Project Ultra

Proof of Concept: Semi-Unsupervised Sentiment Analysis

Abstract

This paper presents a study of sentiment analysis on Twitter messages. Several linear and non-linear methods for sentiment analysis are introduced, all of which are designed to operate in a *semi-unsupervised* way. Furthermore, these methods are capable of analyzing Twitter text messages in real-time, since they only require a very small set of manually generated training data. These novel methods are based on the *key observation* that information carried by tweets is either centralized around or in parallel with the hash tags and the *key assumption* that hash tags with clear sentiments are primarily used accordingly.

Fan Yu
University of Michigan
Email: fanyuchn@umich.edu
November, 2016

1 Project Introduction

In this work, I developed *prototype methods of "semi-unsupervised"*[1] *sentiment analysis on social media text messages.* These methods are designed to solve the conflicting situation where: on one hand, social media related applications are time-sensitive; on the other hand, traditional sentiment analysis algorithms require large training data set which would take a long time to generate manually. Thus, *it is important to develop a sentiment analysis method with a good enough accuracy and minimal dependence on the size of training data.*

The case chosen to study is the 2016 presidential election of the United States, because this is certainly a major media and public focus point. Twitter is chosen as the data source, primarily due to its free Twitter API. Section.3 demonstrates some interesting preliminary results. Section.4 elaborates on the method of "semi-unsupervised" sentiment analysis. Section.6 provides information regarding to the data set chosen for this project. Section.7 outlines how this project (the codes) is structured. The Github link of this project could be found in Ref.[8].

2 Project Design

This project is currently divided into three phases.

2.1 Phase 1: Data Cleaning

Phase 1 of this project is devoted to data cleaning. Since we have the 2016 election as a clear focus point, and that the Twitter API was setup with only a geo-location filter[2], only a small part of the raw data would contain information needed for our study. *In order to extract these relevant data, I adopted a dynamic method for data cleaning.* Specifically, during the chronological scan through the data set, one will build up and update constantly dictionaries of hash tags and Twitter users that are more closely involved with the 2016 election[3]. At the same time, we are using these dictionaries to extract relevant information.

The key feature is that these dictionaries are dynamically adjusted. For example, #job was overall the most frequently used hash tag, and it *was* indeed related to the 2016 election. However, since it was so widely used in other contexts that were irrelevant to the 2016 election, this #job would be excluded from our dictionary. By doing so, we would *not* loss too much tweets concerning both job market and the 2016 election, because such tweets would most likely call other hash tags that were exclusively related to our topic under study. But without excluding #job, we would fetch up so many tweets that were irrelevant, complicating our analysis. Please note that our approach is quite different from Ref.[1] where

[1] The "semi-unsupervised" is compared with the traditional NLP methods which require *manually* generated training corpus. My method attempts to perform sentiment analysis in a more or less unsupervised fashion, with as little manual input as possible, by exploiting Twitter users' behavior pattern.

[2] Tweets collected were those with geo-location enabled which indicated its origin from the continental U.S..

[3] Hash tags and Twitter users have different filtering standards.

the set of rules for filtering data was fixed. My approach would adjust dynamically to new hash tags and new hot topics that might arise during a campaign.

2.2 Phase 2: Sentiment Analysis

Phase 2 of this project is devoted to sentiment analysis. During my previous attempts at this subject, I realized that models with only statistical variables, though powerful, is not very effective when it comes to figuring out individual's opinion, nor when it comes to understand what kinds of information specific individual is receiving. If we want to determine whether a specific user holds a positive or negative opinion toward a candidate, it won't be enough to simply check "whether this user called this candidate's name in his message or not". Instead, one has to analyze "what this user said about this candidate in his or her message". *In short, sentiment analysis of tweet text message is absolutely necessary.*

A thorough study regarding sentiments carried by tweet messages is conducted and several methods for performing sentiment analysis are proposed. To summarize the major conclusions:

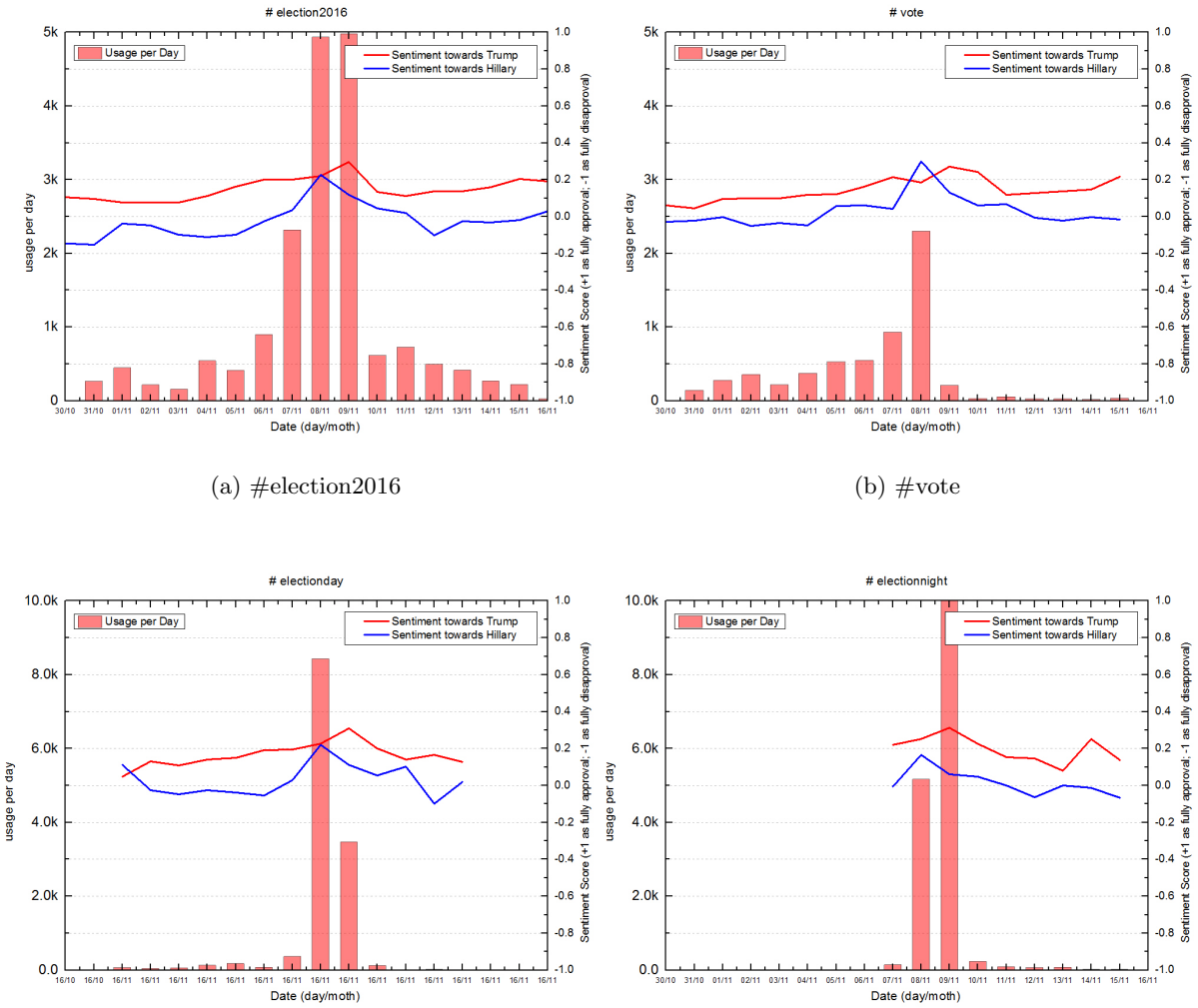
- The key observation that information and sentiments expressed by Twitter users are usually centralized around hash tags. See Section.4.
- Methods for estimating sentiments carried by hash tags are proposed. See Section.4.1 for the general idea and an iteration method, Section.4.2 for the linear method and Section.4.3 for the nonlinear method.
- Estimation of sentiments carried by selected hash tags using the 2016 election data set offers a preliminary proof that the key observation, the key assumption and my methods are correct. See Section.3.1.
- Estimation of sentiments of tweet text messages is performed using hash tags only. Accuracy of such estimation is manually verified as at least 75%. See Section.4.4.
- Arguments are made regarding "why traditional sentiment analysis method, given its nature as supervised learning and its reliance on manually generated training corpus, is inherently unsuitable for social media applications". See Section.4.
- A traditional sentiment analysis approach on tweet text messages is carried out using the Long Short Term Memory (LSTM) method and the "Semi-Automatically" generated training data set. See Section.5.

