Compass: Spatio Temporal Sentiment Analysis of US Election

What Twitter Says!

Debjyoti Paul, Feifei Li, Murali Krishna Teja, Xin Yu, Richie Frost
University of Utah
{deb,lifeifei,teja614,xiny,rfrost}@cs.utah.edu

ABSTRACT

With the widespread growth of various social network tools and platforms, analyzing and understanding societal response and crowd reaction to important and emerging social issues and events through social media data is increasingly an important problem. However, there are numerous challenges towards realizing this goal effectively and efficiently, due to the unstructured and noisy nature of social media data. The large volume of the underlying data also presents a fundamental challenge. Furthermore, in many application scenarios, it is often interesting, and in some cases critical, to discover patterns and trends based on geographical and/or temporal partitions, and keep track of how they will change overtime.

This brings up the interesting problem of *spatio-temporal sentiment analysis* from large-scale social media data. This paper investigates this problem through a data science project called "US Election 2016, What Twitter Says". The objective is to discover sentiment on Twitter towards either the democratic or the republican party at US county and state levels over any arbitrary temporal intervals, using a large collection of geotagged tweets from a period of 6 months leading up to the US Presidential Election in 2016. Our results demonstrate that by integrating and developing a combination of machine learning and data management techniques, it is possible to do this at scale with effective outcomes. The results of our project have the potential to be adapted towards solving and influencing other interesting social issues such as building neighborhood happiness and health indicators.

CCS CONCEPTS

•Information systems →Online analytical processing engines; Spatial-temporal systems; Sentiment analysis; Social tagging;

KEYWORDS

Spatio-Temporal; Sentiment Analysis; Scalable Framework; Bursty events; Election; Crowd Sentiment

1 INTRODUCTION

Since the inception of Twitter, people have been using the platform to express their opinion about current affairs, politics, business, sports, finance and entertainment. There are studies with statistics [36] showing that twitter is being used predominantly by people of age under 30 and the voice of young generation matters in paving road to the future of any country. Growing social media usage coupled with enhanced computing technologies have enabled us to analyze peoples opinion from their tweets at a large scale. An event like election that attracts the interest of crowd and has significant impact on society is worth analyzing.

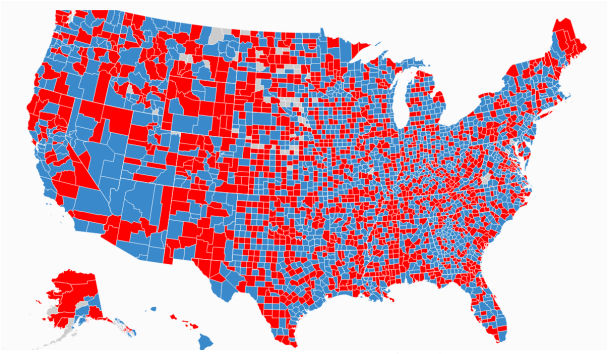


Figure 1: Popularity of Republican *(Red)* and Democratic *(Blue)* parties at US county level returned by Compass **for a query time interval; http://www.estorm.org.**

In this work we provide a framework to analyze the sentiment of the masses in spatio-temporal domain for any topic of interest. In particular, we used the US Presidential Election 2016 as a concrete example and analyzed the sentiment of crowd regarding this election. Masses express their feelings in tweets thus making it a valuable source of peoples true opinion. The vast amount of raw data that is generated in Twitter during an important and long event such as a Presidential Election poses a unique challenge to gather, analyze and extract information. Furthermore, people are inquisitive about the popularity of political parties with respect to geographical location; also about how breakout events, news and media affairs change the masses sentiment over time.

Using the Twitter stream as our primary data source and restricting ourselves to geo-tagged tweets, we developed a Sentiment Analysis framework to analyze and visualize large *spatio-temporal data*. We dub the framework Compass, which stands for Comprehensive Analytics on Sentiment for Spatiotemporal Data. Compass facilitates the end user to select an arbitrary time range to visualize popularity of the two political parties for each county (or state) of US for the specified time range. Alongside we present a *bursty* event detection technique to capture major event or subevents that happened before the US election. The objective is to capture the reaction of people on such events early in the process.

