CS 229 - Project Final report *Hooyeon 'Haden' Lee; hlee0 (05382015); 12/16/2011*

Title: Using Twitter to Estimate and Predict the Trends and Opinions

1 Introduction

The most common, traditional way to collect people's opinions is random sampling and asking survey questions by phone. If a News media is interested in knowing how popular Obama is, it would call up a thousand people and ask how they would rate Obama. The data collected in this manner is considered a gold standard. However, this approach bears shortcomings such as costs, limited targeted people, etc.

In this paper, we attempt to predict people's opinions and trends by analyzing the Web data such as Twitter. There are several interesting questions we could ask, given billions of Tweets: how people feel about economy; how people rate Obama; what people think about the top five contestants left in the TV show, American Idol. Collecting polls by asking people in person or by phone is costly, but if we could get the same results from the freely available Web data, it could supplement or supplant the conventional way of collecting polls[1].

2 Problem Description

To formalize our problem, let us define a time series variable $p(t)$ that we would like to predict; for instance, Obama Job Approval rate that ranges in 0% and 100% could be such a time series variable. Then our problem simplifies to predicting $p(t)$ using the Twitter data.

Since the Tweets are also time series data, we can denote another time series variable $q(t)$ that can be obtained by some text analysis over the Twitter data (this could be, for instance, a certain word's frequency at time t). Hence, we could try to correlate these two time series $p(t)$ and $q(t)$ via learning models such as linear regression model. For β-day forecast, we would correlate $q(t)$ with $p(t+\beta)$ (e.g. predicting β days ahead).

3 Related Work

Google recently developed an online tool called *Google Flu Trends* [2], which attempts to predict the number of influenza patients based on the search queries on Google.com. The actual number of patients in the United States is reported by the Center for Disease Control (CDC), but the report comes out with a 2-week lag. In this work, the authors attempt to train models and predict via a simple linear regression model over the log-odds of two events: $p(t)$ which is the fraction of physician visits among all visits and $q(t)$ which is the fraction of influenza-related search queries. This simple model provided a good estimate of the number of flu patients, which matched the actual CDC reports. In their model, β was set to zero (hence, predicting the same day trends).

In another related work 'From Tweets to Polls'[1], the authors attempt to solve a very similar problem using the Twitter data. The authors trained a linear model after applying the sentiment analysis over each Tweet by checking whether a Tweet contains a positive word and/or a negative word using a standard lexicon; each Tweet could be marked either positive or negative or both. For predicting Obama's Job Approval rate, they limited the input space to the set of the Tweets that contain a word 'Obama,' on which the sentiment analysis was performed. A simple linear regression model was slightly better than their baseline prediction. For predicting other tredns such as the consumer confidence index, the authors report that their model did not work well.

4 Data

4.1 Data Collection

We used the publicly available Tweets from Jul'10 to Nov'11, which were crawled via public Twitter APIs. For the trend data, we only had time to work with the Obama Job Approval rate data from Gallup.

4.2 Data Statistics

In this work, we used all public Tweets from 7/15/2010 to 11/30/2011, inclusive, and the Gallup's Obama Jab Approval polls data from 7/15/2010 to 11/30/1011, inclusive. During this period, we have the following statistics (here, 'Qualified Tweets' refer to the Tweets that contain 'Obama' (case-insensitive) as a substring):

# of days	504
# of Tweets (Total)	11,470,198,016
# of Tweets (Daily)	22,758,329
# of Qualified Tweets (Total)	1,737,394
# of Qualified Tweets (Daily)	34,471

Figure 1 shows the time vs. volume of Tweets plot while Figure 2 shows the fraction of Qualified tweets. The green line in Figure 1 (the number of Qualified Tweets) is scaled up by multiplying by 100. The left-most vertical line is when Osama Bin Laden was killed; the number of qualified Tweets bursts out on that data while the total number of Tweets does not change much. On the other hand, during the week of 10/11/2011, the 'Occupy Wall Street' movement/riot was prevailing around the globe, and the total number of Tweets spiked during this week (the right-most vertical line), while the fraction of qualified tweets did decrease by a bit. These two plots together suggest that people do share their opinions via Twitter.