2.3 Phase 3: Statistical Results

Phase 3 of this project is built on Phase 1 to further explore the data set and generate training/predicting data sets for machine learning algorithms. One could also pull some interesting results that are purely statistical, like ranking of most popular hash tags, most active users, etc. However, it is those purely statistical results combined with Sentiment Analysis from Phase 2 that would lead to the most powerful and meaningful conclusions.

3 Results Demonstration

3.1 Sentiments of Selected Hash Tags



(a) #election2016

(b) #vote

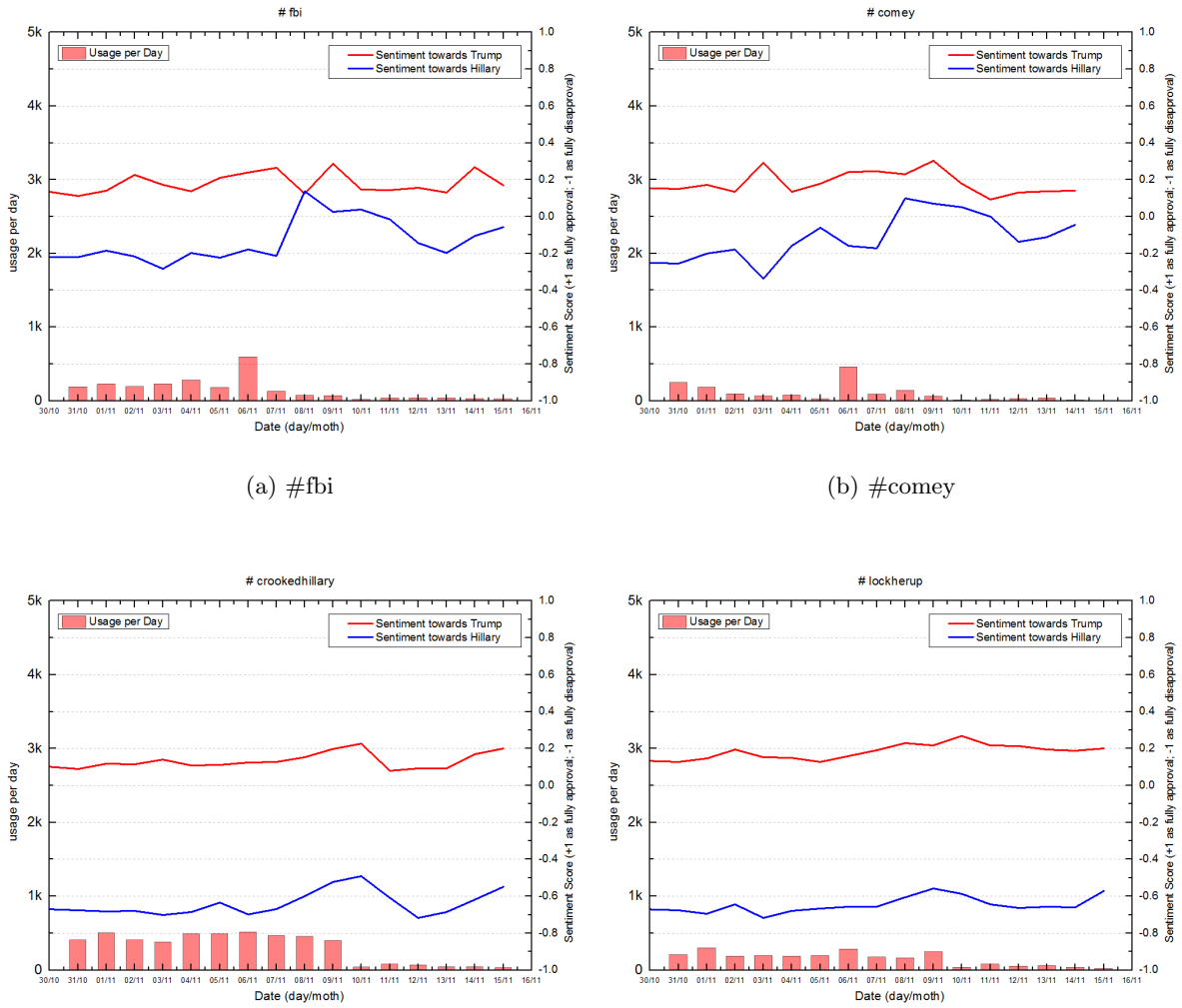
(c) #electionday

(d) #electionnight

Figure 1: Sentiment Scores of Hash Tags related to Voting and Election

To begin with, let's look at hash tags that were directly related to "voting", during the two weeks before and after the Nov 8th election day. Shown in Fig.1 below are the four hash tags #election2016, #vote, #electionday and #electionnight. One could observe that their daily usage peaks around Nov 8th[4]. Nonetheless, there are differences: #election2016 was the more generally used hash tag; #vote saw increasing usage in the leading up to Nov 8th, because Twitter users were urging their friends to cast their votes on the election day; #electionday's and #electionnight's usage were more localized around

