Analyzing 2016 US President Election with Author-Topic Model

Yicheng Pu
yp653@nyu.edu

Tianyu Wang
tw1682@nyu.edu

Abstract

US president election has always been an hot research topic in social and political science, these days many researches attempt to combine machine learning and Twitter data to solve this task. However, most of works focused on supervised learning and activities around specific candidates. We proposed an unsupervised method uses Author-Topic Modeling to extract topics of each states' tweets, then make predictions based on states' topic distributions. Our methods achieved accuracy approaching human expert's and outperformed supervised classification of FastText.

1 Introduction

Twitter has been widely used as a source of public opinions. Thus with Twitter data, it is possible to gain some insights of the US Presidential election, whose results are strongly affected by public opinion. Besides from that, another important factor of the election is geolocation, since the US presidential election is voted by electors from each state.

Most works to date has focused on sentiment analysis of tweets, which could provide indications of changes in opinion for each specific candidate. However, we expected to uncover the hidden variation of people's opinions on everything, instead of getting limited in specific candidate. The biggest obstacle to approach the problem is the insufficiency of labeled data. Hence in this paper we chose an unsupervised learning method – topic modeling to solve this issues.

Topic modeling has been a classical tool for analyzing opinions in text, but traditional LDA[1] doesn't include geographical factor. In this paper we chose the Author-Topic model(ATM)[2] in which each author is associated with a distribution of topics, but we replaced author with states. We would like to compare topic distributions over different states during 2016 election, and to test whether their variations reflect political leanings. As a comparison, we also used supervised FastText[3] model to predict twitter users' political leaning, and aggregate predictions for single users into predictions for states.

2 Related Work

The tremendous amount of active users in Twitter has contributed a lot of information. A number of attempts have been made to infer the popular attitudes with mainstream information instead of collecting public opinion queries. In 2010, Brendan had explored the correlation between sentiment word frequencies and political opinion. [4] After that, Kazem enhanced Brendan study by introducing geographic tagged, and uncovered candidates popularities locally. Meanwhile, they had claimed the validation of predicting elections by using Twitter data.[5] Following works got better performance as the nature language processing techniques improve. [6][7]

3 Model

3.1 ATM

Traditional LDA model assumes that a document is composed of different topics, and each topic has a specific distribution of related words. It simulates the process of document generation by firstly picking topics from their prior distribution, then picking words according to the picked topic.

Here the Author-Topic Model(ATM) adds another latent variable a, which represents authors, into the original LDA model. And it adds another assumption that the authors of each document would also affect the topic assignment.

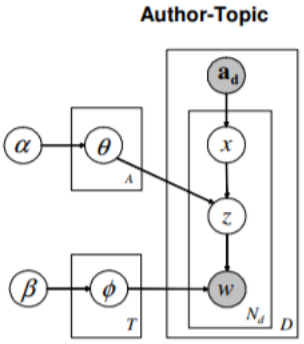

Figure 1: ATM

a_d: Authors for each document

x: Author of a given word

$\theta \sim \tilde{D}irichlet(\alpha)$: Distribution over topics for each author

z: Topic assignment for each word

$\phi \sim Dirichlet(\beta)$: Word distribution for each topic w: Word

To explain the generative process more specifically, once a list of authors a_d are given, we uniformly select an author x from a_d. In our case, tweets only have a single user, so this step could be omitted. Then we choose a topic z from x's specific topic distribution θ_a generated from a Dirichlet prior. With z we could generate word w from z's specific word distribution ϕ_t, which was also generated from a Dirichlet prior. This model is illustrated in Figure 1.

After the generative model is set up, we infer the optimal parameters from their distributions. Here we used Gibbs Sampling[8], a relatively simple method which constructs a markov chain to approximate parameters and updates it by counting. This inference part is well-described in the third section of original paper[2] so we will skip it here.

3.2 FastText

FastText[3] is an efficient and powerful text classification model, which has been widely used in text classification.

The mechanism of FastText is fairly simple as shown in figure 2. Given a list of words from a document, we encode each word as a one hot vector w_i whose length is vocabulary length and only the element at the words index is set to 1. Then we build an embedding Π with size of $(vocabularysize, embeddingsize)$. The dot product of w_i Π gives us the word embedding v_i. To get the document vector, we simply take the mean of v_i for each word index i in the document. After that, the vector is sent to a linear layer to get a final classification score.

