NAME: Will Arliss
HW06/HW07 — STAT/CS 287
DATE: March, 2020

H4.P1.1

I stored the data using a dictionary with 'tweet IDs' as keys and a list of timestamp and text as values. I chose to store the data in this way because I saw that some timestamps on the tweets are identical. Because dictionaries can't hold duplicate keys, I was losing roughly 30,000 tweets from the original json files. Using sequential IDs to reference a list of timestamp and text eliminates this issue. I stored the data by reading in lines from the unzipped JSON file and using error handling to find tweets that needed repairs before being decoded.

H4.P1.2

From the twitter data file, 58575 lines were able to be accepted without repairs necessary. 18000 lines needed to be repaired before they could be accepted. The repairs can be broken down as follows: 10415 json lines were missing a leading curly bracket ("{") and 7585 lines were missing a closing curly bracket ("}"). After repairs, there are 76575 records present in the data.

H4.P1.3

I stored the twitter data in a dictionary. The keys in the dictionary are formatted as ID numbers for each tweet (applied in order that they are decoded from 1 to 76575) and the values are formatted as lists. The value lists contain one datetime entry and one nested list containing the text from each respective tweet. The datetime entries are formatted as a datetime.datetime (year, month, day, hour, minute, second). The nested list is the tweet itself; the tweet is stripped of all special characters, formatted to lowercase, and each word is an individual element of the list. See HW04_warliss.py, lines 11-76.

H4.P1.4

I sorted the tweets into Obama and Romney lists by searching each individual word of a tweet for the substrings 'barack' and 'obama' as well as 'mitt' and 'romney'. If the substring was found, that tweet was added to the respective list. This was repeated on every tweet. See HW04_warliss.py, lines 80-112.

H4.P2

See HW04_warliss.py, lines 116-157 for relevant code. See Figure 1: Tweets per Hour About Romney and Obama for the graph produced. There is visible a large drop off and successive rise in tweets about Obama in the hours of 8:00 PM and 9:00 PM on November 6th.