[4]As for why those hash tags only recorded a few thousands calls, even on the election day, please check the Section.6 below



(a) #fbi

(b) #comey

(c) #crookedhillary

(d) #lockherup

Figure 2: Sentiment Scores of Hash Tags related to FBI's latest investigation of candidate Hillary Clinton

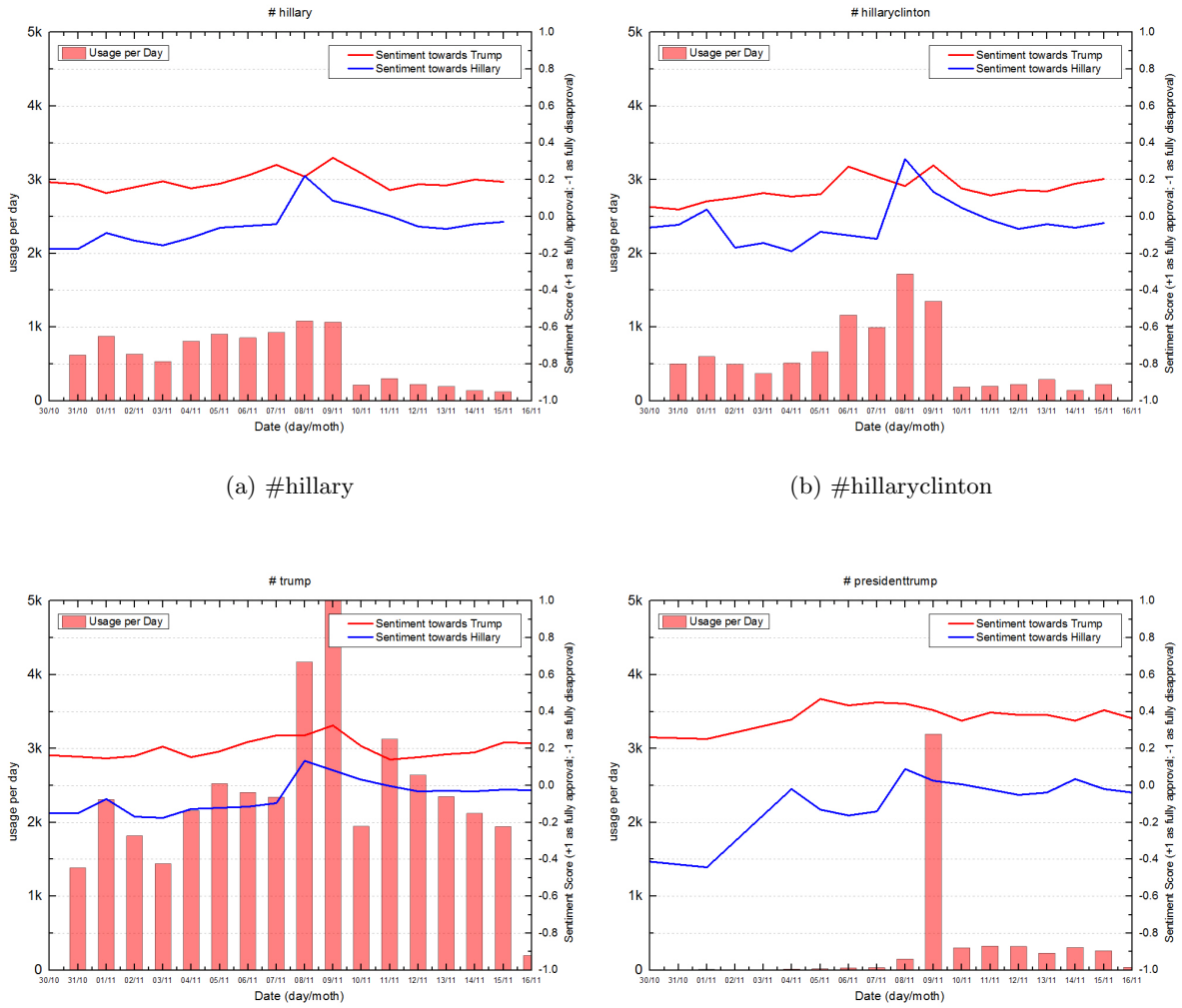
Nov 8 9th[5]. Furthermore, all those neutral hash tags demonstrated a strong correlation in their sentiment scores[6]. Overall, these four hash tags demonstrated a more positive sentiment score towards candidate Donald Trump; and sentiments towards both candidates saw an increasing trend during the last few days before Nov 8th[7]. However, the surprising observation is how quickly the sentiment towards candidate Hillary Clinton dropped after Nov 8th[8], given the fact that candidate Hillary Clinton's supporters should be seriously disappointed by the result, thus more incentives to "say something" on the Internet.

[5]Please note that #electionnight recorded more than 10,000 usage on Nov 9th, reflecting people waking up to the unexpected election result.

[6]The sentiment score towards each candidate is in the range between -1 and 1, with -1 indicating complete negative sentiment and 1 indicating complete positive sentiment. For each hash tag, a pair of sentiment scores are assigned or calculated. One could *roughly* treat the numerical sentiment score as probability of a hash tags expressing positive or negative sentiment towards specific candidate. The actual meaning of this numerical sentiment score will be discussed in Section.4.

[7]This might due to the fact that supporters on both sides were rallying for their candidates.

[8]Equally surprising is the sharp increase around Nov 6 8th.



(a) #hillary

(b) #hillaryclinton

(c) #trump

(d) #presidenttrump

Figure 3: Sentiment Scores of Hash Tags related to candidates themselves

Secondly, let's look at hash tags related to FBI's latest investigation of candidate Hillary Clinton's email affairs. The strong correlation between sentiment scores of #fbi and #comey is straight forward to understand, as these two hash tags were the most used when Twitter users were discussing the investigation. The sharp decrease and increase of sentiment scores around Nov 3rd and Nov 7th corresponded to the reopening and closing of this latest investigation respectively. However, the improved sentiment scores on Nov 8th may also due to the fact that more supporters of candidate Hillary Clinton became active on the election day. Furthermore, it is interesting that the daily usage of #crookedhillary and #lockherup decreased sharply post Nov 9th. This raises the question: Do candidate Donald Trump's supporters sincerely want to have an investigation of candidate Hillary Clinton's *potential* wrong doings? Or, Are these hash tags merely campaign slogans? If #crookedhillary and #lockherup were mostly used by *real* Twitter users, then it would be difficult to accept the second assumption. However, if #crookedhillary and #lockherup were mostly used by *some* Twitter users as campaign slogans, then we

might have a more *sinister* scenario[9].