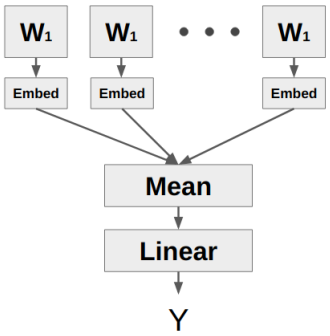

Figure 2: FastText

4 Experiments

4.1 Data

The dataset is provided by Harvard[9], it contains tweets collected during 3 presidential debates and election day of 2016 election. Since training ATM is highly time-consuming, and larger data size doesn't necessarily brings higher performance, we sampled 10% data from the original dataset. We also observed that the tweet volume is highly imbalanced over states, as shwon in figure 3.

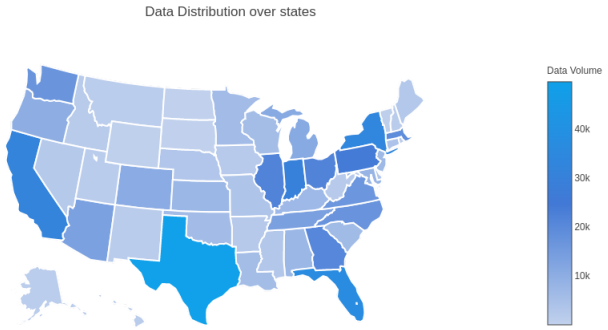

Figure 3: Data Volumes over states

4.2 Data Cleaning

The raw data is pretty unorganized and noisy, hence we applied multiple steps of data cleaning on it. The very first step is to get users locations, most twitter users add their location in their profile page, but the format is uncertain. We only kept the format of (states,city) or (states) or (USA,states). Then we removed user with more than 2000 followers to avoid celebrities opinions, since they could not be counted as a good example of local opinions. User followed more than 2000 other users have also been removed since they are likely to be bots. We also removed users who tweeted more than 5 times to filter out strong personal opinions.

For tweets, we only kept English tweets and converted all text into lower cases. Then for each tweet, we removed common English stopwords, all numbers and punctuations , all words with extremely low frequency(<3) and extremely higher frequency(>200). Extremely short words with number of letters less than 3 are also removed, we didn't remove any long text since they could be twitter username mentioned in tweets and would be helpful for extracting topics. After data cleaning, the number of tweets shrinked to 175,313.

4.3 Author-Topic Model

4.3.1 Perplexity

Before applying ATM to ranking prediction, we assured that our model has successfully learned the latent distribution by introducing perplexity to evaluate. Perplexity measures the difference between model's word distribution with real word distribution, a lower Perplexity indicate better capacity of the model.

$$perplexity(w_d|a) = exp(-\frac{lnp(w_d|a_d)}{N_d})$$

We fed tweets and geographic labels into ATM, then tested the performance of number of topics for Perplexity. As shown in 4, the number of topics strongly affects Perplexity, increasing topics generally helps decreasing perplexity, but the curve converged at around 300 topics.

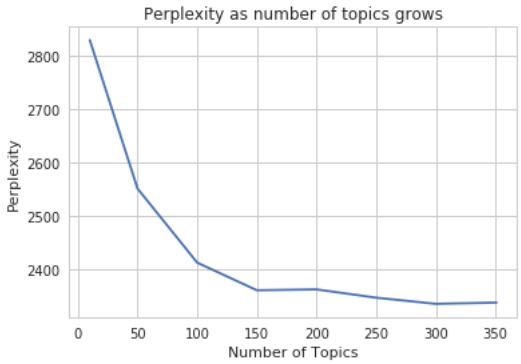

Figure 4: PP

4.3.2 Ranking Evaluation

We believe that there exists a gap between twitter users' opinion distribution and real opinion distribution, thus predicting support rate directly might need other supplemental data, and not feasible for just using twitter text data. Alternatively, we predicted the ranking of GOP favorability. GOP favorability of swing states, then evaluated the ranking with the following formula:

$$ACC = \frac{1}{\Sigma_{i=1}^{11}\Sigma_{j=1,j\neq i}^{11}1}(\Sigma_{i=1}^{11}\Sigma_{j=1,j\neq i}^{11}1(r_i > r_j) = 1(\tilde{r_i} > \tilde{r_j}))$$

where r_i is the predicted rank of state i, r_j is the real rank of state i. This function compares states' ranks pairwisely with real ranks.