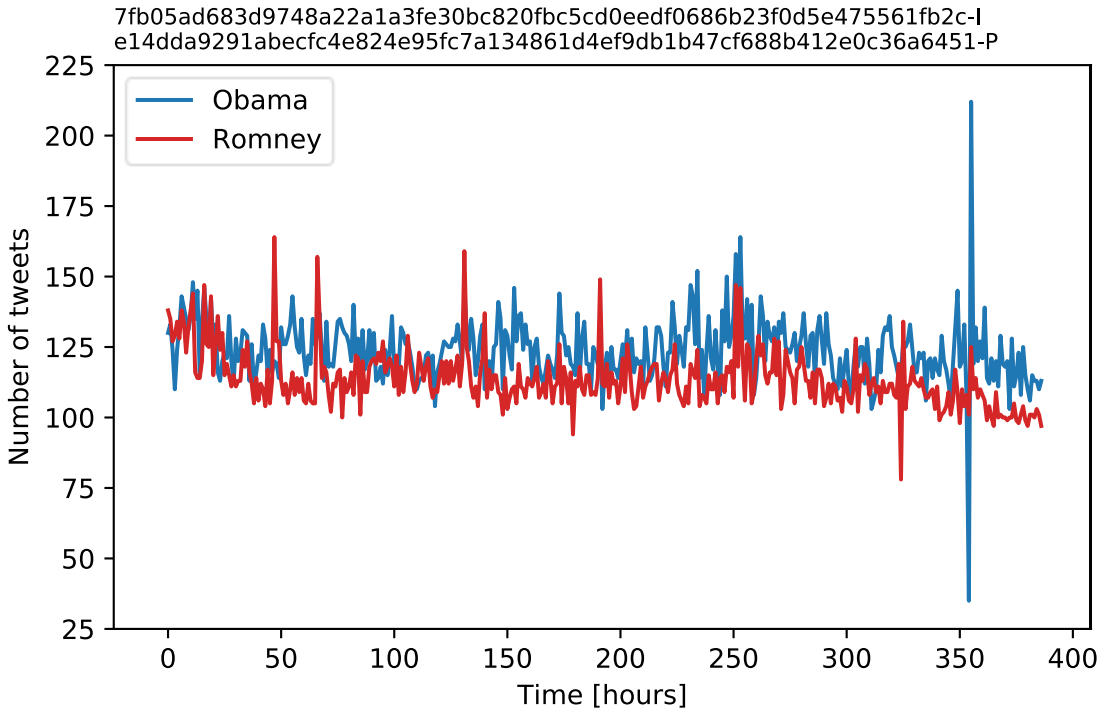

Figure 1: Tweets per Hour About Romney and Obama

H4.P3.1

One problem associated with the choice of this coefficient is the occurrence of misspelled words. This is a problem with practically any keyword analysis method. The occurrence of a misspelled word will cause frequency statistics to be misrepresentative of the true data. Also, this coefficient only reports on where the keywords appear, it does not provide information on context or implications of the word. I unaware of any alternative methods of dividing keywords between two corpuses.

H4.P3.2

See Table 1: Yule Coefficients for Obama and Romney Twitter Words. A word's relative position on the list implies that it is most heavily used in tweets relating to its respective column. (Maybe: if the word appears in a tweet, the absolute value of its Yule coefficient is the likelihood that the tweet it appears in is about the word's respective column – i.e. if the word 'ellenpage' appears in a tweet, there's a 99.14% chance that the tweet is a Romney tweet.) See HW04_warliss, lines 161-226.

Table 1: Yule Coefficients for Obama and Romney Twitter Words

Obama Words	Yule Coef	Romney Words	Yule Coef
tix	0.98485	ellenpage	-0.99138
confirm	0.97590	reflect	-0.98571

busabusss	0.97403	freed	-0.98374
z	0.97333	sethmacfarlane	-0.98165
submit	0.97333	sashay	-0.97895
katyperry	0.97143	sandyshurricane	-0.97661
expertise	0.96970	godspeed	-0.97333
officials	0.96610	considering	-0.97143
magicjohnson	0.96296	remembered	-0.96923
bbcbreaking	0.96296	90999	-0.96757
jay	0.96117	defiance	-0.96610
instructions	0.95876	puto	-0.95455
animator	0.95833	futures	-0.95455
seaboard	0.95745	dictated	-0.95312
forall	0.95506	seize	-0.95238
privatesector	0.95455	47percent	-0.95122
steven	0.95455	occasion	-0.95122
lenadunham	0.95402	shoq	-0.95122
spy	0.95349	closet	-0.95000
orleans	0.95238	bigot	-0.94702
bruce	0.95238	gym	-0.94595
huracÃ¡n	0.95122	vagina	-0.94444
performing	0.95000	pretended	-0.94444
champion	0.94872	metaquest	-0.94444
madonnas	0.94737	bowl	-0.94444
ofaco	0.94595	georgelopez	-0.94118
singlehandedly	0.94521	rcmahoney	-0.93548
ofava	0.94203	flashback	-0.93443
chant	0.94030	measured	-0.93333
winds	0.93939	ncgop	-0.93258
maya	0.93750	electorals	-0.93103
gottavote	0.93617	jconason	-0.93103
highlights	0.93548	beltwaybaca	-0.92857
ohvotesearly	0.93548	latinas	-0.92857
proudofobama	0.93352	amphitheater	-0.92593
eric	0.93333	dictators	-0.92481
lays	0.93333	overflow	-0.92308
accomplishment	0.93220	thread	-0.92308
phyrefyter	0.93103	expand	-0.92000
ofanh	0.93103	evanaxelbank	-0.91667
jm	0.92857	rallys	-0.91667
commitment	0.92857	attended	-0.91667
tammybaldwinwi	0.92593	1100	-0.91489