Last but not least, let's look at hash tags related directly to both candidates. For #hillary and #hillaryclintion, one could clearly observe a negative correlation between the two sentiment scores[10] up until Nov 9th. It would be easy to interpret this negative correlation since the 2016 election is well known as the most divisive one in recent history. But statistically, this is also due to the fact that #hillary and #hillaryclintion are "neutral" hash tags, meaning it could be used with hash tags like #imwithher as well as hash tags like #lockherup[11][12]. Thus, the trend of sentiment scores of these "neutral" hash tags were controlled by the more active side on Twitter. With this understanding, we could conclude that #hillary and #hillaryclintion were slightly dominated by candidate Donald Trump's supporters, until ∼Nov 7th. Once again, I want to point out the usage of #hillary and #hillaryclintion dropped sharply post Nov 9th while the usage of #trump persisted.

4 Ideas for Semi-Automatic Sentiment Analysis

In the context of social media, sentiment analysis occupies a central place when one tries to understand what is going on, because it is not enough to know "who talked with whom", rather, one needs to know "who talked *what* with whom". The traditional sentiment analysis method is a supervised learning, with the training corpus usually generated manually. What I intend to argue here, is that manually generating a training corpus for NLP applications on social media is neither ideal nor feasible. Instead, one should exploit Twitter users' unique expressing habits and adopt a method resemble that of unsupervised learning. *Please note that in this project, we would only work on tweet messages that come with one or more hash tags. Tweet messages without any hash tag would not be studied at this point.*

To begin with, let's consider the "golden standard" training corpus for a sentiment analysis application, which is usually generated by human beings. Of course, manually generated corpus takes human resources and money. *But the most critical short coming of such a "golden standard" corpus is that it won't be able to keep up with the flowing, dynamic context of social media.* As shown by previous analyses, people's sentiments and most discussed topics change from day to day, sometimes rapidly. However, most social media applications are intended for advertising, promoting and predicting. Thus, it is important to *develop method for rapidly generating a good enough training corpus roughly in real-time speed.* One that will be able to keep up with the shifting context on a social media.

[9] I have found many "Twitter bots" during my analyses. For example, Twitter users who routinely post dozens of tweets per hour, which is not realistic for a real human being. Once I finish Phase 4 of this project, I would be able to supplement my arguments with data.

[10] Red for candidate Donald Trump, blue for candidate Hillary Clinton.

[11] Please note that this same negative correlation was also observed for other "neutral hash tags" like #fbi and #comey in Fig.2.

[12] Please note that there is no observable negative correlation between sentiment scores of #trump, because this hash tags was overwhelmingly used by candidate Donald Trump's supporters. It may also due to the fact that there were so many discussions on Twitter about candidate Donald Trump himself, without mentioning candidate Hillary Clinton nor calling hash tags related to her.

To develop such a method, one need to fully exploit Twitter users' expressing habits. For example, text messages on social media are usually very short[13]. Particularly for the case of Twitter, where one could not lay down a few words, people's messages are very succinct. This means people are making references to keywords and hash tags of which the meaning and sentiment are well known among Twitter users. This, however, represents a unique opportunity. Since Twitter users' messages are very succinct, most of what they want to express are allocated to the few keywords and hash tags. An example is in order to illustrate this point. This message *"for you non-stop! God Bless you for what youve endured these past 510 days, fighting for America #Salute #MAGA"* was twitted out early in the night of Nov 8th, and it called two hash tags #Salute and #MAGA. Without #MAGA, one could *only* tell that this user expressed good wishes for his or her candidate. It is with #MAGA that we are certain this user was a supporter for candidate Donald Trump. This demonstrates to what degree information could be centralized in a tweet, even when this tweet is already pretty long.

Consequently, *the need to figure out sentiments carried by keywords and hash tags is more important than the need to analyze structures and word frequencies of tweet text messages.* Thus, in the following subsections, I will focus on how to estimate sentiments carried by hash tags. Section.4.1 elaborates on challenges one will face and introduces the first method of estimating sentiments carried by hash tags through an iteration method. Section.4.2 proposes a linear method as an improvement over Section.4.1. Section.4.3 provides an additional nonlinear method using a single layer neural network.

4.1 Estimating Sentiments Carried by Hash Tags: Introduction

An important observation is that tweet messages usually come with more than one hash tags[14]. Thus in order to estimate the sentiment of a tweet, the *simplest* method is to manually mark the sentiment of those most frequently used hash tags, then use these manually marked hash tags to get a rough estimation of the opinions expressed by tweet messages. In this way, we are laboring over hash tags, emojis or other tokens rather than the text messages. *Here we made two key assumptions:*

1. sentiment of specific hash tag could be determined accurately, like #vote4hillary.
2. for most of the time, hash tags with clear sentiment bias are used according to their biases; in other words, expressing methods like satire, irony and exaggeration are rarely used.

However, if sticking to these assumptions directly, one would encounter two difficulties:

1. of those highly used hash tags, most are *neutral*, like #trump and #hillary. These hash tags are not necessarily used to express support or dislike. Thus one could not assign a sentiment to them without studying exactly how they were used. See Fig.4.

[13]For my data set, among tweets that called election-related hash tags, only half would have more than 10 words that are neither other users' names nor web-page links.

[14]At current stage of this project, I only study tweets with at least one hash tag that was related to the 2016 election.

Nov 8th		Nov 9th	
hash tag	**total call**	**hash tag**	**total call**
electionday	8431	electionnight	15873
electionnight	5168	trump	8620
imwithher	5061	election2016	4981
election2016	4936	maga	3948
trump	4173	electionday	3473
maga	2964	presidenttrump	3194
vote	2303	notmypresident	2669
trumppence16	2279	trumptrain	2311
imvotingbecause	2102	elections2016	1894
hillaryclinton	1719	makeamericagreatagain	1813
trumptrain	1451	trumppence16	1645
nevertrump	1355	donaldtrump	1528
draintheswamp	1094	trump2016	1480
makeamericagreatagain	1093	imwithher	1356
hillary	1082	hillaryclinton	1352
myvote2016	1050	hillary	1066
ivoted	1027	electionresults	945

Figure 4: Hash tags that were most used on Nov 8th and Nov 9th. Except those hash tags marked in red, the others should be treated at neutral hash tags.

2. among hash tags that are clearly biased towards certain sentiments, like #vote4hillary or #hillaryislier, more of them express negative rather than positive sentiments. As a results, the corpus one could get in this way[15] is heavily biased towards the negative sentiment.