4.3.3 Ranking Methods

After we got the 300 topics for each state, we firstly divided the topic distribution of each state by their sum to get the normalized author-topic distribution matrix AT with shape (51,300), then extracted two hand-crafted features:

1. Republican topics T_{red}: topics that have scores larger than mean+0.05*standard deviation in safe Republican states, but have scores lower than mean-0.05*standard deviation in safe Democrat states.
2. Democrat topics T_{blue}: topics that have scores larger than mean+0.05*standard deviation in safe Democrat states, but have scores lower than mean-0.05*standard deviation in safe Republican states.

Then we applied 12 different methods to predict swing states' ranking, to illustrate them well, we have to make some notations first.

We denote Republican safe states' author topic distribution by $AT_{rep,:}$, and Democrat safe states' by $AT_{dem,:}$, swing states by $AT_{swing,:}$

We also denote three main methodologies: f_{rf},f_{NN} and $f_{\sim NN}$:

1. $f_{rf}(X_{train}, X_{test})$ is a random forest classifier, which is trained on X_{train}, then predict X_{test} and rank the states in X_{test} by their predicted probability score.
2. $f_{NN}(target, X_{test})$ ranks states in X_{test} by their euclidean distance from the mean of target.
3. $f_{NN}(target, X_{test})$ ranks states in X_{test} by their negative euclidean distance from the mean of target.

Here are our 12 methods and their abbreviation:

1. 'rf all topics ':$f_{rf}((AT_{rep,:}, AT_{dem,:}), AT_{swing,:})$
2. 'NN red all topics ':$f_{NN}(AT_{rep,:}, AT_{swing,:})$
3. 'NN blue all topics ': $f_{\sim NN}(AT_{dem,:}, AT_{swing,:})$
4. 'rf union topics ': $f_{rf}((AT_{rep,T_{red}\cup T_{blue}}, AT_{dem,T_{red}\cup T_{blue}}), AT_{swing,T_{red}\cup T_{blue}})$
5. 'NN red union topics ': $f_{NN}(AT_{rep,T_{red}\cup T_{blue}}, AT_{swing,T_{red}\cup T_{blue}})$
6. 'NN blue union topics ': $f_{\sim NN}(AT_{dem,T_{red}\cup T_{blue}}, AT_{swing,T_{red}\cup T_{blue}})$
7. 'rf red topics ': $f_{rf}((AT_{rep,T_{red}}, AT_{dem,T_{red}}), AT_{swing,T_{red}})$
8. 'NN red red topics ': $f_{NN}(AT_{rep,T_{red}}, AT_{swing,T_{red}})$
9. 'NN blue red topics ': $f_{\sim NN}(AT_{dem,T_{red}}, AT_{swing,T_{red}})$
10. 'rf blue topics ': $f_{rf}((AT_{rep,T_{blue}}, AT_{dem,T_{blue}}), AT_{swing,T_{blue}})$
11. 'NN red blue topics ': $f_{NN}(AT_{rep,T_{blue}}, AT_{swing,T_{blue}})$
12. 'NN blue blue topics ': $f_{\sim NN}(AT_{dem,T_{blue}}, AT_{swing,T_{blue}})$

4.3.4 Topics

After we determined topic numbers and ranking methods, we trained ATM on our data then investigated the topic distribution over different states and political leanings.

As shown in figure 4.3.4, the topic distribution between different states (here we used Wyoming and Washington D.C. as an example) are very distinguishing. We compared the mean of all safe republican states with mean of all safe Democrat states, and this averaged distribution showed strong divergence as well.

To filter topics with high noise or consistency in both groups, we hand-crafted the feature extraction process described in last section to get high-quality Republican and Democrat topics, here shown in figure 4.3.4.