Table 1: *Top words with respective train/test errors with* $\mu = 3$

Rank	Word	Train Err (Single)	Test Err (Single)	Train Err (Agg)	Test Err (Agg)
1	cong	1.88353	3.44086	1.88353	3.44086
2	nov	1.89899	4.3944	1.7073	3.83679
8	#2011	1.9683	3.92031	1.30232	3.85882
9	#thatawkwardmoment	1.973	5.88304	1.2216	4.41469
11	#nv	1.9891	4.69948	1.16341	5.07407
15	#vote2010	1.9996	5.07752	1.12839	3.5898
17	#whyivote	2.00431	4.94258	1.1247	3.61808

5 Method

5.1 Assumptions and Pre-Processing

First of all, given a prediction problem (or a topic), we must admit that not all Tweets are related to the given topic; in fact, the majority of Tweets would be irrelevant. Hence, it is important to limit our input space to a proper subset of Tweets. This introduces a different but relevant problem called 'topic analysis'; given a Tweet, can we identify what topic(s) this Tweet belongs to? Since this problem itself is an interesting, difficult problem, we do not attempt to solve this problem in this work, but we would simply filter the Tweets as a pre-processing step. For instance, for the Obama job approval prediction, we could simply consider the Tweets that contain 'Obama'.

Second of all, as the authors in [1] pointed out, Twitter has its own language in the sense that people post Tweets using hash tags, URLs, Internet slangs, etc., in order to express their opinions within 140 characters. Considering that each Tweet consists only of 11 words on average, this is barely a short sentence. Hence, a simple sentiment analysis may not work well for analyzing the twitter data. In this work, we propose that the term frequency count (uni-grams, bi-grams, etc.) and aggregated term frequency (top k uni-grams, etc.) could be used in learning the time series prediction model.

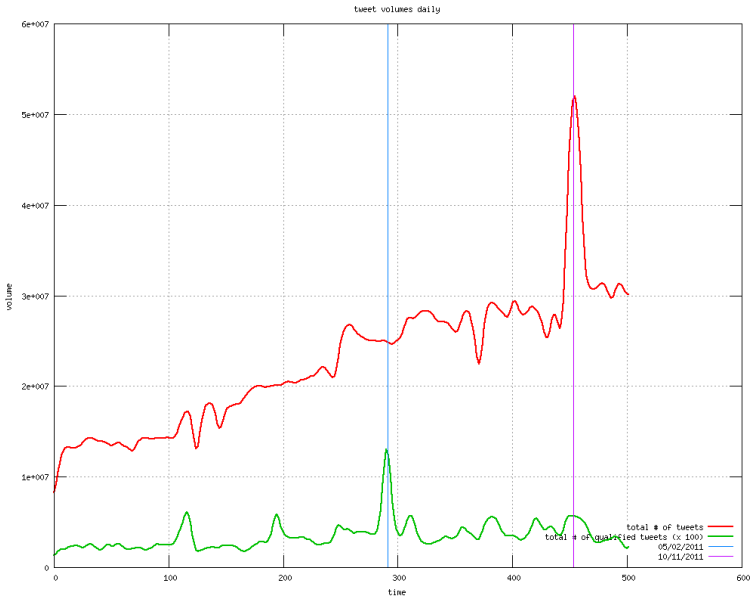

Figure 1: *Time vs Tweet Volume*

5.2 Learning Models

We used linear regression model where the features are word frequencies and target variables are Obama's Job Approval rate. To be more formal: $n = 504$ is the total number of days. In our Vocabulary (English words and hash-tags), we have $m = |V| = 346,880$ words. For each word w_i, the corresponding feature vector $x^{(i)} \in \mathbb{R}^n$ is defined as: $x_j^{(i)}$ is the word frequency of w_i on day j. In particular, $\sum_{i=1}^{m} x_j^{(i)} = 1$ for all j (i.e. the sum of word frequencies of the same day equals to one). $\vec{y} \in \mathbb{R}^n$ is defined as the Gallup's poll data.

5.2.1 Linear Regression - Single Word

We can easily think of a linear regression model to learn the parameters $\theta^{(i)} \in \mathbb{R}^2$ using just a single word w_i by solving the famous normal equation.