To solve these difficulties, I developed a *method for dynamically expanding the set of hash tags with sentiment scores*. The idea is that, we could use the ∼200 manually marked hash tags as a starting point. Using an iteration method to "spread out" the sentiments: through each iteration, neutral hash tags will have their initially neutral sentiment scores adjusted, according to their correlation with manually marked ones. At the same time, manually marked hash tags will also have their sentiment scores adjusted according to their correlation with neutral ones. *Summarizing the Dynamic Iteration Method:*

1. Manually mark 200 hash tags[16] that carry very clear sentiments.

2. Since people usually call multiple hash tags together, thus it would be possible to build up a "networked" dictionary for each hash tag, recording: what other hash tags were used with, what was the usage of these "networked" hash tags.

3. Estimate each hash tag's correlation with other hash tags. Updating this hash tag's sentiment score according to scores of its "networked" hash tags and the correlations.

[15] Using the simple approach of deciding each tweet message's sentiment according to the manually marked ones.

[16] The ∼200 manually marked hash tags could be found here: https://github.com/Nimburg/Ultra_Phase1v5_Phase2v2_Phase3v1/tree/master/Ultra_Phase3v1/Data.

4. when updating sentiment scores of hash tags, the maximum number of iteration, the early stop criteria as well as the learning rate are set up in such a way that: for clearly biased hash tag, its sentiment score would at most change 50% per day; for neutral hash tags, its sentiment score could change almost 100% per day, much more flexible than clearly biased ones.

It is worth mentioning that for my current data set, using the expanded set of marked hash tags to estimate sentiments of tweet messages directly, even without actually looking at the text messages, is already very accurate. As explained above, this might due to the *correct key observation* that most information from a tweet is centralized around hash tags and keywords.

4.2 Linear Method of Estimating Sentiments Carried by Hash Tags

In previous Section.4.1, we tried to expand the set of hash tags with sentiment scores by assigning new scores and adjusting existing scores according to correlations between hash tags. *The motivation is that by expanding the set of hash tags with sentiment scores, we would be able to estimate sentiments of more tweets, particularly those that didn't call any hash tag with manually marked sentiment scores.*

An improvement over the simple iteration method could be achieved *if we handle the problem of estimating unknown sentiments carried by neutral hash tags in the context of classifying tweet text messages according to their sentiments*[17]. Since we don't have a "golden standard" training corpus, we will treat the sentiment of a tweet message estimated using manually assigned sentiment scores as *a good approximation to the true* Y *value*[18] and assume that this approximated Y value depends linearly on the combined sentiments of hash tags called by this tweet[19][20], then one could perform a *linear classification with a Ridge penalty term* on the set of Y and X values, with the coefficient vector β as sentiment scores of the expanded set of hash tags that we are trying to estimate. Mathematically:

$$\hat{\beta}^{ridge} = \underset{\beta}{argmin} \left\{ \sum_{i=1}^{N} l(y_i, x_i\beta) + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{1}$$

where N is the total number of tweets, p is the total number of hash tags under study. Since we are estimating sentiment scores for both candidates, β_j would be the pair of sentiment scores for the jth hash tag[21] and y_i would be a 2-dimensional vector as the sentiment of the ith tweet[22].

[17] Ideally, the target variable will be *true* sentiment of each tweet text message, the predictor variable will be *both* the neutral and manually marked hash tags called by each tweet.

[18] Here we use the key assumption that for most of the time, hash tags with clear sentiment biases are used accordingly. See Section.4.1.

[19] Here we define the combination of hash tags called by each tweet as the X vector. Thus the dimension of X vector p equals the total number of hash tags of which sentiment scores we want to estimate. For a specific tweet, its X vector would be a vector of 0 and 1, where 1 corresponding to the hash tags that it called.

[20] In this setting, the *true sentiment scores of each hash tag* would be the *true value* of the corresponding element on the coefficient vector.

[21] β_j is the jth row of matrix β. The matrix β is initialized as all zeros.

[22] y_i is the ith row of the target matrix Y

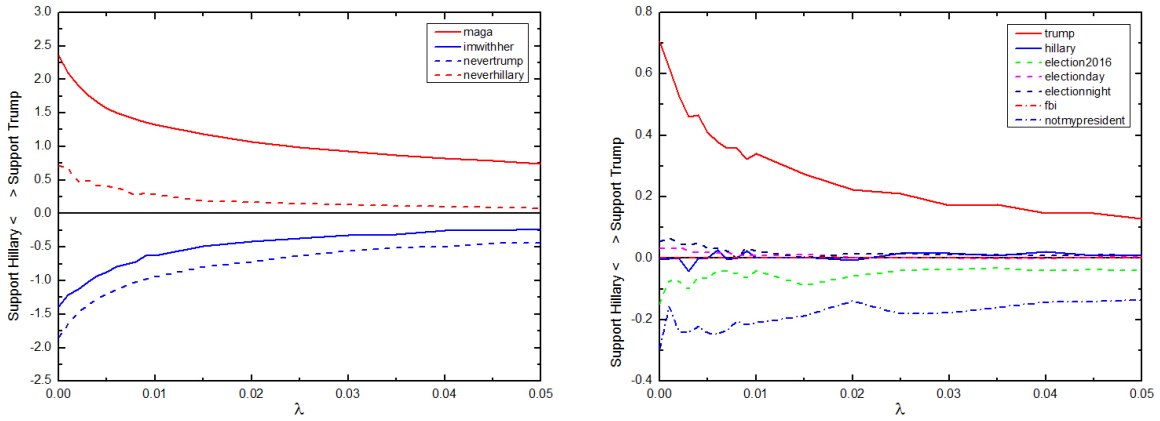


Figure 5: The shrinkage of β_js with increasing Ridge penalty, where β_j is the pair of sentiment scores of the jth hash tag under study. Left hand-side are four "manually marked" hash tags. Although their β_js were initialized as zeros, the model nonetheless provided a correct estimation. Right hand-side are neutral hash tags, of which the sentiment scores are much smaller, reflecting less weights were allocated to them by the model. Data taken on Nov 10th.