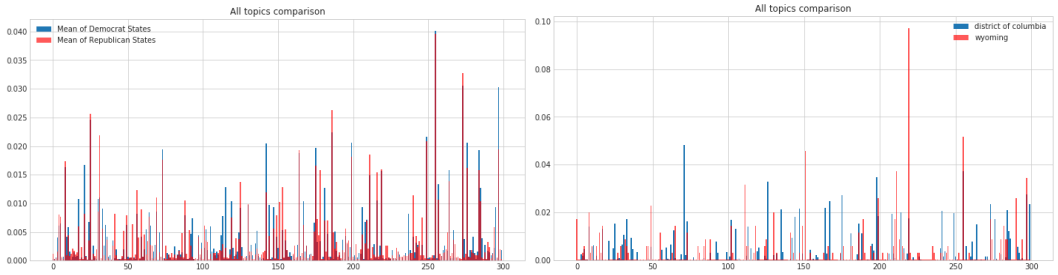

Figure 5: All topics

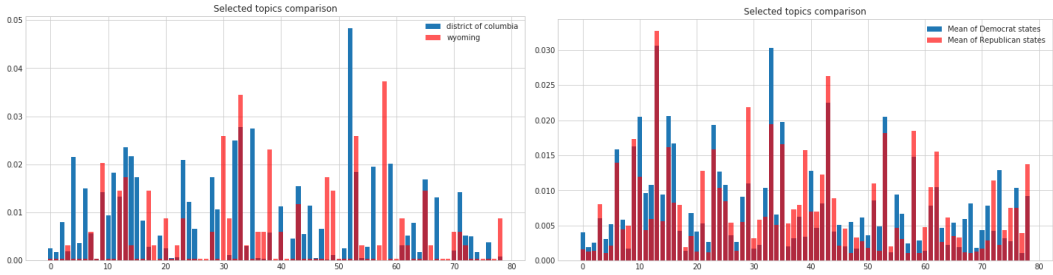

Figure 6: Selected Topics

In figure 7, we present some examples of selected Republican topics and Democrat topics.

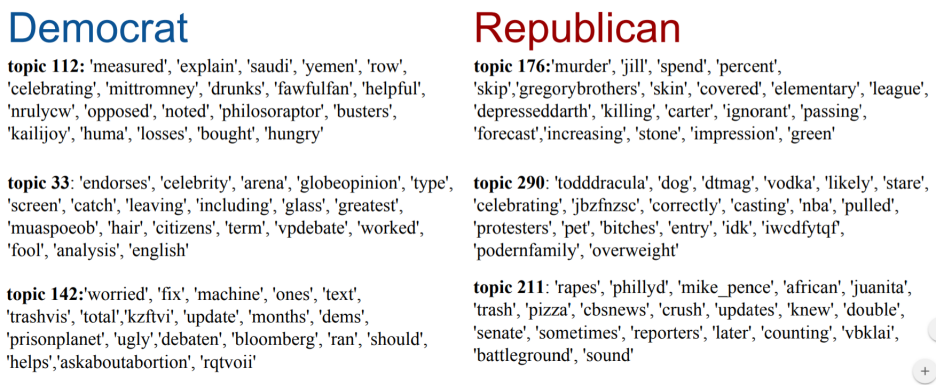

Figure 7: Topics

4.4 Results

Similar to LDA, ATM generated topic distribution is unstable. To evaluate its performance more precisely, we experimented different number of training tweets, ranging from 60,000 to 170,000, and replicated each experiment 5 times. Final results are calculated by their mean accuracy.

We also tuned FastText with different hyper-parameter settings, and the results are shown in table 1. The best result of FastText is 0.6.

N-gram	Epochs	Training loss	Ranking ACC
1	20	0.18	0.57
1	25	0.17	0.59
1	30	0.11	0.54
2	20	0.16	0.59
2	25	0.09	0.59
2	30	0.11	0.57
3	20	0.14	0.59
3	25	0.10	0.57
3	30	0.1	0.60

Table 1: Table of FastText results

The final result for ATM is shown in table 8, we also plotted line plots 9 and bar plots 10 to better illustrate this table with many attributes.

n_doc	n_topic	n_vocab	max_iter	rf all topics	NN red all topics	NN blue all topics	rf union topics	NN red union topics	NN blue union topics	rf red topics	NN red red topics	NN blue red topics	rf blue topics	NN red blue topics	NN blue blue topics
62800.0	300.0	13140.0	30.0	0.626446	0.626446	0.547107	0.685950	0.633058	0.530579	0.689256	0.626446	0.487603	0.570248	0.517355	0.583471
73730.0	300.0	14542.0	30.0	0.619835	0.606612	0.580165	0.685950	0.573554	0.560331	0.656198	0.649587	0.500826	0.580165	0.494215	0.593388
90029.0	300.0	16771.0	30.0	0.629752	0.586777	0.593388	0.682645	0.593388	0.566942	0.656198	0.672727	0.438017	0.685950	0.451240	0.659504
117729.0	300.0	20269.0	30.0	0.682645	0.709091	0.537190	0.719008	0.712397	0.497521	0.728926	0.738843	0.411570	0.676033	0.550413	0.586777
175313.0	300.0	27390.0	30.0	0.666116	0.702479	0.563636	0.702479	0.699174	0.504132	0.672727	0.719008	0.365289	0.705785	0.487603	0.639669