The Compass framework is generic where specialized machine learning models can be integrated to do specific task. For election analysis we deployed machine learning and deep learning models

KDD '17, August 13-17, 2017, Halifax, NS, Canada

DOI: http://dx.doi.org/10.1145/3097983.3098053

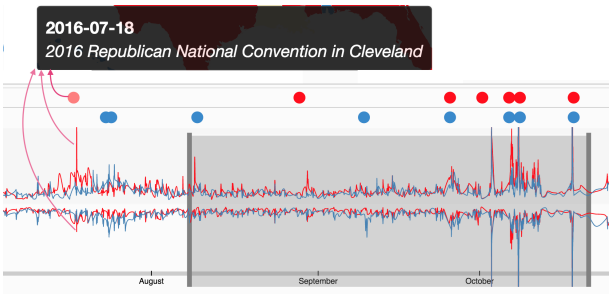


Figure 2: Bursty event timeline and brush selection for time range based analysis (gray window is a query time range).

with minimal human involvement to obtain highly accurate results. The main modules of Compass include:

(1) Tweet Classification Model;
(2) Sentiment Analysis Model;
(3) Bursty Event Detection Module;
(4) Spatio-temporal Analysis Framework;
(5) Visualization.

The integration of these modules enables an effective end-to-end analysis of spatio-temporal sentiment analysis over large spatio-temporal data. In the context of US Election, Republican and Democratic are two major US political parties. Figure 1 shows the popularity of the two parties at US county level based on the sentiment score for a queried time range returned by Compass. Red color counties indicate a higher sentiment score for the Republican party, whereas blue color counties favor the Democratic party.

The line charts with red and blue lines in Figure 2 represent the bursts of tweets about Republican *(red)* and Democrats *(blue)* as identified by Compass. Tweet bursts are mapped with the grand-truth events happened during that time. The positive vertical-axis represents a surge of positive sentiment tweets and the negative axis represents a surge of negative tweets about that party. The red and blue circle in Figure 2 are the events related to either Republican or Democratic party respectively. Compass has accurately detected all important surges related to significant events happened during the course of the election season. For example, the first red circle represents the event of the *Republican national convention in Cleveland*, it influenced positive reaction among republican supporters which can be seen with a positive red line surge in timeline.

Compass architecture. Figure 3 presents an overview of Compass' architecture. The framework starts with the collection of geo-tagged tweets from Twitter APIs. These geo-tagged tweets are passed through the tweet *classification model* where non-political tweets are rejected from further processing. For each political tweet the classification model has classified, the module also returns a probability of it being republican-related or democrat-related tweet. Next the *sentiment model* measures the polarity of each tweet and assigns a sentiment score in the range [0,1]. The *geo-mapping module* maps each tweet to a US county based on its geotag value. If the tweet is from non-US origin, we assign it to the country its geotag value is associated with. After political affiliation and sentiment score are assigned to tweets, they are persisted in the backend database. The spatio-temporal online analytical system module and the bursty event detection module utilize this database to enable end users to interact with Compass through its frontend visualization layer.

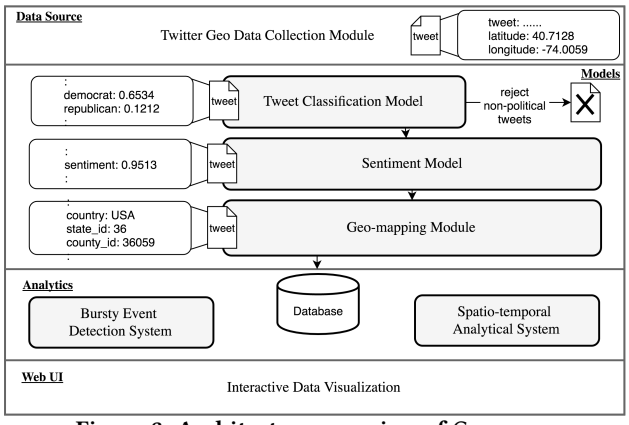


Figure 3: Architecture overview of Compass**.**

The bursty event detection module calculates a continuous *burst score* for the specified query time range. We have built a web-based visualization layer with javascript and D3.js interacting with the backend via RESTful APIs. The spatio-temporal analytical module with RESTful APIs calculates an aggregated sentiment score for each county based on end user's requests. Detailed discussion on each module is given in Section 3.

That said, Compass's tweet classification model uses little amount of input from human and achieves high accuracy. A pipeline of two classifiers is used to first detect *whether a tweet is related to politics* and then *to detect the political affiliation*. While the first model predicts the relevancy of a tweet to politics, the second model takes the tweets that are classified as political by the first model and detects the political affiliation. We describe more about the architecture of political tweet classifier in Section 3.2.