$\theta^{(i)} = [(x^{(i)})^T x^{(i)}]^{-1}(x^{(i)})^T \vec{y}$

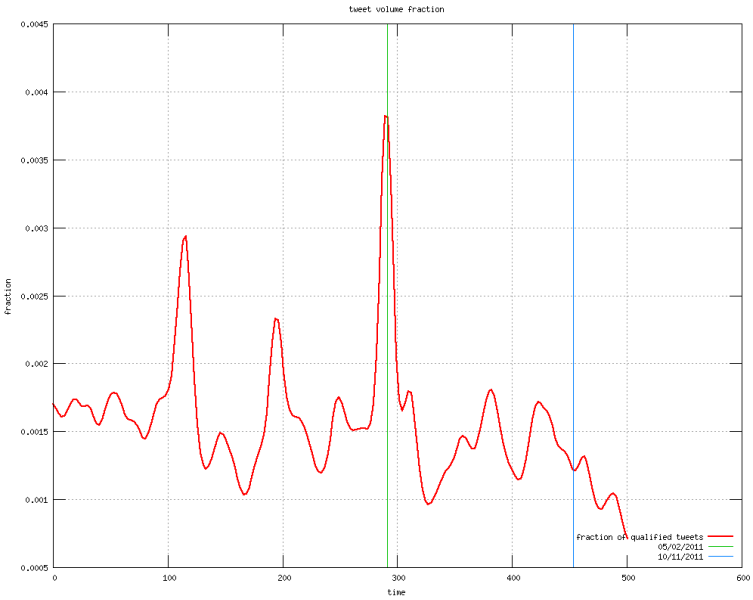

Figure 2: *Time vs Qualified Tweet Fraction*

5.2.2 Linear Regression - Aggregate Words

Now, suppose we learned $\theta^{(i)}$ for all m words. We can sort them by the training error to get the best k words that work best on training set, and use these top k words altogether to learn a new linear regression model. In this case, the parameters $\Theta^{(k)}$ we want to learn would be in dimension $k+1$. Learning Θ is identical to learning θ in that we just have to solve the normal equation:

$\Theta^{(k)} = [(X^{(k)})^T X^{(k)}]^{-1}(X^{(k)})^T \vec{y}$

where X is the design matrix containing the top k feature vectors as rows. This approach mimics the work in [2].

5.2.3 Time Parameters

Notice that the above linear regression models try to predict 0-day forecast (same-day). However, it would be more useful and practical if we could produce β-day forecast. For instance, the Tweets posted today may be a good indicator of the true polls data in 3 days or a week. Hence, we introduce the time parameter β that indicates how far in the future we would like to predict. In our work, we would consider the values $\beta = 0, 1, \ldots, 30$ ($\beta = 7$ means predicting the trends 7 days ahead). Then, we would instead try to minimize the sum of squared errors:

$||y_{j+\beta} - (\Theta_0^{(k)} + \sum_{i=1}^{k} \Theta_i^{(k)} x_j^{(i)})||^2$

6 Experiments and Findings

6.1 Experiment Setup

Single-word Model

The time parameter β varies between 0 and 30. Since we have 504 days of data, we train our model on days $[t_s \ldots t_e)$ and test our model on days $[t_e \ldots 504)$ (our days are 0-indexed). With these three parameters fixed, we learn single-word linear regression models for each word in our vocabulary V (recall that $|V| = 346,880$), and we sort the words by how well they fit the target variables (Obama Job Approval rates).

Aggregate Model

Based on this measure, we pick the best k words (where $k = 1 \ldots 100$) and train an aggregate-word linear regression model as explained earlier. Our findings focus on using these aggregate-word models, and our single-word models play a role as a feature selector for ggregate-models.