We used prediction form human expert J. Scott Armstrong[10], who correctly predicted every election since 2004 to 2016 within tiny prediction error, as a reference.

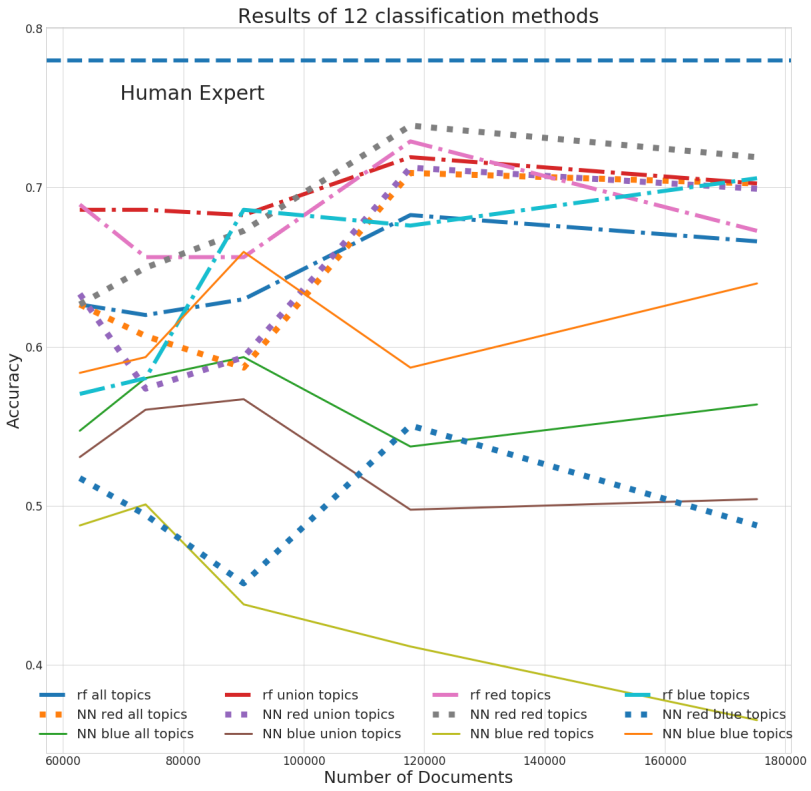

Figure 9: ATM predictions in lines

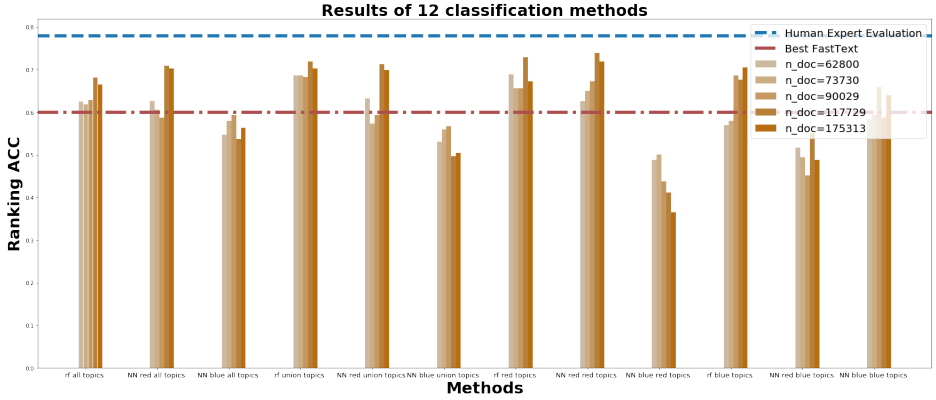

Figure 10: ATM predictions in histogram

From figure 10, we could see that among the 12 methods, most of them successfully beat the FastText baseline, indicates that unsupervised Author-Topic Model actually has strong prediction power. Also these methods' performance generally gets better as number of documents increases. The best method's best performance is around 0.75, which closely approaches human expert's 0.78 accuracy. Even the most basic methods, such as applying a random forest or nearest neighbor on all topics, got an accuracy around 0.7 when the data size is large enough.