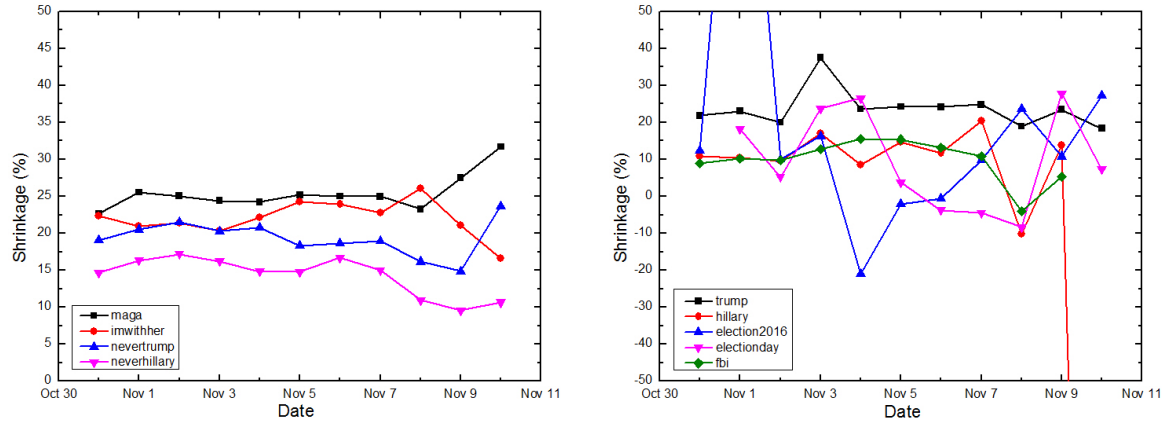


Figure 6: Shrinkage in percentage (β_j with $\lambda = 0$ divided by β_j with $\lambda = 0.05$) from Oct 31st to Nov 10th. Left hand-side are four manually marked hash tags. Right hand-side are neutral hash tags.

To have an expanded hash tags set, or, to have an estimation of the unknown sentiments of neutral hash tags, a standard linear classification with negative log-likelihood would be sufficient, since we already included neutral hash tags into our predictor variables. However, such an estimation would be of little value because we don't know *how reliable the result is*. To solve this problem, one could include a Ridge penalty term into Equation.1, exploiting the property[23] that Ridge regression will shrink coefficients along the principle components in the p-dimensional space, with the high-variance direction shrinking slower. In the end, coefficients of high variance variables[24] will have more weight in the model. In other

[23] See Section 3.4.1 of Ref.[5].

[24] Effectively hash tags that are more capable of determining sentiments of tweets. For example, #crookedhillary would correspond to a principle component of higher variance compared with #vote.

words, if the value of β_j corresponding to a neutral hash tag shrinks slower, then we would expect this hash tag's sentiment scores to be more reliable.

This is exactly what happened in Fig.5 and Fig.6. Fig.5 provides values of β_js of selected hash tags. One could observe that both manually marked and neutral hash tags have a decreasing β_j with increasing Ridge penalty term, represented by λ. Naturally, manually marked hash tags would have share bigger weights inside the model, thus larger β_j values. Nonetheless, several neutral hash tags like #trump, #notmypresident and #election2016 also shared rather significant weights inside the model. *The method of estimating reliability of our estimations of these neutral hash tags is to study their shrinkage compared with that of manually marked hash tags.* Manually marked hash tags, due to their large share of weights inside the model, would shrink smoothly, as indicated by the left hand-side of Fig.6. At current setting, their $\lambda = 0.05$ values would shrink to at least 15% of their $\lambda = 0$ values. And their shrinkage behaviors should be smooth across time. For the neutral hash tags, the story is very different. From the right hand-side of Fig.6 one could observe that neutral hash tags shrink more, and their shrinkage level fluctuate violently across time, indicating their strong dependence on *the context*[25]. Among these neutral hash tags listed here, #trump was the most stable. #fbi was also relatively stable, with a larger shrinkage level corresponding to its smaller weights inside the model, if one compares #fbi against #trump.

4.3 Nonlinear Method of Estimating Sentiments Carried by Hash Tags

Let's start with the Gaussian Mixture Model. One could imagine that each hash tag has a certain distribution on the 2-dimensional sentiment space of Fig.7. Hash tags like #vote4hillary and #MAGA would distribute more closely to the positive Y and X axes. At the same time, each tweets would be represented as a point on this space. The reason that hash tags are thought as a distribution is because hash tags are used by many tweets with many different expressions and sentiments. This is similar to the case of Gaussian Mixture Model where the position of each data point could be considered as a result determined by several Gaussian distributions. Similarly, sentiment of a tweet could be treated as a function of sentiments of the hash tags it called.

The nonlinear method I introduce here uses a single layer neutral network (phNNeT)[26] which projects the combination of hash tags called by each tweet onto the sentiment space of Fig.7. Then it uses the projection to determine the sentiment of the tweet. *Details of the algorithm are listed below*:

1. each x_i represents the combination of hash tags called by a tweet[27]. y_i represents the sentiment score of this tweet[28]. The input matrix X has two dimensions: 1st as number of tweets N, 2nd as number of hash tags under study p. The output vector Y has its dimension equal to N.

[25] For example, how were they used on a specific date. The violent fluctuation of #hillary on Nov 9th was a good example.
[26] phNNeT stands for Projected Hidden space Neural Network.
[27] Similar to Section.4.2, each x_i is is a vector of 0 and 1, with dimension p equal to number of hash tags.
[28] Similar to Section.4.4 and Section.5, sentiment scores for each candidate is in the range between -1 and 1. We convert a *pair* of sentiment scores into $y_i = 1$ if candidate Donald Trump's sentiment score is *larger than that of candidate Hillary Clinton plus a threshold*. Similarly for the case of $y_i = 0$.

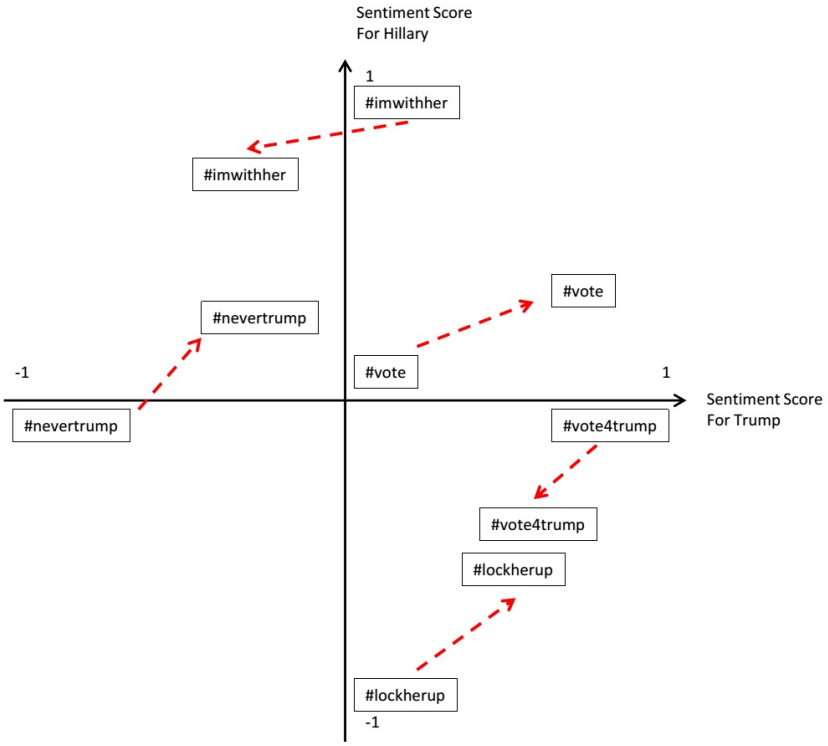


Figure 7: Illustration of the concept. The X and Y axes create the 2-dimensional sentiment space, onto which the $p*2$-dimensional hidden "vector" $f(x_i)$ will be projected as p points. Of course, most of these p points will be locked at the origin, representing hash tags that are *not* called.