Compass's sentiment model measures the polarity of a tweet and maps it to a *sentiment score value* in [0,1], where values close to 0 is considered very negative and values close to 1 being strongly positive. Tweets with values around 0.5 are considered neutral. We trained the sentiment analysis model on our twitter data and obtained high accuracy with deep learning methods and fasttext [23]. We present the architecture of this model in Section 3.3.

The bursty event detection module leverages a novel bursty model with a strong mathematical interpretation that we have developed. The definition of *burst* is different from the concept of *trending* where in later the number of tweets matters. Bursty events gain popularity in a short time but might not be prevalent for longer time hence usually get overlooked by trending topics. Bursty events are important in capturing emerging topics and developments in spatio-temporal data and applications. Our bursty model is built over a novel mathematical model and has successfully detected significant events/sub-events which are bursty in nature.

The rest of the paper is organized as follows. Section 2 surveys related works. Section 3 presents the design details of Compass's multi-classification model, sentiment model, bursty event model, spatio-temporal processing module, and its visualization layer. A comprehensive experimental evaluation is provided in Section 4 to validate the design of Compass and its effectiveness. We conclude the paper with some final remarks in Section 5.

2 RELATED WORK

Next we briefly discuss approaches related to tweet topic classification, sentiment analysis (in general), and *bursty* event detection, as well as sentiment analysis for US election.

Tweet classification. Our political tweet classification starts with preparation of a training data set where we use topic modeling to learn about political words. Unsupervised methods for topic modeling using Latent Dirichlet Allocation (LDA) [7], pLSA [19] and modified versions of them are state-of-the-art methods [1, 3, 6, 42]. *Scalable topical phrase mining from text corpora* [14] uses LDA to predict high quality topics using phrases. More recent work *TweetSift* [28] uses external *knowledge base* and their own word embedding model to classify tweets. Word2vec [32] word embedding model computes vector representations of words. In our work word2vec plays a role in enriching our political keywords set. Godin et. al. [17] uses LDA and sampling to recommend hashtags. LDA performs well for documents of considerable size but faces limitations with microblogging and tweets [47]. Our method uses LDA on news articles and word embedding model trained on twitter data to create a much richer set of keywords. Logistic Regression, Support Vector Machine (SVM), Naive Bayes is used predominantly for text classification [15, 22, 30].

Sentiment analysis. From past decade opinion mining on text data has been a popular research topic. Pang et. al. [35] gives a comprehensive survey on incipient opinion mining research. Twitter sentiment analysis with machine learning approaches like SVM [21], lexicon based [38], LDA [13, 26] and neural network [12, 39] etc. Vosoughi et. al. [41] used contextual linguistic feature to do sentiment analysis. More recent methods include sequence processing techniques with Recurrent Neural Network (RNN) [27] variants like Long Short Term Memory (LSTM) [18], C-LSTM [48]. FastText [23] proved to be a very efficient and accurate technique for sentiment analysis. Stanford twitter sentiment corpus [16] is a standard dataset that we used to evaluate our models.

Bursty event detection. "Rising sharply in frequency" is defined as burst by Kleinberg [25]. Zhu et. al. [49] modeled "bursts" as cumulative average of frequency over sliding windows of different size. They applied Haar wavelet decomposition as their basis to detect "bursts". Shamma *et al.* [37] defined *peaky topics* which is modeled with normalized term frequency score. Lu *et al.* [29] defined *trendy topic* with a variant of Moving Average Convergence Divergence prediction method to find trending score. Al Sumait *et al.* [2] and Cataldi *et al.* [10] used window based approach with online LDA and aging theory respectively. Xie *et al.* [46] is the only work in document stream domain, defined "bursty" as rate of incoming stream over time. They have used idea of acceleration and Exponential Moving Average to model bursty topic. Our work is different in the aspect of modeling "burstiness" measure and also provides novel data structure to perform "bursty" queries over an arbitrary query sliding window on large historical data.

US election. To the best of our knowledge, few scholar publications [4, 9] have used opinion mining on Twitter to analyze US election. However their aim is to predict the popular votes and bias without involving spatial and temporal distribution of twitter data. Bovet et al. [9] build a twitter user network and used their tweets hashtag to find the opinion about the user. Our work is different in the sense of building a spatial-temporal sentiment map through the Compass framework for the US Election 2016.

