Politwitics: Summarization of political tweets

Erin Summers and Kristin Stephens

May 9, 2012

1 Introduction

Political blogs are a natural opinion space: people more often blog about political opinion, rather than facts. Naturally, a multi-modal opinion space arises between different political opinions and parties. In particular, Twitter provides a unique micro-blogging outlet.

The goal of this project is to summarize the political landscape of Twitter. We cluster the tweets and select the top representatives of each cluster, which provides a more simple and succinct way to get an overall view of the political twitterverse. We also evaluate the top words trending within a particular time window.

2 Background

Twitter is a unique collection of short statement no longer than 140 characters. Because of the concise document size, and the casual audience, the use of slang abounds. In addition to misspellings of words and colloquial slang, it is typical for users to abbreviate words and Internet slang such as "LOL" for "laugh out loud" or less common slang such as "IMO" for "in my opinion." This creates a challenge since it increases the vocabulary.

Multi document summarization (MDS) is a type of news aggregation and is used to reduce a large corpus of documents into a concise summary, which expresses the key topics within the entire corpus [2], [1], [3], [4], [5], [6], [7], [8]. The advantage of employing MDS is that a person can quickly ascertain the key points within a long list of documents, rather than sift through the whole corpus.

The MDS algorithms work by first discovering the top hidden topics within the corpus. Next, the distance of each sentence from the top topics is measured using various heuristics. A summary is constructed by piecing together the most relevant whole sentences for each topic.

MDS is typically applied to a corpus of long documents, rather than short tweets. Additionally, the typical input documents are assumed to be standard, coherent texts. Because tweets are so short and noisy, we stray from the standard implementation of MDS and do not offer full paragraph summaries, but rather tweet-sized summaries of the corpus.

3 Data Collection and Preprocessing

We used Twitter4J [11] to create a twitter filter for a list of keywords including: candidate names, political words, and hot topics. See table 1 for the keywords we used. Unfortunately Twitter4J's API uses OR logic when a filter keyword contains multiple words. This requires us to reprocess all the tweets we receive from the filter confirming the entire multi-word keyword is present in the tweet.

After removing irrelevant tweets, we process the resulting tweets using Ruby to create a dictionary and bag of words vector for each tweet. We create two dictionaries using the top words based on word frequency. The word frequency in the first dictionary is simply word count. The second dictionary's word frequency is a weighted count based on the age of the tweets the words came from. The younger the tweet the more weight it's words have in the count for the word frequency. We used equation 1 to calculate how much the words in a given tweet counted towards the word's frequency in the dictionary.

$$\frac{\text{tweet timestamp} - \text{start time}}{\text{total timeframe}} \tag{1}$$

For example, the youngest tweet's word would add closest to 1, while the words in a tweet halfway through the timeframe would add $\frac{1}{2}$.

Political Candidates	Political Words	Hot Topics
Bill Clinton	caucus	abortion
Dick Cheney	democrat	birth control
George Bush	election	death penalty
Herman Cain	libertarian	gay adoption
Hilary Clinton	republican	gay marriage
Hillary Clinton	rockthevote	gay rights
Joe Biden	vice president	gun control
John Boehner		roe wade
John McCain		
Mitt Romeny		
Nancy Pelosi		
Newt Gingrich		
Obama Barack		
Ron Paul		
Santorum		
Sara Palin		

Table 1: List of keywords used in twitter filter

After creating the dictionary we choose the size of the dictionary and therefore the size of the resulting sparse matrix. Each tweet's bag of word vector is created using the chosen dictionary and the sparse matrix output file is in binary for fastest processing when we read it into Matlab.

Our data spans from March 29th to the present day, since Twitter4J is continuously running. A histogram of the word count vs. word frequency in Figure 1 shows that the word count follows a Zipf logarithmic distribution. Total number of tweets (after preprocessing for relevancy) are shown in Figure 2. There is a spike in the number of tweets around 4/10. After collecting a few weeks worth of tweets, we noticed that the Twitter4J API was using OR logic on keywords and removed some of the bigrams from our keyword list such as "tea party". Thus, we were able to collect more tweets since we were previously hitting the %1 filter stream maximum. All together, we have collected 6.2 million relevant tweets.

4 Analysis

Using term-frequency inverse-document-frequency (TF-IDF), we observe the top words within a collection of tweets. We employ the clustering algorithms, k-means, fuzzy c-means, and non-negative matrix factorization (NMF).