firefighter	0.92593	zacharyquinto	-0.91367
ofaia	0.92593	jljacobson	-0.91304
craigatfema	0.92593	taxreturns	-0.91304
chapter	0.92308	qualities	-0.91250
madison	0.92308	fresh	-0.91111
greeting	0.92000	10moredays	-0.90909
location	0.91980	kissimmee	-0.90909
yourfavwhiteguy	0.91837	marnus3	-0.90909
vota	0.91667	mrburlesk	-0.90476
teaching	0.91667	hilary	-0.90476
peter	0.91667	prestoncnn	-0.90476
shortgo	0.91667	evader	-0.90476
formed	0.91667	solve	-0.90323
fights	0.91429	implying	-0.90244
dadt	0.91304	releasethereturns	-0.90000
campus	0.91304	fairfax	-0.90000
ofanc	0.91304	hurcanesandy	-0.90000
destiny	0.91045	uniforms	-0.90000
israelfor	0.90909	infomercial	-0.90000
ucf	0.90909	blasted	-0.90000
theellenshow	0.90909	otoolefan	-0.90000
booing	0.90698	dgjackson	-0.89796
begun	0.90678	robportman	-0.89744
tony	0.90476	manchester	-0.89474
row	0.90244	badgering	-0.89474
safer	0.90164	presidentelect	-0.89474
libertylynx	0.90000	tommyforwi	-0.89474
theblazetv	0.90000	blackrepublican	-0.89474
toddkincannon	0.90000	realstaceydash	-0.89388
aceofspadeshq	0.90000	conviction	-0.89286
eastern	0.89796	p0tus	-0.89068
paulmccartney	0.89744	morrowchris	-0.88991
themick1962	0.89744	wrangler	-0.88889
ofanv	0.89610	congressman	-0.88889
ofaoh	0.89474	scumbag	-0.88889
recruit	0.89474	lavellsays	-0.88889
latinosforobama	0.89474	i4	-0.88889
commend	0.89474	dirkz1	-0.88889
therealmix	0.89474	romneylies	-0.88571
citing	0.89362	believeinamerica	-0.88571
fri	0.89333	tedcruz	-0.88571

basketball	0.89286	azmoderate	-0.88571
54	0.89130	rooting	-0.88298
16000	0.88889	tom	-0.88235
tpbgirl	0.88889	1118	-0.88235
🙏	0.88889	wut	-0.88235
strikes	0.88889	widely	-0.88235
reservations	0.88889	kathrynjscags	-0.88235
brave	0.88889	rickgorka	-0.88095
lesbian	0.88889	sayfie	-0.87629
❤	0.88889	ethics	-0.87500
tjmshow	0.88889	chelered	-0.87500
nvdecides	0.88679	devinmillington	-0.875
voucher	0.88406	etchasketch	-0.875
matthew	0.88235	intention	-0.875
rramirez44	0.88235	kasinca	-0.875
ruling	0.88235	anndromneys	-0.875

H4.P3.3

The words found by the yule coefficients make sense for the most part. There is a large number of celebrity names (Ellen Page, Magic Johnson, Katy Perry). At first this was surprising to me, but as I thought more about the nature of twitter, I realized that when celebrities make certain tweets, they might be retweeted many times. Every occurrence of a retweet results in their name being tweeted again. It also makes sense that certain phrases appear in a particular column. For example, it is logical that 'latinsforobama' would appear in the Obama column. One thing that remains confusing to me is why certain numbers (1100, 90999, 16000) keep appearing in the analysis. What were the numbers in reference to and why are some more heavily used in Obama tweets than Romney tweets (and vice versa)?

H6.P1.1

The output of the polarity_scores function is a dictionary of four different key-value pairs. Each value consists of a number between -1 and 1. The keys are "neg", "neu", "pos", and "compound". "Neg" represents the proportion of words in the string that imply negative sentiment, according to their polarity score ranging from -4 to 4. "Neu" represents the proportion of words that imply neutral sentiment or none at all and "pos" represents the proportion that imply positive sentiment. "Compound" represents the sum of the polarity scores for each word in the string, normalized between -1 and 1. For the compound score (as well as the polarity scores), a negative score indicates negative sentiment and a positive score indicates positive sentiment. A score of 0 represents neutrality. Polarity scores are predetermined according to a lexicon dictionary of 7502 different tokens.