3 THE DESIGN OF COMPASS

3.1 Twitter Geo Data Collection

We collected geo-tagged tweets from two APIs of Twitter. One is the 1% streaming API and the other one is the location based search API. From the 1% streaming API we only filter tweets with geo-location. For the location API, partitioning search locations and prioritizing the number of queries to different partitions to collect the maximum number of tweets is essential, for example, New York City is more likely to generate more tweets, and at a much higher frequency too, than a small town in southern Utah. We partition the global map into disjoint bounding boxes, where each bounding box represents a query region to twitter's location based search API. The naive approach is to send queries for these bounding boxes to the location based search API in a round robin fashion. But that clearly doesn't reflect dramatic difference in terms of data arrival rates at the regions represented by these bounding boxes. We can not only collect more tweets, and but also reflect the data arrival rates more accurately if we predict the boundaries of the bounding boxes of search and the frequency of queries for each bounding box to make, based on the geo-location statistics from the geo-tagged tweets. That said, we collected 7 days of tweets from the 1% API to get the geo-location statistics, from which we derived a partitioning strategy so that bounding boxes are of different geometric size and guarantee that the data arrival rate of each bounding box is roughly the same. We also decided the query frequency of each bounding box based on the statistics in a multi-threaded environment, which helps avoid the rate limit error resulted from exceeding the query rate constraint enforced by Twitter . Collected geo-tagged tweets from the two APIs are streamed to different services and stored in a backend database.

Since we have to deal with a large amount of tweets, and tweets are represented in Json format, we have used a clustered MongoDB instance as our backend database. The MongoDB instance is running over a cluster of 16 nodes.

3.2 Political Tweets Classification

In order to distinguish tweets related to politics from non-political ones and then subsequently to find the particular political alignment, a semi-supervised approach is adopted. Our focus primarily is to use minimal human input to design a system that can perform the aforementioned classifications with good generalization.

Using just a list of keywords that can detect politicalness and the political leaning of a tweet will not be feasible because of two reasons. The first reason being the infeasible task of storing the list of political affiliation keywords given the volume with which new content is generated on Twitter. Secondly, even keyword filtering does not guarantee correct classification since tweets matching with keywords might have different context all together and hence can lead to false classification. On the other hand, many machine learning methods need a properly labeled dataset to train a model. But preparing such dataset is manual labor intensive and can be very time consuming depending on the size of the dataset. Henceforth,

Compass uses semi-supervised techniques to prepare its training data for building its political tweets classification module.

The two important parts of the module are training data preparation and creation of classification model. This approach is generic and can work with any topic e.g. politics, sports, finance etc.

3.2.1 Training Data Preparation. Initially, we collect news articles from well known news sites such as *NYTimes, Fox News, CNN* etc. A news crawler module with python scrapy and splash has been used to collect articles and sanitizing them. The scraped articles have the structure given in Listing 1.

```
{"article": "text .....", "author": "author name",
"date": "2016-08-04", "focus": "description",
"link": "http://...", "origin": "NYTimes",
"title": "news title"}
```

Listing 1: Article structure.

These articles are then fed to a *Latent Dirichlet Allocation* (LDA) topic modeling algorithm to find the keywords related to politics. Our hypothesis is that US politics can be represented as a *multinomial distribution of words* that are often associated with it. Another assumption is about the issues involved and being discussed about specific US political parties. E.g. In Election 2016, *email scams* is a topic related to Democrats; similarly *federal tax pay* is often associated with a Republican subject (Donald Trump refused to release his tax return).

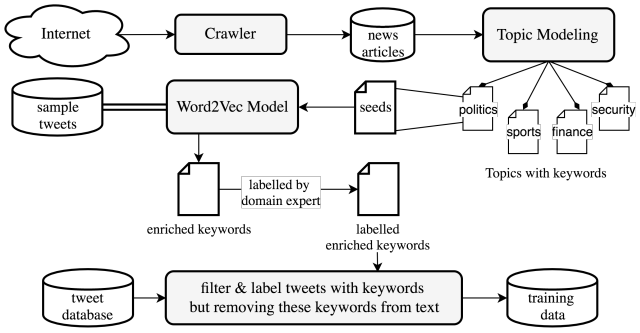


Figure 4: Training data preparation in Compass.