The bag-of-words representation of the data is $X \in \mathbb{R}_+^{|T|\times|V|}$, where $|T|$ is the number of tweets and $|V|$ is the size of the dictionary. In each clustering method, the number of clusters, C is chosen a-priori.

4.1 TF-IDF

Term-frequency, inverse-document frequency provides a way to adjust the weight of the words in a collection of documents based on the relevancy of the word. The TF-IDF score gives less weight to words that occur in other document and gives more weight to words which are repeated often within a document. The term frequency tf is simply the number of times the word occurs in a particular document. The inverse document frequency idf and $tf - idf$ are given below:

$$idf(w,T) = \log\left(\frac{\text{size of corpus}}{\#\text{ of tweets where word } w \text{ appears}}\right), \quad (2)$$

$$tfidf(w,T) = tf(w,T)idf(w,T). \quad (3)$$

Using TF-IDF, we can access top words in the documents and whether or not any of these words are n-grams. TF-IDF replaces the word counts in a tweet with scalings based on how relevant that word is to the tweet within the corpus.

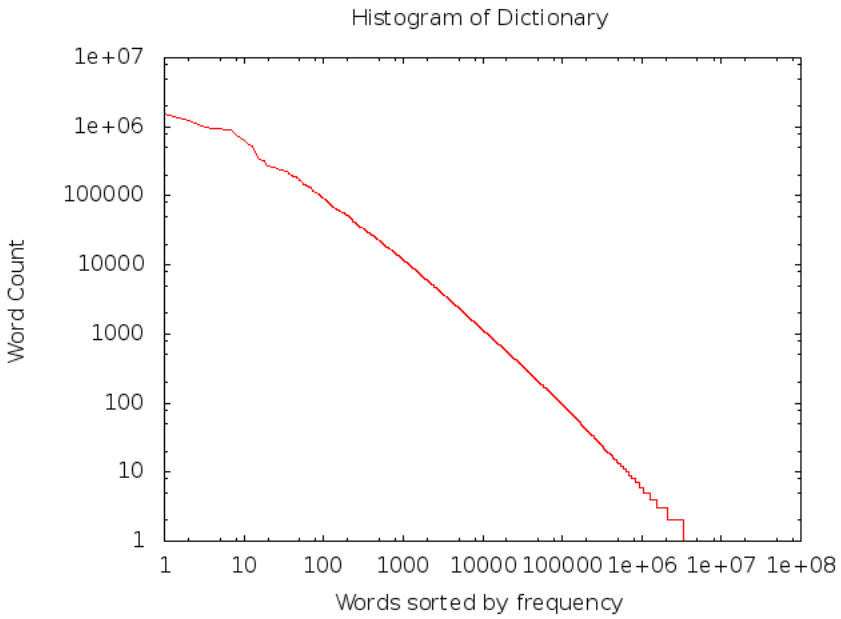

Figure 1: Word Count Histogram follows a Zipf Distribution

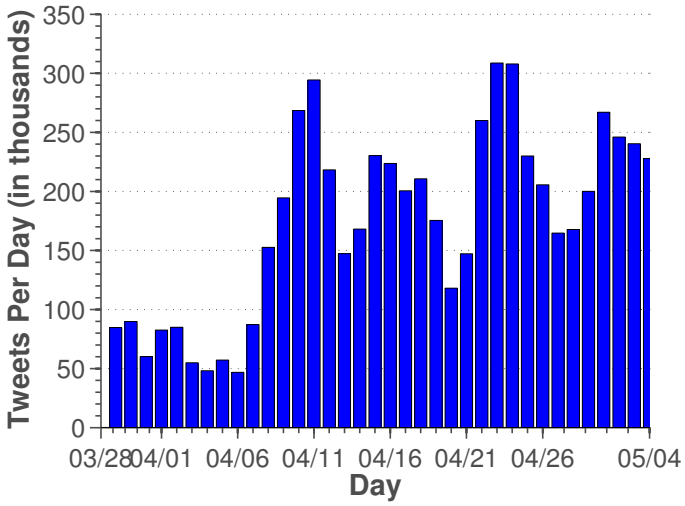

Figure 2: Tweets Per Day from 03/29/2012 to 05/04/2012

4.2 K-means

K-means is a simple, nondeterministic clustering process, which maximizes the similarity between the cluster and the documents in the cluster.

$$\sum_{k=1}^{C} \sum_{i \in c_k} \frac{x_i \cdot c_k}{|x_i||c_k|}, \tag{4}$$

where x_i is the *ith* tweet and c_k is the *kth* cluster. We use a cosine similarity measure, rather than a distance measure.