H6.P1.3

One drawback/criticism of VADER is that it has a difficult time processing negations. If a sentence contains many positively scored words, it will likely have a high compound score. This is a problem because if a negation such as “not”, “doesn’t”, or “isn’t” precedes those positive words, VADER may not read all those words as their inverse. If a sentence contains words like “good”, “happy”, or “excited”, the inclusion of the word “not” before those words will negate them. VADER may recognize “not good” as negative, but the rest of the words will still be read positively as “happy” and “excited” even though the “not” applies to them as well. An expression similar to this that VADER might have a hard time interpreting is “not too bad”. The term bad has an obvious negative sentiment. However, the full expression “not too bad” actually carries positive sentiment in common vernacular. It might be a good idea to include common expressions as well as individual words and characters in the VADER lexicon to avoid confusions like this. Another drawback of VADER is that it doesn’t always deal with clauses properly. While there are some key clause indicators that VADER is programmed to recognize, it is not perfect. This is a problem because sentiment can change across clauses. A sentence like, “I am excited, but not very happy”, has a positive clause in the beginning and a negative clause at the end. The positive sentiment in the first clause should be lessened by the second clause, but VADER may not always do this. Including clauses greatly increases the complexity of a sentence. Analyzing individual words of a sentence can be very misleading. The way English vernacular works makes it possible for many individual words to have positive connotations on their own, but a negative connotation when strung together in a specific way.

H6.P1.4

VADER’s lexicon contains 7502 tokens total. See HW06_warliss.py lines 299-310.

H6.P2.1

The filtering functions are located in HW06_warliss.py lines 98-158. They are applied to the code in lines 314-342. The keywords() function is an exact copy from HW04. The twitter data is saved in a different (more efficient) format in HW06 than in HW04, so some formatting was necessary before passing the HW06 data into the HW04 function. This can be found in lines 332-338. Most of the differences in the results in the word cleaning. The word cleaning function used in HW06 can be found in lines 65-94. This function does not remove all special/punctuation characters as the function in HW04 does. Instead, it looks at each individual word of the tweet and compares it to a premade list of emoticon characters taken from the VADER lexicon (see lines 285-291). In this way, emoticon tokens as well as exclamation points are preserved in the data. In HW04, all special characters are removed from the data. Additionally, the HW06 word cleaning functions tests if a word is entirely uppercase. words that are entirely uppercase are preserved as they appear. Words that are merely capitalized or have a capital letter within it are converted to lowercase. So, in HW06 the tweets are formatted as a string almost all lowercase, with the exception of fully capitalized words, and contain emoticon characters and exclamation points. The

HW04 tweets are formatted as a list of upper and lowercase letters and includes no special characters or punctuation.

The filtering functions of HW06 are much simpler than that of HW04. The functions temporarily convert the tweet string to all lowercase and look for the word “obama" or “romney” then adds them to the proper list accordingly. The HW04 function looks at each individual word in the tweet (made lowercase) and changes a state variable to indicate if a word does or does not contain “obama" or “romney”. This is done through a series of for loops, if statements, and break statements.

H6.P2.2

See Table 2: Corpus Contingency. Cell Obama(C1)-Obama(C2) shows the number of tweets in the HW04 Obama corpus that are in the HW06 Obama Corpus. Cell Obama(C1)-Romney(C2) shows the number of tweets in the HW04 Obama corpus that are in the HW06 Romney corpus. Cell Romney(C1)-Obama(C2) shows the number of tweets in the HW04 Romney corpus that are in the HW06 Obama corpus. Cell Romney(C1)-Romney(C2) shows the number of tweets in the HW04 Romney Corpus that are in the HW06 Romney corpus. Most tweets stayed in the same corpus between HW04 and HW06. See HW06_warliss.py lines 345-361 for output code.