2. the hidden matrix H has two dimensions: both equal to p. The output matrix W has two dimensions: 1st equals to p, 2nd equals 2[29]. We use the $sigmoid(x) = 1/(1+e^{-x})$ as the *activation* function and *tanh* as the *output* function. Thus

$$f(x_i) = tanh(b_2 + sigmoid(b_1 + x_i \cdot H) \cdot W) \tag{2}$$

   $f(x_i)$ will be a matrix of dimension $p*2$, with only the rows corresponding to the 1s in x_i being nonzero. $f(x_i)$ could be understood as a set of coordinates for all the p hash tags under study, but the hash tags that are *not* called by this tweet will be locked at the origin.

3. the sentiment for the tweet will be a single number, in this case either 0 or 1. It is determined by comparing sums of coordinates along X and Y axes plus a threshold.

$$sentiment(x_i) = s \tag{3}$$

$$if \left\{\sum_{i=0}^{p} tanh(f(x_i))\right\}_s \geq \left\{\sum_{i=0}^{p} tanh(f(x_i))\right\}_t + threshold(f(x_i)) \tag{4}$$

   Here s and t equal 0 or 1. The threshold function has a minimum value. And it is further adjusted by the separations between hash tags of each tweet on the sentiment space. The larger the separation, the higher the threshold. One could go without the *tanh* of Equation.4.

[29] As we are projecting the hidden values onto the 2-dimensional sentiment space.

4. For those ∼200 hash tags with manually marked sentiment scores, it is recommended to initialize their hidden values according to their scores. For hash tags without manually marked sentiment scores, their initial hidden values will be zero.

In this way, during the training processes, hash tags more often used together will be pulled closer on the sentiment space. Since we used *tanh* function and a dynamically adjusted threshold to convert hidden values into sentiment score for each tweet, hash tags with clear biases will stay close to positive or negative ends of the X and Y axes. The reason we dynamically adjust the threshold according to separations among hash tags called by each tweet is very similar to the idea of Ridge regression: we want to avoid model variables get into large positive/negative values by canceling each other.

4.4 Results of Sentiment Analysis using only Hash Tags

	Number of Tweets between '2016-11-07 00:00:00' and '2016-11-10 00:00:00'	Note
Tweets collected as relevant to Election 2016	186505	Nov 9th I have analyzed only the first half day's data.
Tweets left after N_word Filter	22218	Filter as: contain at least 10 words that are neither user name nor https. Nov 7th: 5147; Nov 8th: 9301; Nov 9th: 7770
Tweets as "Support Trump" by hand-marked tags only	12898	Nov 7th: 2998; Nov 8th: 4337; Nov 9th: 5563
Tweets as "Support Hillary" by hand-marked tags only	8958	Nov 7th: 2057; Nov 8th:4789; Nov 9th: 2112
Tweets as "Support Trump" by Joint Sets of Tags	12876	Nov 7th: 2996; Nov 8th: 4317; Nov 9th: 5563;
Tweets as "Support Hillary" by Joint Sets of Tags	8550	Nov 7th: 1885; Nov 8th: 4760; Nov 9th: 1905;

Figure 8: Statistics of predictions made on tweet sentiments, using only hash tags without analyzing text messages. Joint set of tags are the set of manually marked hash tags and the expanded set of hash tags. Please check Ref.[4] for details about the thresholds, estimated prediction accuracy as well as the ∼300 tweets I manually analyzed.

Fig.8 is a summary of prediction results, using data from Nov 7th to Nov 9th. I also manually marked ∼300 tweets from Nov 7th to estimate the accuracy of predictions made by joint hash tag sets[30]. One could find them in Ref.[4]. The prediction accuracy for tweets related to both candidates are ∼75%[31].

[30] Using both the ∼200 hand-marked hash tags set and the larger expanded hash tags set from the iteration process.

[31] Although this 75% is estimated by checking ∼300 tweets (less than 2% of the total predictions made), yet it should at least offer a preliminary proof of the effectiveness of my method.

I feel that it is necessary to offer an elaboration on "Why using hash tags alone, without looking at the text message itself, could achieve a reasonably good prediction accuracy". *This has everything to do with twitter users' habit of writing their messages and expressing their opinions.* Twitter users have several distinct habits:

1. their messages are very succinct. This means twitter users have to express whatever they want to say through these commonly used keywords and hash tags. Thus, information is highly concentrated, to the point where if one get the few hash tags correct, he could get the entire message correct.

2. in addition to the first point, the use of hash tags also normalizes twitter users' language.

3. tweet messages are context-heavy. People write tweets to communicate with their friends, with whom they share a lot of "common languages". Words like "tax paper", "emails", "Benghazi" are much more *meaningful* that they usually are. However, traditional NLP methods won't be able to catch it.

5 Long Short Term Memory Method

Tag	Freq	Tag	Freq	Tag	Freq	Tag	Freq	Tag	Freq	Tag	Freq
UNKNOWN_T	0	on	25	out	50	america	75	trump2016	100	great	125
.	1	that	26	about	51	people	76	time	101	benghazi	126
!	2	crookedhillary	27	just	52	now	77	cant	102	country	127
the	3	not	28	no	53	demsinphilly	78	election	103	dncinphl	128
to	4	are	29	but	54	by	79	-	104	because	129
,	5	be	30	at	55	how	80	why	105	bernieorbust	130
a	6	we	31	makeamericag	56	im	81	draintheswam	106	donald	131
is	7	vote	32	they	57	me	82	make	107	there	132
of	8	hillary	33	as	58	dumptrump	83	see	108	than	133
imwithher	9	will	34	trumppence16	59	clinton	84	lockherup	109	say	134
for	10	``	35	get	60	more	85	would	110	good	135
and	11	''	36	was	61	us	86	want	111	love	136
maga	12	all	37	:	62	trumptrain	87	going	112	them	137
you	13	with	38	like	63	2	88	never	113	been	138
nevertrump	14	...	39	up	64	or	89	right	114	were	139
trump	15	have	40	can	65	votetrump	90	hillaryclinton	115	hillaryforprisor	140
in	16	he	41	has	66	when	91	only	116	way	141
?	17	dncleak	42	dont	67	one	92	need	117	)	142
i	18	her	43	its	68	dnc	93	him	118	bernie	143
this	19	your	44	who	69	u	94	over	119	win	144
;	20	my	45	dncleaks	70	know	95	did	120	lets	145
&	21	so	46	our	71	president	96	voting	121	$	146
amp	22	what	47	his	72	4	97	think	122	hrc	147
neverhillary	23	if	48	do	73	go	98	their	123	obama	148
it	24	she	49	from	74	an	99	should	124	got	149

Figure 9: Word to Index dictionary. Top 150 most frequently used words.

