

```
    w.write('\n')
    for j in range(len(X)):
        w.write(str(X[j][i]) + ',')
    w.write(str(y[i]))
w.close()
"""
X = np.array(X).T

clf = SVC()
#clf = RandomForestClassifier()
#clf = LogisticRegression()
clf.fit(X, y)

X2 = [[x for x in np.log(test['wordcount'])], [x for x in test['MinOfDay']]]
for e in emots:
    X2.append([x for x in test[e + 'rate']])
y2 = np.array([1 if x else 0 for x in test['Conf']])
X2 = np.array(X2).T

X3 = [[x for x in np.log(newsample['wordcount'])], [x for x in newsample['MinOfDay']]]
for e in emots:
    X3.append([x for x in newsample[e + 'rate']])
y3 = np.array([1 if x else 0 for x in newsample['Conf']])
X3 = np.array(X3).T

p = clf.predict(X)
p2 = clf.predict(X2)
p3 = clf.predict(X3)

tp = 0
tn = 0
fp = 0
fn = 0
for i in range(len(y)):
    if p[i] >= 0.5:
        if y[i] == 1:
            tp += 1
        else:
            fp += 1
    else:
        if y[i] == 1:
            fn += 1
        else:
            tn += 1
print('Training')
print('True positives', tp, 100*tp/len([yi for yi in y if yi == 1]))
print('False positives', fp, 100*fp/len([yi for yi in y if yi == 0]))
print('True negatives', tn, 100*tn/len([yi for yi in y if yi == 0]))
print('False negatives', fn, 100*fn/len([yi for yi in y if yi == 1]))

tp = 0
tn = 0
fp = 0
fn = 0
for i in range(len(y2)):
    if p2[i] >= 0.5:
```

```
        if y2[i] == 1:
            tp += 1
        else:
            fp += 1
    else:
        if y2[i] == 1:
            fn += 1
        else:
            tn += 1
print('Test')
print('True positives', tp, 100*tp/len([yi for yi in y2 if yi == 1]))
print('False positives', fp, 100*fp/len([yi for yi in y2 if yi == 0]))
print('True negatives', tn, 100*tn/len([yi for yi in y2 if yi == 0]))
print('False negatives', fn, 100*fn/len([yi for yi in y2 if yi == 1]))

pos = 0
neg = 0
for i in range(len(y3)):
    if p3[i] >= 0.5:
        pos += 1
    else:
        neg += 1
print('New data')
print('Positives', pos, 100*pos/len(p3))
print('Negatives', neg, 100*neg/len(p3))
```

APPENDIX F.