Two major advantages of k-means are that it is simple to implement and computationally cheap, since at worst it involves inner products of vectors. The standard implementation of k-means forms hard clusters, meaning that each tweet belongs to only one cluster. This is a disadvantage since it does not allow for partial membership of multiple clusters. Many tweets involve opinions comparing one candidate to another. Hence, if the clusters formed represented singular candidates, such a tweet should have membership in multiple clusters.

Data: $X \in \mathbb{R}^{|T| \times |V|}$, bag of words representation of tweets
initialization: randomly select k tweets as the initial clusters;
while *labels*$_{i-l}$ *!= labels*$_i$ **do**
  label$_i \leftarrow$ argmax(cossim(tweet$_j$, cluster$_k$));
  cluster$_k \leftarrow$ avg(tweet$_j$ in cluster$_k$);
end

Algorithm 1: k-means

4.3 Fuzzy C-means

Fuzzy c-means [10], also known as soft k-means, is similar to k-means, except it is a soft-clustering process and allows tweets membership in multiple clusters. The objective is to minimize the cost

$$\sum_{i=1}^{|T|} \sum_{k=1}^{C} u_{ik}^{m} \left(1 - \frac{x_i \cdot c_k}{|x_i||c_k|}\right), \ 1 < m < \infty, \tag{5}$$

where U is the partition matrix and m is the fuzzifier. The choice of m is critical. In the limit $m = 1$, c-means behaves as a hard clustering. In our case, we chose $m = 3.2$ in order to get convergence.

Data: $X \in \mathbb{R}^{|T| \times |V|}$, bag of words representation of tweets
initialization: randomly initialize U such that $\sum_i U_{ij} = 1 \ \forall j$;
while *tolerance not reached* **do**
  $c_k = \frac{U^m}{\sum_{w \in V} c_{k_w}}$;
  $D_{kj} = 1 - \text{cossim}(t_j, c_k)$;
  $U_j = \frac{D^{\frac{-2}{m-1}}}{\sum_i D_{ij}}$
end

Algorithm 2: Fuzzy c-means

4.4 Non-negative Matrix Factorization (NMF)

Non-negative Matrix Factorization (NMF), [9] seeks to find a subspace of X such that the objective cost

$$\|X - WH\|_2^2 \tag{6}$$

is minimized, for $W \in \mathbb{R}^{|T| \times C} \geq 0$ element-wise and $H \in \mathbb{R}^{C \times |V|} \geq 0$ element-wise. Like fuzzy c-means, NMF is a soft-clustering method.

Initialization for NMF is tricky. It is possible that the products in the denominator of the H and W updates are zero for some entires, which leads to division by zero errors. On solution is to initialize H and W to be full, random matrices, but this is too expensive computationally. Another solution is to initialize H and W to be sparse, random matrices with a density much higher than X. We found that using at least a density of 0.2 yields convergence.

obama	0.93	rt	0.33
president	0.62	republican	0.33
romney	0.41	gay	0.32
barack	0.40	clinton	0.32
mitt	0.40	via	0.32
campaign	0.36	#obama2012	0.30
paul	0.35	control	0.30
ron	0.34	abortion	0.30
people	0.33	@barackobama	0.30
first	0.33	election	0.29

Table 2: Top 20 words from TF-IDF words and scores

Data: $X \in \mathbb{R}^{|T| \times |V|}$, bag of words representation of tweets
initialization: randomly initialize W and H to sparse random matrices of a certain density ;
while *tolerance not reached* **do**
  $w_{ik} = w_{ik} \frac{(XH^T)_{ik}}{(WHH^T)_{ik}}$;
  $h_{kj} = h_{kj} \frac{(WX^T)_{kj}}{(W^TWH)_{kj}}$;
end

Algorithm 3: NMF

5 Results

The goal of this project is to retweet relevant tweets. As the political tides are ever-changing, we focus on analysis of a particular day, namely May 5th 2012, which contains 150,000 tweets. Preliminary analysis revealed that bi-grams were not present in the top words from TF-IDF. Thus, we built a dictionary using only unigrams and removed stopwords from the dictionary. Using a weighted dictionary only affected the tweets over a small time window. For this study, we use an unweighted dictionary on the snapshot of tweets. We limit the size of our dictionary to 1000.