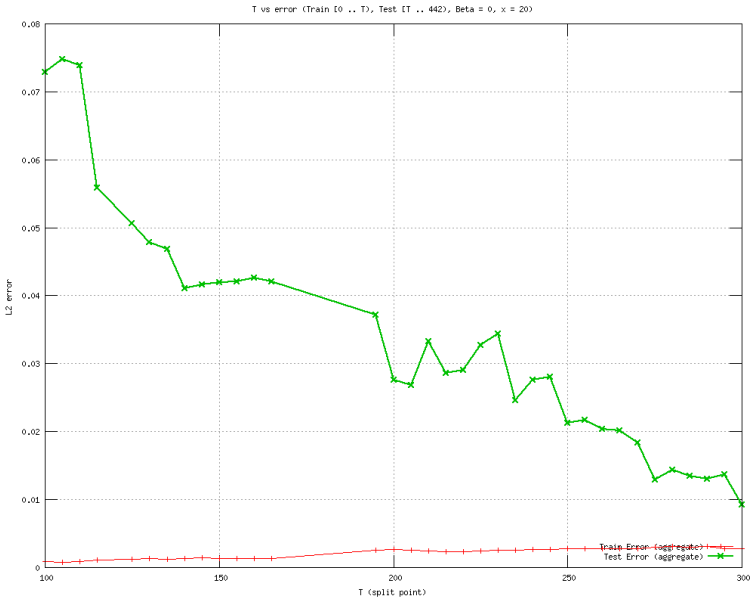

Figure 3: *Training Set Size vs Error*

6.2 Results

Table 1 summarizes the train/test errors when $\beta = 3$, showing some of the top 20 words due to the limited space. Notice that we see some politically relevant words such as 'cong' (congress), '#vote2010', '#whyivote', etc. The following sections will discuss what each of the parameters tells us about the data.

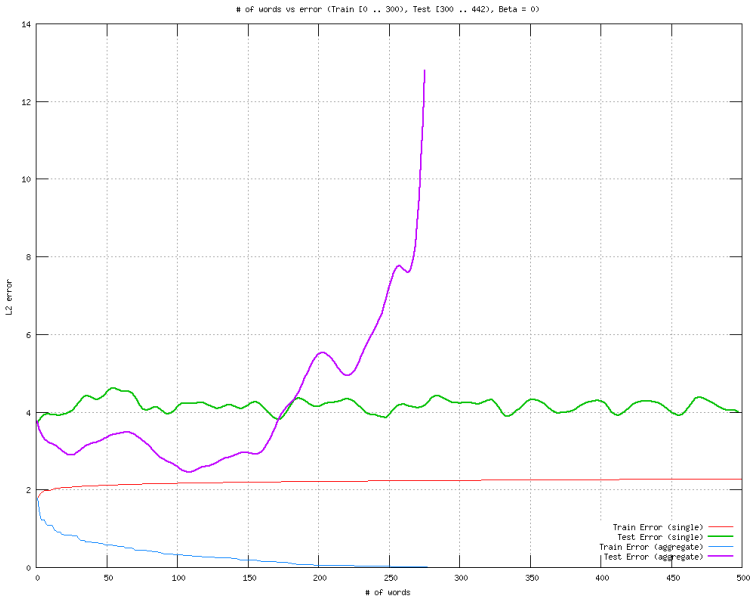

Figure 4: *Aggregate Model Size vs Error*

6.2.1 Effects of Training Set Size

For fixed β and x (the number of words used in aggregate model), as we vary $T = t_e - t_s$ (the size of training set), we would expect that the larger training set will help our model learn the target trends. Training on 5 days of data would make our model too simple, while training on 200 days of data would yield a more reasonable model. Figure 3 shows how train/test errors change when we vary t_e from 100 to 300 (x-axis) while we fix parameters $\beta = 0, t_s = 0, x = 20$. In this plot $T = t_e$ represents the split-point of train set and test set. As we see, the test error (in bold green) has a decreasing trend in that our models benefit from more data, while train error (in red) increases and decreases but does not fluctuate much.

6.2.2 Effects of Size of Aggregate Model

For fixed β, t_s, and t_e, an aggregate model is built on top of the best x words measured by single-word models. By varying x, we can see the effects of having fewer or more words in aggregate model. Naturally, we would expect that our models will fit the training data better as we have more words (hence more features) – we can overfit. However, for the test-error, it will initially decrease as our model can capture the trend better, but eventually it overfits and blows up.