Table 2: Corpus Contingency

	Obama (C2)	Romney (C2)
Obama (C1)	47249	15218
Romney (C1)	15218	43484

H6.P2.3

The tidy file is structured as a .csv file with 7 columns in it and 76,576 rows. The first row is text defining the column names. The column names are ID – an arbitrary unique number assigned to each tweet to facilitate sorting processes, Datetime – the date and time that the tweet was executed, Corpus – “O” or “R” depending on if the tweet is about Obama or Romney, Negative – the proportion of the words in the tweet that have a negative score, Neutral – the proportion of the words in the tweet that have a neutral score, Positive – the proportion of words in the tweet that have a positive score, and Compound – the compound VADER polarity score. I chose not to include text in the file because I figured that most of the information needed from the text of the tweets is already captured by the four sentiment scores and because only including an ID number instead of text saves space. I included corpus membership to avoid having to re-sort the tweets after loading them back in. The tweets will need to be in two corpuses for analysis.

H6.P3

The mean compound score of tweets in the Obama corpus is 0.099 and the mean score in the Romney corpus is 0.105. The average compound score of all tweets is 0.101. Romney's mean score is 0.06 points greater than Obama's. This is surprising considering the outcome of the election. Within the Obama corpus, 41.6% of the tweets have a positive compound score and 24.5% of the tweets have a negative compound score. Within the Romney corpus, 46.3% of the tweets have a positive compound score and 26.5% of the tweets have a negative compound score. (The remainders here are neutral scores). The mean proportion of negative words in the Obama corpus is 0.11 and the mean proportion in the Romney corpus is 0.12. The mean proportion of neutral words in the Obama corpus is 0.829 and the mean proportion in the Romney corpus is .8. The mean proportion of positive words in the Obama corpus is 0.062 and the mean proportion in the Romney corpus is 0.08. It is not surprising that the statistics for negative, neutral, and positive scores are so similar because Twitter is generally hostile on both sides of the political spectrum. The fact that the proportions are so similar yet the mean compound score for Romney is higher means that the intensity of the positive words were stronger in the Romney corpus or the intensity of negative words were stronger in the Obama Corpus. See Figure 2: Obama Compound Scores Histogram for a distribution of compound scores for the Obama corpus. There appear to be more positive scores (corroborated by the 41.6% statistic reported above) than negative scores and there is a large spike right around 0. See Figure 3: Romney Compound Scores Histogram for a distribution of compound scores for the Romney corpus. There appear to be more positive scores than negative scores (corroborated by the 46.3% statistic reported above) than negative scores and there is again a large spike around 0.