The Long Short Term Memory (LSTM) method is a very powerful method for sentiment analysis. Like most neural network methods, LSTM is a supervised learning method. At its current configuration, my codes performs NLP and LSTM in this way:

	Number of Tweets	Note
Tweets: analyzed by LSTM	114445	total number of tweets analyzed by LSTM
Tweets: training data as support Trump	71045	training data generated by the Joint sets of hash tags, both manually set scores and the expanded set of scores
Tweets: matching training data as support Trump	69111	LSTM classification by argmax() matching training data
Tweets: perfectly matching training data as support Trump	26483	LSTM predicted Probability distribution with a 100% on Trump
Tweets: training data as support Hillary	43400	training data generated by the Joint sets of hash tags, both manually set scores and the expanded set of scores
Tweets: matching training data as support Hillary	42261	LSTM classification by argmax() matching training data
Tweets: perfectly matching training data as support Hillary	16183	LSTM predicted Probability distribution with a 100% on Hillary

Figure 10: Statistics of predictions made by LSTM, compared with training data generated using the "Semi-Automatic" method.

1. using NLTK package to tokenize tweet messages, creating word-to-index and index-to-word dictionaries. Each tweet message will be represented as a list of indexes. The top 150 most used words are given in Fig.9.

2. training LSTM using this "Semi-Automatically" generated training data set. The LSTM returns hidden value vectors[32]. The prediction for each tweet message is made by a numerical average over the corresponding hidden value vector.

Please note that from Fig.10, the mismatch between the "Semi-Automatically" generated training data set and the LSTM predictions is also small, proving that *the LSTM is very effective as a supervised learning method.*

Compared with all the previous methods, LSTM assigns weights onto *both* hash tags and keywords from tweet text messages. This is verified by Fig.10. For both candidates, only ∼30% of the LSTM's predicted probability distribution are *in a perfect match of the training data*, while the rest demonstrate a slightly mixed probability distribution. This indicates that weights of the model are further distributed towards other keywords or high frequency words.

Since we don't have a sizable, manually generated training data set as the "golden standard", it is difficult to argue at this point whether distributing weights of model into non-hash tag words would

[32]Length of each hidden value vector is equal to length of corresponding tweet message

improve the performed of the LSTM or not. *Particularly, whether the predictions by a LSTM trained using the "Semi-Automatically" generated training data set would outperform predictions made by hash tags alone.* Nonetheless, I would like to point out here that *one might as well use the LSTM model in an unsupervised way: using the LSTM to observe and extract patterns from tweet text messages that have a strong correlation with certain combination of hash tags.* Please note that those word marked blue in Fig.8 are not hash tags, yet they were frequently used and clearly carried positive or negative sentiments. The LSTM might be able to find the correlation between these words and hash tags. And when words and hash tags expressing opposite sentiments were used together, a mixed probability distribution would be generated by the LSTM, indicating an abnormal case or even *"correcting" the training data set.* Without a "golden standard" training data set, it would be difficult to prove my arguments. However, I attached some of those mismatches between the LSTM prediction and "Semi-Automatically" generated training data set in Ref.[7], in case the reader like to explore a little more.

6 Introduction to the Dataset of this Project

In order to achieved the primary objective, one has to download and analyze huge amounts of social media data. The reason that a large, continuous data set is necessary is:

1. we what to have a continuous data set because we are trying to understand individual's shifting opinion.

2. each individual's activity is sparse in time, simply because most people doesn't spend all day on Twitter.

Thus one needs to collect data for a reasonably long period of time. In this case, I have so far collected[33] 40 days' worth of data, among which 18 continuous days in July[34], 7 continuous days in October[35]and a continuous data set from Oct 31st on-wards. On average, each day's worth of data is 7GB with 3 million tweets. I am only collecting tweets with geo-information indicating its user located within the United States.

It is worth mentioning that 3 million tweets per day is apparently too few for such a user base. My guess is that the amount of tweets I could collect is limited by my twitter API and my query speed. However, since tweets were downloaded randomly (those that are got/missed by my API), it won't have any negative effects on my analysis. Furthermore, for most parts of my analyses, various filters were implemented on the raw data. For example, only tweets with hash tags related to keywords "trump" and "hillary" were considered relevant to the topic under study. Such filters would reduce the number of tweets that were analyzed in Phase 2 and Phase 3.

[33]commented on Nov 16th, 2016.
[34]July 13th to Aug 2nd, 2016.
[35]Oct 15th to 21st, 2016.

7 Technical Details

This set of codes are written in Python 2.7, using libraries that includes: theano, pymysq, nltk and other routine libs like numpy and pandas. Data management requires MySQL. The tokenization is performed with the NLTK package. The LSTM training and predictions codes are adapted from the tutorial codes from Ref.[6]. I adapted the training codes so that it would work on my data sets; I added prediction codes so that it would work with other parts of my Phase 2.

References

[1] Hao Wang, Dogan Can, Abe Kazemzadeh *A system for real-time Twitter sentiment analysis of 2012 U.S. presidential election cycle*, Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics

[2] Efthymios Kouloumpis, Theresa Wilson, Johanna Moore *Twitter Sentiment Analysis: The Good the Bad and the OMG!*, Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media

[3] The ∼200 manually marked hash tags could be found here: https://github.com/Nimburg/Ultra_Phase1v5_Phase2v2_Phase3v1/tree/master/Ultra_Phase3v1/Data

[4] The manually analyzed tweets could be found here: https://github.com/Nimburg/Ultra_Phase1v5_Phase2v2_Phase3v1/blob/master/Results_Demo/Prediction_by_Tags_only.xlsx

[5] Trevor Hastie, Robert Tibshirani, Jerome Friedman *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, Springer Series in Statistics

[6] Deeplearning: http://deeplearning.net/tutorial/contents.html

[7] Mismatches: https://github.com/Nimburg/Ultra_Phase1v5_Phase2v2_Phase3v1/blob/master/Results_Demo/LSTM_Predicts_Mismatches.xlsx

[8] Github deposit of this project: https://github.com/Nimburg/Ultra_Phase1v5_Phase2v2_Phase3v1