In Figure 4, we can clearly see this effect. As we add more words to the aggregate model, we see that the train error continues to decrease (in blue), while the test error (in purple, bold) is minimized at around 105 words, and then starts to blow up; after having 270 words, the aggregate model's test error was too huge to fit the plot. The red line represents the train error of each word in single word model, and this is monotone-increasing because we sorted the words by their train errors. The green line represents the test error of a single word, which appears to be a bit random as single word is clearly not a good predictor.

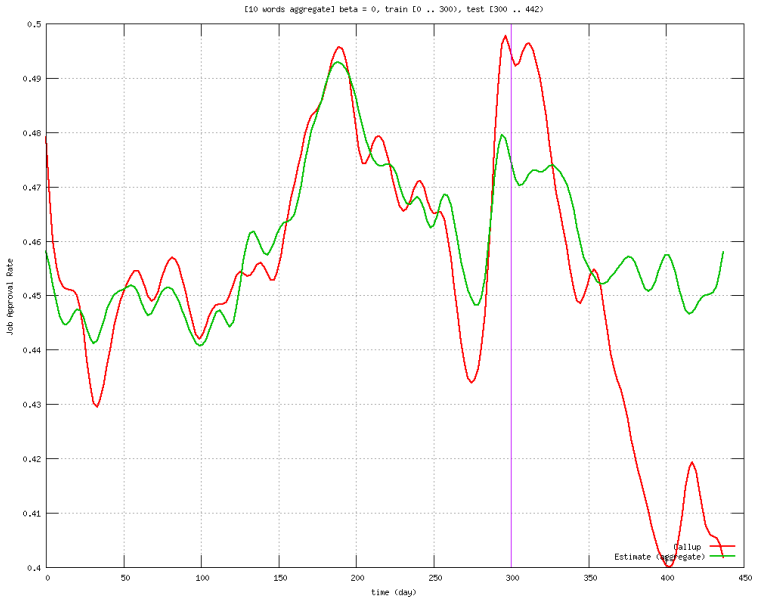

Figure 5: *Prediction Using 10 Words*

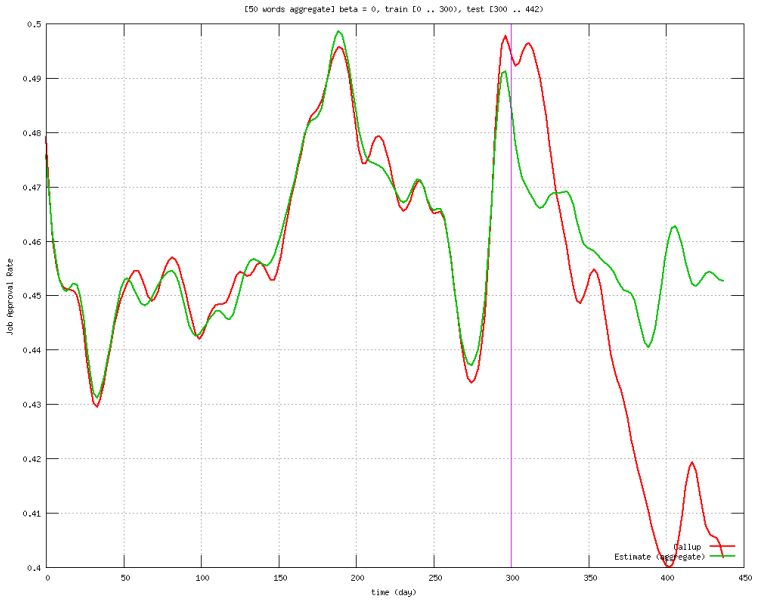

Figure 6: *Prediction Using 50 Words*

For comparison, Figure 5 and Figure 6 show how our aggregate models predict the trends when we keep all parameters the same except for the number of words used in aggregate model. In these two plots, the red curve is the target trend (Gallup data), and the green curve is our model's prediction. For these plots, we fixed $\beta = 0, t_s = 0, t_e = 300$, but the numbers of words used are 10 and 50, respectively.

Notice that Figure 5 does capture the trends in training set quite well, while Figure 6 captures the trends in training set almost perfectly. The 50-word model captures the trends in test set much better than the 10-word model.