After running TF-IDF, the twenty highest weighted words on 05/05/2012 are in Table 2 **obama, president, romney, barack, mitt, campaign, paul, ron, people, first, rt, republican, gay, clinton, via, #obama2012, control, abortion, @barackobama, election**.

Though many of the top words from TF-IDF were also keywords for our filter, TF-IDF was not only able to discover the most relevant keywords, but included others as well, such as the #obama2012 tag. Compared to an earlier analysis, candidates that had dropped out like Rick Santorum are notably missing from this list. Now that most of the Republican contenders have dropped out, the conversations appear to be more centered around Obama, Romney and Ron Paul, who recently won a primary in Maine, which generated a lot of tweet traffic.

Results from k-means, fuzzy c-means and NMF cluster are in Tables 3-5. Each method divided the tweets into six clusters. Main topics for the k-means clusters are Barack Obama, Ron Paul, Mitt Romney, gay marriage, with a few noisy clusters. The c-means iteration appears to have centered around only Barack Obama and Ron Paul, with many Barack Obama clusters. NMF clustering performs poorly, perhaps due to issues balancing computational intensity and initialization. Of the three methods, k-means performs the best. The number of tweets in each cluster is shown in Figure 3.

C-means also did not perform as well as k-means. This can be attributed to the choice of the fuzzier m. In our implementation, if m is too small, then there is no convergence. However, if m is too large, then U becomes ill-posed and complex.

In both cases of soft clustering, none of the medoids had membership in clusters other than their strongest cluster, which is understandable. It is possible that the algorithms would create clusters that are soley a mix of two topics. However, the results indicated that the clusters were somewhat homogeneous, particularly in NMF.

6 Lessons Learned

The biggest lesson we learned is Twitter data is extremely noisy. Even when our keywords are matched perfectly many of the tweets are not of very high quality, ranging from spam to inappropriate. To add to

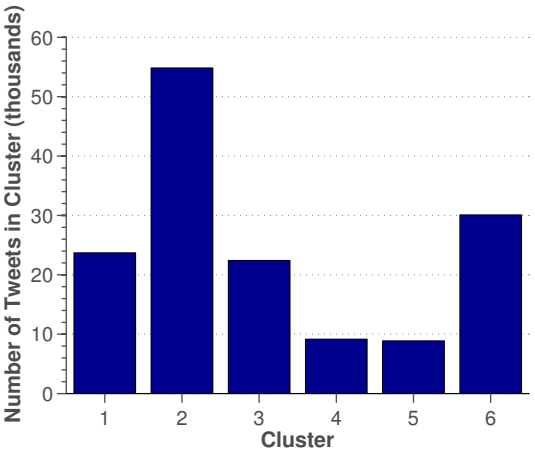

Figure 3: Number of tweets in each cluster, Kmeans

cluster 1	barack obama mask: .x{color:#83c22d;margin: 0px;font-size :12px}.y{color:#a56eba}barack obama mask political masks (... http://t.co/1s8m823y
cluster 2	@dykstradame ron paul is his very own brand of republican nightmare: racist,hypocrite, homophobe and laissez-faire ana ...
cluster 3	"dogs against romney"...dogs against romney super pack member jeff clegg protested mitt romney in idaho falls, id... http://t.co/udh5inwi
cluster 4	clearly, we can say & affirm that gay marriage is a most vicious assault on the beliefs of christians, muslims and jews!
cluster 5	romney mulai kejar obama dalam jajak pendapat: sebuah jajak pendapat nasional terbaru menunjukkan calon presiden... http://t.co/xnjtzurd
cluster 6	charlie rose - vitaly churkin / historians on president obama (april 8, 2010): president obama and president me... http://t.co/stswgns7