Finding a classification rule for emails and speeches.

```
cols_needed = ['surprise', 'sadness', 'anger', 'joy', 'disgust',
            'trust', 'fear', 'anticipation', 'positive', 'negative', 'wordcount']
cols_needed += [c+'rate' for c in cols_needed.copy()[:-1]]
cols_needed = {c: [] for c in cols_needed}
source = [x for x in df2['Source']]
source += [x for x in df[df['Sender'] == 'Hillary Clinton']['Sender']]
for c in cols_needed:
    cols_needed[c] += [x for x in df2[c]]
    cols_needed[c] += [x for x in df[df['Sender'] == 'Hillary Clinton'][c]]

corder = [c for c in cols_needed]
cvals = [source] + [cols_needed[c] for c in corder]
corder = ['Source'] + corder
cvals = np.array(cvals).T
df3 = pd.DataFrame.from_records(cvals)
df3.columns = corder
for c in corder:
    if not c == 'Source':
        if c[-4:] == 'rate':
            df3[c] = df[c].astype(float).fillna(0.0)
        else:
            df3[c] = df[c].astype(int).fillna(0)

def train_or_test(s):
    return int(random.random()+0.5)
def from_email(s):
    if s == 'Hillary Clinton':
        return 1
    return 0
df3['trainingdata'] = df3['Source'].apply(train_or_test)
df3['Email'] = df3['Source'].apply(from_email)
training = df3[df3['trainingdata'] == 1]
test = df3[df3['trainingdata'] == 0]

labs = ['wordcount']
X = [[np.log(int(x)) for x in training['wordcount']]]
for e in emots:
    X.append([x for x in training[e + 'rate']])
    labs.append(e)
y = np.array([x for x in training['Email']])
labs.append('Email')
X = np.array(X).T

X2 = [[np.log(int(x)) for x in test['wordcount']]]
for e in emots:
    X2.append([x for x in test[e + 'rate']])
y2 = np.array([x for x in test['Email']])
X2 = np.array(X2).T

#clf = SVC()
clf = RandomForestClassifier()
#clf = LogisticRegression()
```

```
X_red = X[:, [0, 1, 4, 6]]
X_red2 = X2[:, [0, 1, 4, 6]]
clf.fit(X, y)

p = [x for x in clf.predict(X)]
p2 = [x for x in clf.predict(X2)]

tp = 0
tn = 0
fp = 0
fn = 0
for i in range(len(y)):
    if p[i] >= 0.5:
        if y[i] == 1:
            tp += 1
        else:
            fp += 1
    else:
        if y[i] == 1:
            fn += 1
        else:
            tn += 1
print('Training')
print('True positives', tp, 100*tp/len([yi for yi in y if yi == 1]))
print('False positives', fp, 100*fp/len([yi for yi in y if yi == 0]))
print('True negatives', tn, 100*tn/len([yi for yi in y if yi == 0]))
print('False negatives', fn, 100*fn/len([yi for yi in y if yi == 1]))

tp = 0
tn = 0
fp = 0
fn = 0
for i in range(len(y2)):
    if p2[i] >= 0.5:
        if y2[i] == 1:
            tp += 1
        else:
            fp += 1
    else:
        if y2[i] == 1:
            fn += 1
        else:
            tn += 1
print('Test')
print('True positives', tp, 100*tp/len([yi for yi in y2 if yi == 1]))
print('False positives', fp, 100*fp/len([yi for yi in y2 if yi == 0]))
print('True negatives', tn, 100*tn/len([yi for yi in y2 if yi == 0]))
print('False negatives', fn, 100*fn/len([yi for yi in y2 if yi == 1]))
```

APPENDIX G.

Additional figures and tables.