6.2.3 Comparison with Random Models

Previous results confirmed that aggregate models do work better as we use more words, which begs the following question: what happens if we just randomly pick words from our V and train aggregate models? Would these model fit the trend curve just as perfectly as our models? If the answer is yes, then our models are not any better than randomly picking a set of words and learn linear regression models. Here, we fix all the parameters β, t_s, t_e, and x while we train our aggregate models with a different set of words: the best x words (our model) and the random x words (random model). This random model serves as a baseline in our work.

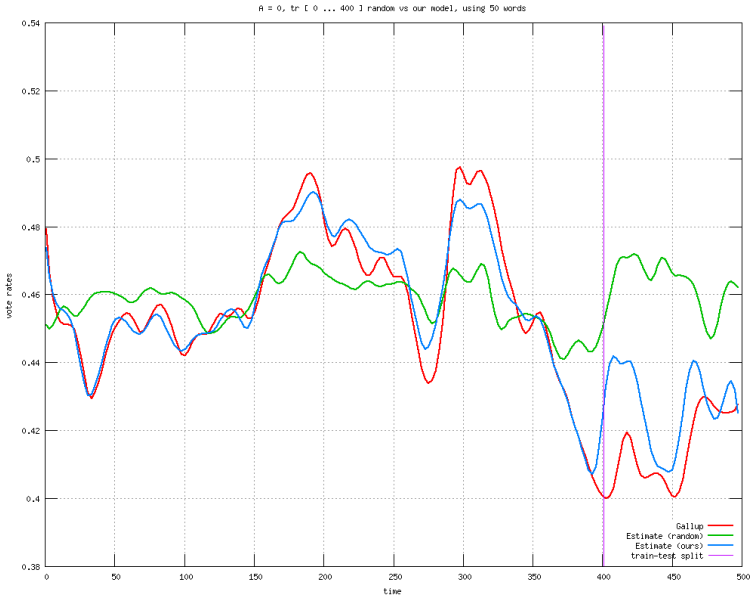

Figure 7: *Our Model vs Random Model*

Figure 7 clearly shows that our model fits the test data very well while the random model cannot even fit the training set very well. The difference is more obvious in the test set (on the right side of the vertical line). Here, the parameters used are $\beta = 0, t_s = 0, t_e = 400, x = 50$. This confirms our assumption that filtering the features (words) by how well they fit the target trend individually, via single-word modes, is a good feature selection process (because we used the same aggregate linear models just with different set of features).

Figure 8 shows L2 errors of our model and random model when we increase the number of words used in aggregate model. The train error continues to decrease as before (our error in red and random model in blue). However, We see an increasing trend for the test error of random model (in purple, bold), while our model's test error decreases at first and then stabilizes (in green). This clearly shows that our model performs significantly better than the random model. In this plot, parameters were $\beta = 1, t_s = 0, t_e = 300$.