Using the LDA [7] model on articles collected over a period of one month from July 2016, we obtained keywords associated with US politics. We named/treated this set of keywords as *seeds*. Since, the lingo used in news articles is different from Twitter , this *Seeds* list in its current form would not be of much help in classifying tweets. Hence, to address this problem, we make use of Word2Vec [32, 33], another unsupervised method, to embed the words in a dense semantic space. We trained our *word2vec* model on a snapshot of Twitter data with 8.5 million tweets from the month of July, 2016 to find similar words to our existing seeds in the semantic vector space. For each *seed* word, we find its k nearest neighbors using cosine distance and add them to a new list called *Enriched Keywords* if they are not already present. This enables us to extend our *seeds* list to a more comprehensive list; see Figure 5. A sample of such *enriched keywords* is given in Table 1.

Even though the nearest neighbors method gives a very good list of enriched keywords, since Word2vec models global context, there will be some noisy words in this list, hence Compass makes use of

Party	Keywords
Republican	gopfail, gop's, BoycottTrump, GoTrump,LockHimUp, evangelical, gopers, imwithhim, donthecon, maga, PRyan, MittRomney, ColoSenGOP, TheFloridaGOP, ChrisChristie RealBenCarson,GovPenceIN, etc.
Democrat	jillnothill, HillaryForAmerica, imnotwithher, huma, obamas, NotReadyForHillary, imwithhernow, hillaryforprison, Flotus, killary, Libtarded, berniesellsout, lnyhbt, hillarysliesmatter, turncongressblue, etc.

Table 1: Enriched keywords with political affiliation.

domain experts to refine this list and also to provide the political affiliation of the entire keyword set. In total, we have around 300 keywords labeled by the experts.

Compass makes use of this labeling in its classifiers to learn the political affiliation and this the *only part* where Compass needs manual labeling guiding the training of classifiers. Note that in the subsequent sections, we also use manual labeling through crowdsourcing and already-labeled datasets, *but we use them for the evaluation of* Compass*' classifiers*, rather than building its classifiers.

Now from the tweet database Compass is able to filter out tweets having the keywords and label them with *Democrat, Republican or both* based on the affiliation of keyword. It labels the tweets that does not have any of these keywords as non-political, and further remove tweets labeled both democratic and republican since, the assignment of sentiment would become ambiguous in such cases and may require aspect based sentiment analysis which is left as an interesting future work. Figure 4 summarizes the construction of training data preparation in Compass.

3.2.2 Classification Model. This labeled data is now used as the training data to train the classification model. Keeping track of a large number of new keywords related to politics that would be created by Compass' data collection and training data modules on a daily basis would be both infeasible and costly. Hence training a machine learning classifier using aforementioned labeling as the ground truth would obviate the need for updating the keywords on a daily basis. We remove the keywords from our tweets during training to prevent the classifiers from learning the keyword-presence as a feature. This also ensures that our trained models will be generalizable to unseen tweets. We use two classifiers to classify a tweet *first into political or non-political* and *then to find the political affiliation.* That said, Compass' classification model is a set of two linear classifiers with the first one classifying a tweet into political or non-political classes and the second one detects the political affiliation depending on the output of the first one.

For both of these purposes Compass uses linear classifiers like SVM and Logistic Regression. We have followed multiple approaches to validate our model and make sure that it's generalizable beyond the keywords it has seen during training. Hence we report the performance of both models on two different sets of tweets in the political affiliation classification. The first one containing the keywords collected earlier and a second set of manually labeled tweets using a crowdsourcing layer developed for Compass.