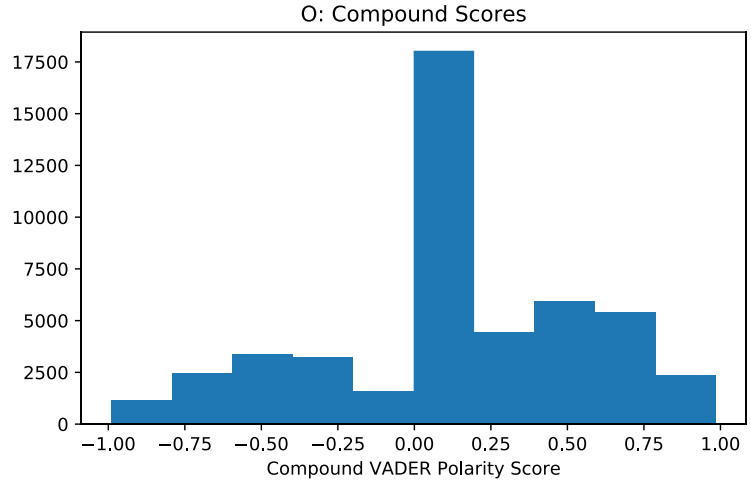

Figure 2: Obama Compound Scores Histogram

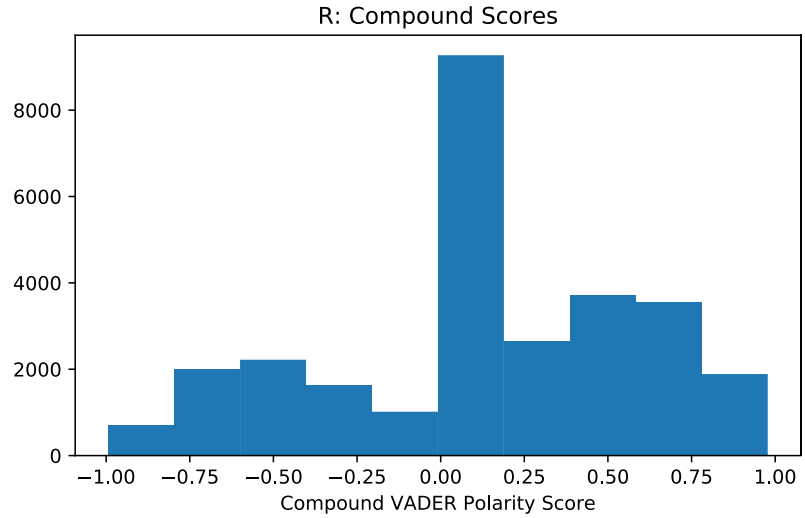

Figure 3: Romney Compound Scores Histogram

H6.P4.1

Figure 4: Twitter Sentiment - Obama vs Romney Figure 5: Twitter Sentiment - Total Election shows the change over time of average twitter sentiment based on VADER polarity scores between Obama and Romney. Figure 4: Twitter Sentiment - Obama vs Romney Figure 5: Twitter Sentiment - Total Election shows the change over time of average twitter sentiment of the entire election. Figure 6: Obama Twitter Sentiment - HW04 vs HW06 Figure 7: Romney Twitter Sentiment - HW04 vs HW06 compares the change in twitter sentiment of Obama between the corpus from HW04 and HW06. Figure 6: Obama Twitter Sentiment - HW04 vs HW06 Figure 7: Romney Twitter Sentiment - HW04 vs HW06 compares the change in twitter sentiment of Romney between the corpus from HW04 and HW06. In Figures 3 and 4, the light blue represents average sentiment of the corpus from HW04, the light red represents average sentiment of the corpus from HW06, and the purple color shows the overlap.