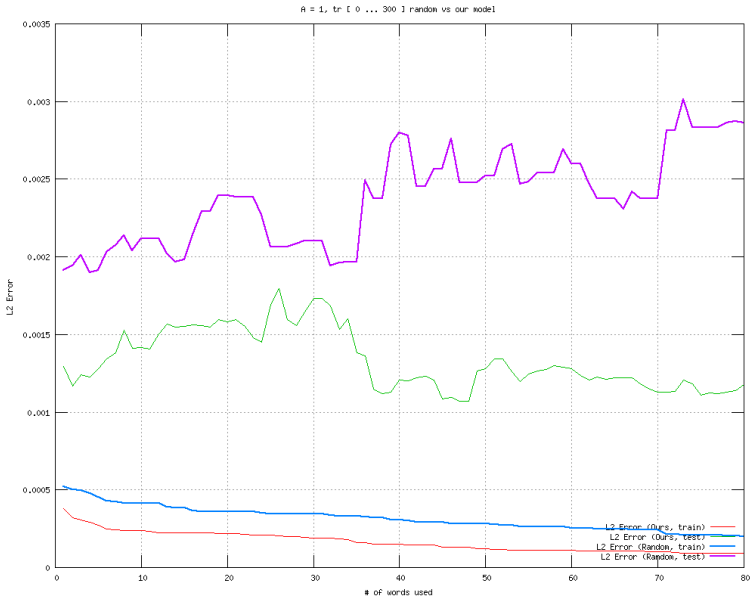

Figure 8: *# of Words vs L2 Error*

6.2.4 Evaluation on Word Set

Although we compared our model against a random word aggregate model (which served as a baseline in our work), we could evaluate our model in different ways. To evaluate whether we selected a good subset of words, we can look at the actual weights (or θ values in our equations) in aggregate model and see how these weights vary. Intuitively, if our word set is bad or random, these weights should fluctuate a lot as we vary the size of aggregate model while keeping other parameters fixed, because in such a set, the words are not meaningful features to predict the trends. On the other hand, if our word set is a good predictor, then we should see a rather stablized weights across the words, such that some words are positively correlated to the trend while others negatively.

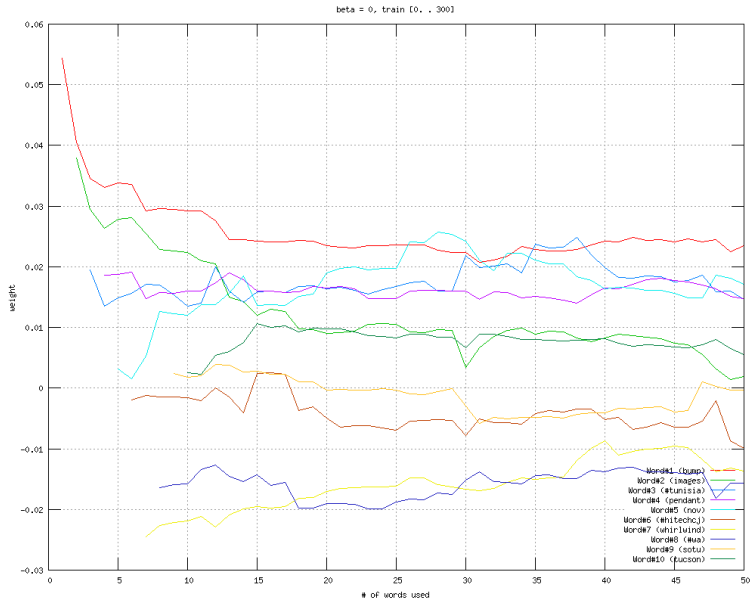

Figure 9: *Aggregate Model Size vs Weights*

Figure 9 shows how the weights of the top 10 words change while we change the aggregate model size from 1 to 50. The x-axis represents the number of words used in aggregate model, and there are 10 curves shown in each plot. For instance, in Figure 9, the top red curve is for the top word 'bump', whose weight is pretty much stablized after about 12 words, while the navy curve at

bottom for the 8-th word '#wa' starts at $x = 8$ (because it is not included in smaller aggregate models) and its weight does not fluctuate much either. Figure 9 was generated with parameters $\beta = 0, t_s = 0, t_e = 300$.

Another way to evaluate our word set is to try to fit the entire time series (both train and test set altogether) usnig the word set chosen from the training set. That is, we select top x words from training set as usual, but we solve the normal equation for the entire period; this implicitly assumes that the aggregate linear regression model is a good model. Figure 10 shows that we can in fact fit the entire time series very nicely; the blue curve (fitting the whole time series) fits the red curve (the target trends) very well, while the green curve (our usual aggregate model) can only fit the training set nicely. This suggests that the way we select the aggregate word set is good, but our model may overfit the training data set which leads to a big error in training set. Here, the paremters were $t_s = 0, t_e = 240, x = 40, \beta = 0$.