Table 3: K means clusters

cluster 1	proof that romney is crazier than i am: http://t.co/2lblibyd
cluster 2	inilah pacar pertama barack obama http://t.co/cbhig65o
cluster 3	checking out "muslim brotherhood invades tampa to re-elect president obama!" on constitutional emergency: http://t.co/msrup7ep
cluster 4	#airjordan #barackobama hello mister president http://t.co/msccg7ye
cluster 5	"i love you back."president obama
cluster 6	ron paul is benefiting the least from super pacs among ... http://t.co/fobk0qqi

Table 4: Fuzzy C means clusters

cluster 1	herbert hoover accepts republican nomination, nbc,c1932: 8x12in print from a high-quality scan of the original.ti... http://t.co/hzabrfsg
cluster 2	tebow, lin, messi, adele, pippa, barack, etc... #time http://t.co/mazfsegi
cluster 3	@wawinburn romney! romney! romney!
cluster 4	#ron paul videohttp://t.co/84tvw3ip
cluster 5	endorse @newtgingrich for president with @hlm25933 on #votizen - https://t.co/oht8cd39
cluster 6	people, people, thats not a giant moon! its big daddy barack's ego.cresting over the horizon on the evening of marx birthday. @barry_o44

Table 5: NMF clusters

the twitter noisy, Twitter4J uses OR logic when a filter keyword has multiple words. This means when we filter for tweets with two words we receive all tweets with one OR the other word (e.g. "tea party" returns all tweets with "tea" or "party").

We learned that taking into account the age of the tweet has a larger impact if the dictionary is built over a smaller window of time. Since the scope of the project is to look at a snapshots of tweets over a day or two, it is not clear what the affect of using a weighted dictionary over a small range. Premilinary results showed that the words from TF-IDF were different between two dictionaries that are weighted and unweighted and built over 2 days worth of tweets. However, one did not appear to be more relevant than the other.

K-means returned the most hetergeneous and relevant clusters, fuzzy c-means clusters were relevant, but homogeneous, and NMF clusters appeared to be somewhat random. In fuzzy c-means, the choice of the fuzzier m is critical and with NMF the initialization of W and H is tricky. Though these algorithms may perform well on large data that is less noisy in the case of twitter the simpler k-means appears to be the best choice.

7 Future Work

For future work, we would like to first, limit the scope of the keywords to a few candidates names. This will reduce unexpected noise and spam significantly. We would also like to build the dictionary based on a trusted, political source. By building a classifier from a current, political blog, we can use mutual information to select the words in the dictionary for twitter.

The performance of the soft-clustering methods was shockingly bad. We would like to investigate reasons why these clustering algorithms performed poorly and investigate other clustering methods. We would also like to utilize the weighted dictionary and better understand the effects of the weighted dictionary on the clustering results.

Our ultimate goal was to retweet the top tweets to our twitter handle `@politwitics`. With the general election on the horizon, we hope to have this system fully running this summer.

References

[1] Inouye, D. and Kalita, J. "Comparing Twitter Summarization Algorithms for Multiple Posts",IEEE Conf on Privacy, Security, Risk, and Trust. pp. 298–306. 2011.

[2] L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova, "Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion," Information Processing & Management, vol. 43, no. 6, pp. 16061618, 2007.

[3] D. Radev, S. Blair-Goldensohn, and Z. Zhang, "Experiments in single and multi-document summarization using mead," DUC-01, vol. 1001, p.48109, 2001.

[4] G. Erkan and D. Radev, "Lexrank: graph-based centrality as salience in text summarization," Journal of Artificial Intelligence Research, vol. 22, pp. 457480, 2004.

[5] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," inEMNLP. Barcelona: ACL, 2004, pp. 404411.

[6] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine* 1," Computer networks and ISDN systems, vol. 30, no. 1-7, pp. 107117, 1998.

[7] Blei, D.M. and Ng, A.Y. and Jordan, M.I. "Latent Dirichlet Allocation". Journal of Machine Learning Research. pp 993–1022. 2003

[8] Canny, J. "GaP: A Factor Model for Discrete Data". Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval. 122-129. 2004.

[9] Lee, D.D. and Seung, H.S. "Learning the parts of objects by non-negative matrix factorization". Nature. 788–791. 1999.

[10] Bezdek, J.C. and Ehrlich, R., "FCM: The fuzzy c-means clustering algorithm". Computers & Geosciences. 191–203. 1984.

[11] Twitter4J, “Twitter4J - A Java library for the Twitter API”. May 2012. URL http://twitter4j.org/en/index.html