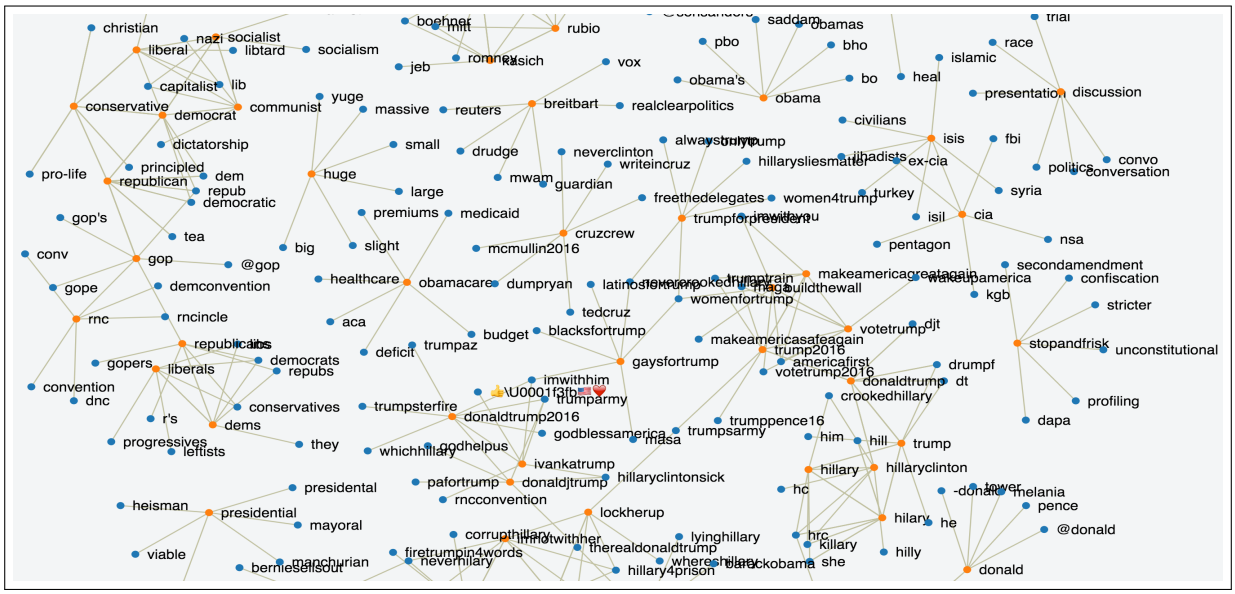


Figure 5: Red node keywords are *seeds* and blue node keywords are derived from *seeds* using cosine distance.

3.3 Sentiment Model

Once tweets are classified into either democrat-related or republican-related groups, Compass uses a sentiment analysis model to figure out whether a particular tweet in a group is *for* or *against* that group. For example, *Go Hillary. I'm with her!* will be classified into the Democrat group, and its sentiment score will be close to 1; whereas *Email leak is a crime!* will *also* be classified into the Democrat group, but with a very low sentiment score.

Compass leverages the Stanford Twitter Sentiment (STS) corpus created by Alec Go et. al.[16] using Distant supervision to train and validate its sentiment analysis classifier. Different machine learning algorithms, including the classical approaches like Naive Bayes, Logistic Regression, SVM and the more advanced and recent algorithms like LSTMs and fasttext, can be used.

3.3.1 LSTM-RNN. RNNs[31] differ from traditional neural networks fundamentally due to "hidden states" that maintain information about the the previous inputs. The hidden state of an RNN at time t is not only dependent on the input at t but also on the hidden state or memory at $t-1$. Since the parameters of the network are shared across all the timesteps, the gradient at each timestep also depends on previous time steps and uses Backpropagation Through Time (BPTT) for training.

LSTMs[18] are special kind of RNNs designed to capture long term dependencies by overcoming the vanishing gradients problem of RNNs. LSTMs improve upon standard RNNs by having the ability to remove or add information in hidden cell state with the help of three gates called the Input, Forget and Output gates. The hidden state h is at time t is calculated in the following fashion:

$$i_t = \sigma(x_t U_i + h_{t-1} W_i), \qquad f_t = \sigma(x_t U_f + h_{t-1} W_f)$$
$$o_t = \sigma(x_t U_o + h_{t-1} W_o), \qquad c_t = \tanh(x_t U_c + h_{t-1} W_c)$$
$$m_t = m_{t-1} \circ f_t + c_t \circ i_t, \qquad h_t = o_t \circ tanh(m_t)$$

where i_t, f_t and o_t are the input, forget and output gates at time t respectively, c_t is the candidate hidden state and m_t is the internal memory of the LSTM cell at time t. U and W are the input to hidden state and hidden state weight matrices respectively; x_t is the input vector at time t. While the input gate controls the amount of newly computed state for the current input the network passes through, the output gate regulates the amount of hidden state to share with the next timestep. The forget gate influence how much of the previous memory we should remember at this time step. The candidate hidden state c is calculated using the current input and previous hidden state while the current memory m_t is computed using the previous memory and the current input. Since LSTM memory units allow the previous state to pass through to the next time step, they don't suffer from the vanishing gradient problem.

In our case, Compass trains a word2vec model on the Twitter sentiment corpus to represent each word as a dense 50 dimensional vector. It passes the matrix of words and embeddings of dimensions *vocabulary size* $\times$ 50 to an embedded layer which will fine tune the word vectors during the training phase. An LSTM layer with 100 hidden units is added. The output of this layer is connected to a single neuron with sigmoid activation. We optimize for cross entropy loss using *adam* [24] optimization algorithm. The model is validated with various dropout ratios to improve regularization.