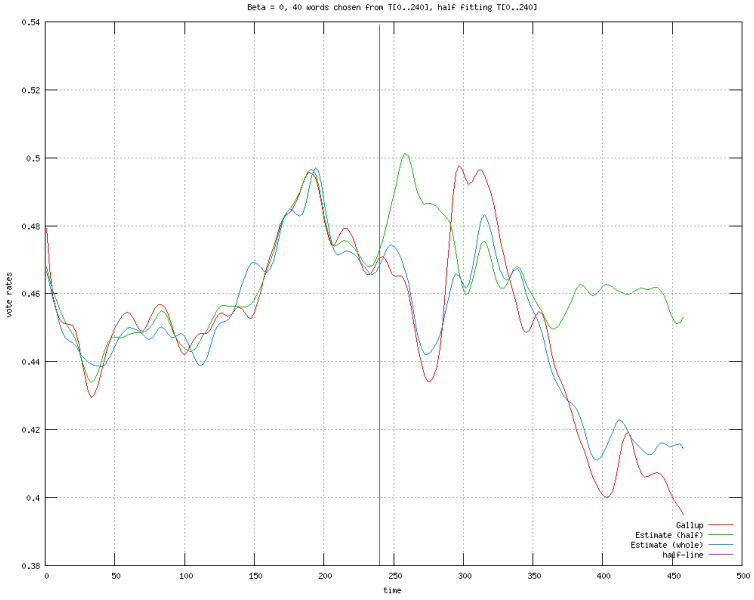

Figure 10: *Fitting Training Set vs Entire Set*

7 Summary and Conclusion

In previous sections, we learned our aggregate models benefit from a larger data set (larger training set) and a larger set of features (more words in model). This is expected and reasonable because our models should collect enough data in order to learn the target trends, and larger set of features helps fitting the target trends.

Section 6.2.3 and Section 6.2.4 confirmed an important fact that our feature selection process via single-word model is a promising process as our aggregate model makes a significantly better prediction that any aggregate model with randomly picked words. We were able to evaluate our model againt the random aggregate models, and confirmed that our model outperforms.

Although we only showed a handful number of plots in this paper, the results were consistent across the parameter space: $\beta \in [0 \dots 30], t_s \in [0 \dots 300], t_e \in [t_s + 50 \dots 504], x \in [1 \dots 100]$. However, for large values of β, our aggregate models started predicting less accurately, and this indicates that predicting the far-future is more difficult than the near-future using Twitter data, which confirms that Twitter data is volatile (which counts for day-to-day trends, in some sense) as the authors in [1] claims in their work, too.

There were of course issues with our models as well. As discussed briefly earlier, our best-performing words in single-word models include junk words that are irrelevant to the topic. To remedy this, our on-going work and future work would be discussed next.

8 Future Work

There are a number of ideas that could improve the performance of our model.

8.1 Natural Language Procesing

In NLP, it is often helpful to use bi-grams (two adjacent words) instead of uni-grams, and our models could easily adopt the bi-gram features as our models do not depend on uni-grams. We are half-way through trying bi-grams; we chose bi-grams such that each word must come from our vocabulary V for the uni-gram model (this ensures that any bi-gram consists of two important uni-gram words), and the most frequent bi-grams are all relevant to our topic, which is different from our uni-gram case. Some of such bi-grams are 'president obama', 'barack obama', 'white house', etc. Recall that in uni-gram model, the most frequent words are 'rt', 'i', 'the', 'to', 'you', etc. in order. Hence, bi-gram model seems to filter our meaningless words automatically, and thus can possibly improve our models significantly. Stemming of words could also help, but we did not seek into this direction.

8.2 Different Trends

A different direction is to predict different trends, such as the popularity vote of American Idol contestants. This is a different problem because it is now a binary classification (i.e. eliminated or stayed). Hence, for the final version, we attempt to implement better learning models and to apply them on at least two different time series data (Obama job approval rate and American Idol contestant popularity).

8.3 More Features

One very useful piece of informatino we explicitly did not use in our model was the user information. For each Tweet, we can identify the user who posted the Tweet, and we could try to add features about the user to improve our model. For instance, if the user mentions 'Obama' in his user profile, we would expect the user to post politically related Tweets more than average users. We believe such extra features could help, but did not pursue this direction in this work.

9 Acknowledgements

This project was jointly used for CS224W, and Professor Leskovec provided guidance on understanding and analyzing experiment results.

References

[1] Bryan R. Routledge Noah A. Smith Brendan O'Connor, Ramnath Balasubramanyan. From tweets to polls: Linking text sentiment to public opinion time series. AAAI Conference on Weblogs and Social Media, 2010.

[2] Rajan S. Patel Lynnette Brammer Mark S. Smolinski Larry Brilliant Jeremy Ginsberg, Matthew H. Mohebbi. Google flu trends. Nature, 2009.