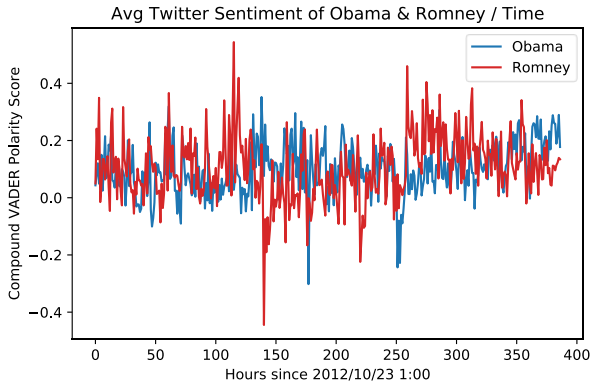

Figure 4: Twitter Sentiment - Obama vs Romney

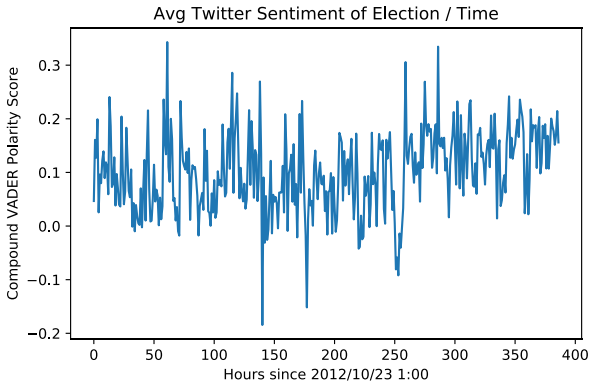

Figure 5: Twitter Sentiment - Total Election

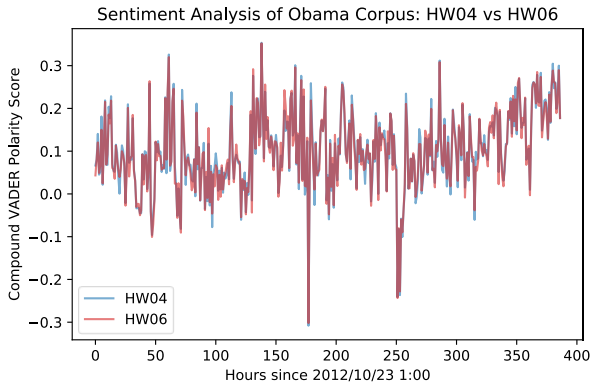

Figure 6: Obama Twitter Sentiment - HW04 vs HW06

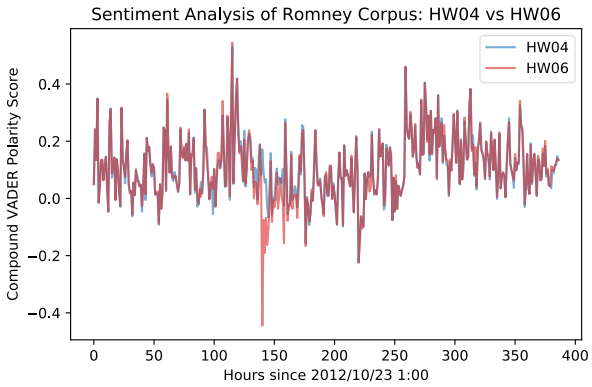

Figure 7: Romney Twitter Sentiment - HW04 vs HW06

H6.P4.2

Interpret your time series plots (300+ words). What do you learn from these visualizations? Is there a consistent change in sentiment over time? Why or why not?

Figure 4: Twitter Sentiment - Obama vs Romney Figure 5: Twitter Sentiment - Total Election shows Twitter sentiment toward the two candidates starting out fairly even. There are several fluctuations where the candidates switch position (top or bottom). Obama finishes well above Romney by the end, which makes sense considering the outcome of the election. It is interesting that the fluctuation in Twitter sentiment toward Romney is so much more volatile than that towards Obama. There is a major drop in sentiment toward Romney about 6 days into the data. Figure 4: Twitter Sentiment - Obama vs Romney Figure 5: Twitter Sentiment - Total Election shows the mean compound score of all tweets (not divided by corpus). I chose to include this to indicate general twitter sentiment toward the election as a whole. The Sentiment seems relatively consistent around 0.1 (positive) with a slight drop in the middle and a slight rise at the end. Perhaps there were more people happy with the election's outcome than there were people unhappy with the outcome; or perhaps people were happy for the election decision to finally be made. Figure 6: Obama Twitter Sentiment - HW04 vs HW06 Figure 7: Romney Twitter Sentiment - HW04 vs HW06 compares the Twitter sentiment assessed by the Obama corpuses from homework 4 versus homework 6. The trends have been made transparent to allow better comparison of the two. The two trends appear to be consistent with one another. The two drops in sentiment toward Obama do not appear as dramatic in this representation. Figure 6: Obama Twitter Sentiment - HW04 vs HW06 Figure 7: Romney Twitter Sentiment - HW04 vs HW06 compares the Twitter sentiment assessed by the Romney corpuses from homework 4 versus homework 6. It is interesting that the corpus from homework 4 does not show such a dramatic drop (or any drop at all) at around time $\cong 175$. Aside from that feature, the trends are again relatively consistent with one another. I found it interesting that the trends of all time series plots had their volatility centered around (more or less) 0.1. I would have expected more of a swing.