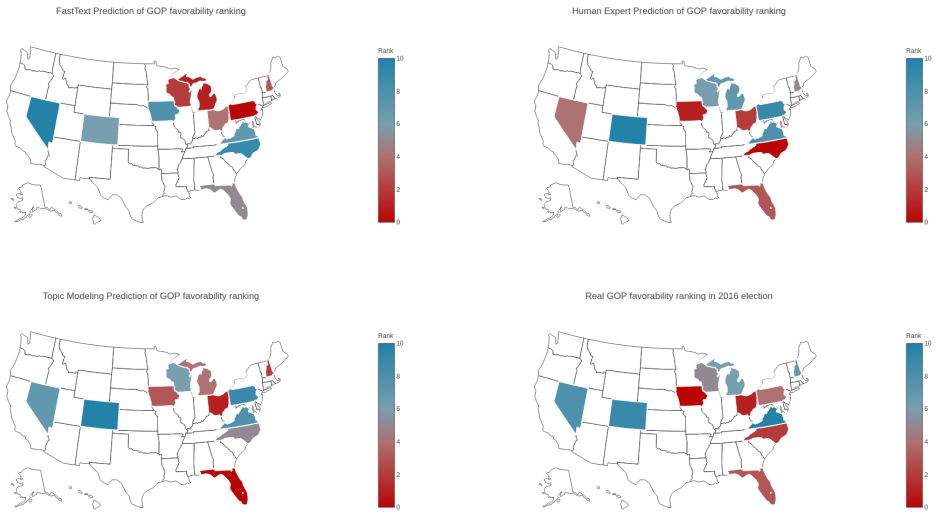

Figure 11: Predicted Ranking

We also visualized our prediction on map to illustrate it more intuitively, as shown in figure 4.4. Our prediction captured general political leaning of big states like Florida, Iowa and Ohio, but lacks the prediction power on states with small population like New Hampshire, because of the strongly imbalanced data distribution.

5 Conclusion

In conclusion, we found that Author-Topic Model worked pretty well on this task, with stable performance approaching human expert's prediction. Usually supervised methods should be able to get good results when there is enough high-quality labeled data. In our case, we do not have enough labeled data, thus FastText didn't get ideal performance. But ATM model doesn't rely on any labeled data as an unsupervised model, and we could also evaluate its capacity through perplexity before the final prediction task, which makes it more controllable.

6 Future Work

The biggest problem of applying machine learning models in this task is the lack of testing instances (election result). We could only get Twitter data back to 2008 election for the most, implying that only 3 potential testing instances: 2008 election, 2012 election and 2016 election are available for our use. This strictly limits us from tuning hyper-parameters, since the model would be hard to generalize on other election results. In the future, as more and more data will be recorded, this method could reach better performance.

Another factor we could explore with topic modeling is time, there are many topic models that include time as another latent variable, like Dynamic Topic Modeling [11]. Since we have tweets from 3 president debates to final election, analyzing the variation of topics over time should also be interesting.

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press, 2004.

[3] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[4] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, Noah A Smith, et al. From tweets to polls: Linking text sentiment to public opinion time series. *Icwsm*, 11(122-129):1–2, 2010.

[5] Kazem Jahanbakhsh and Yumi Moon. The predictive power of social media: on the predictability of us presidential elections using twitter. *arXiv preprint arXiv:1407.0622*, 2014.

[6] Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120. Association for Computational Linguistics, 2012.

[7] Elvyna Tunggawan and Yustinus Eko Soelistio. And the winner is. . . : Bayesian twitter-based prediction on 2016 us presidential election. In *Computer, Control, Informatics and its Applications (IC3INA), 2016 International Conference on*, pages 33–37. IEEE, 2016.

[8] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[9] Justin Littman, Laura Wrubel, and Daniel Kerchner. 2016 united states presidential election tweet ids, 2016.

[10] John Q Wofford Jonathan Ortiz Mario D'Amore Noah Rippner Brian Kelsey, Jeffrey Shih. 2016 us presidential election, 2016.

[11] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.