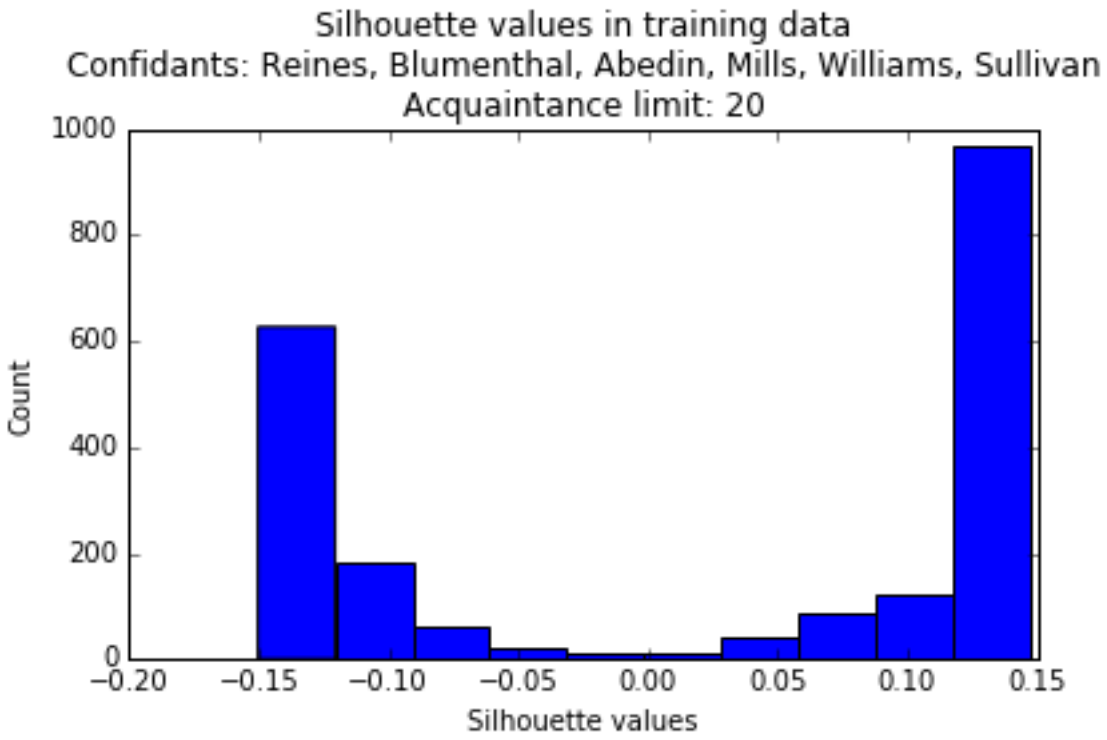

Figure 4. Silhouette values for the confidant/acquaintance clusters using a longer list of confidants. Values near 1 indicate similarity and values near -1 indicate dissimilarity.

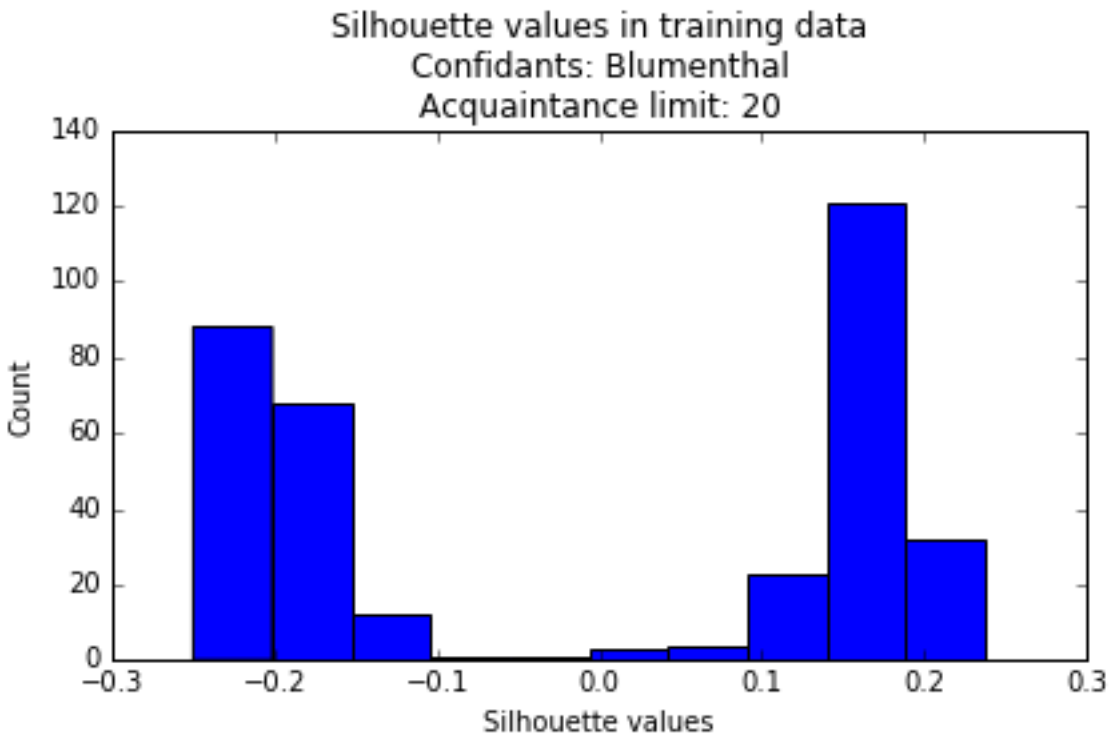

Figure 5. Silhouette values for the confidant/acquaintance clusters using a shorter list of confidants. Values near 1 indicate similarity and values near -1 indicate dissimilarity. Even using just one person as a confidant, the confidant email group is not too similar to itself.