3.3.2 FastText. The fastText[8, 23] model architecture takes the sentence represented as n-gram features and embeds these features using a embedding layer. The embeddings of the n-gram features are then averaged to form the final representation of the sentence and are projected onto the output layer. This model proved to be very fast and achieved results comparable to state of the art in many datasets across different domains.

Compass represents each tweet as a combination of unigrams and bigrams by appending the bigrams after the unigrams. For example, if a tweet is "Vote for Hillary Clinton" and "Hillary Clinton" is in our list of bigrams, then the tweet would be represented as ['Vote', 'for', 'Hillary', 'Clinton', 'Hillary Clinton'] in Compass. This vector would then be sent to the embedding layer to learn the representation for each n-gram. These representations are averaged and projected to the output layer with a single neuron and sigmoid

activation function. We optimized for *binary-crossentropy* using the same *adam* optimizer as mentioned in the LSTM model above.

3.4 Geomapping Module

Peoples opinion and attitude towards public events always varies with region. When one local event happens, it usually results into wider range of discussion and news. To analyze geographic properties of opinions from twitter, we use geotagged tweets as mentioned earlier. Mapping of geotagged tweets to counties is important for faster processing. There are online services for this purpose, e.g., Google provides a Geocoding web service for it, but this is not feasible for large scale processing. Instead, Compass uses high precision GeoJSON i.e. geospatial data interchange format (RFC 7946)[20] to delineate geo-features. We use efficient ray-tracing technique to find a point within a featured polygon. A geo-tree based approach with country → state → county as nodes at each level provides significant pruning power and saves costly comparison significantly. This also increases the throughput of Compass. To further improve scalability and throughput, we extend the above idea to a distributed and parallel setting. In particular, we have integrated the above Geomapping Module into Simba [45], a spatio-temporal analytical engine built over Spark SQL.

3.5 Bursty Event Detection in Compass

The impact of any political event can be noticed by its popularity both in positive and negative sense. Popularity is a vague term associated with event. In social media *trending* is a term used where number of people talking about a event is considered as parameter. These events usually gains attention over a considerable amount of time. Another type of popularity is measured when an event gets tremendous response from people but only for a small period of time. But sometimes they get overshadowed by trending ones since the number of people involved over an extended period might be considerably less. The latter type needs proper mathematical model to measure its popularity. If properly modeled, it can also detect the events that can be trending in future. We call this type of events as *bursty events* influenced from the phenomenon *Burst*.

It is also noticeable that there is often a "spatial locality" effect, where an event usually first becomes popular in local area, then in the state and the whole country. If we can capture the bursts at a local area then we are a step ahead in detecting it. Hence capturing *bursty local events* is critical in large spatial data.

That said, to the best of our knowledge, we found there is lack of proper consensus on the definition of *burst* in various literatures. We present the mathematical model of *burst* by leveraging intuitions from classic physics. In the experiment and result section we show how our *bursty* model is effective for elections, where every now and then election campaigns generate interesting stories. Such an event creates sentiment waves among the crowd on Twitter. It is quite interesting to analyze peoples reaction especially in the event of election where peoples opinion matters and change over time with the development of new events.

Definition 1 (Burst). ***Burst*** *is a phenomenon identified when at least n number of event occurrences (of the same event) happens in τ time where τ is determined from probability distribution of gaps between occurrences.*

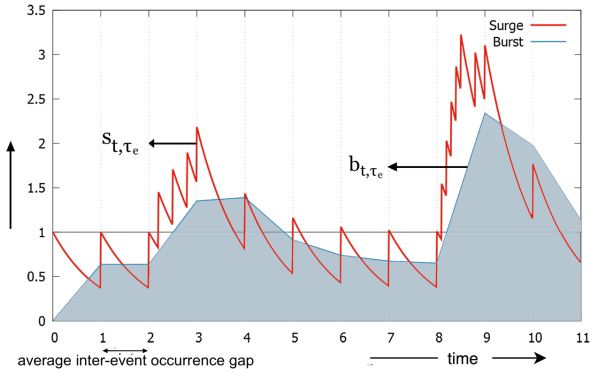


Figure 6: The definition of burstiness.

Note that in the context of Compass, an event occurrence refers to the fact that a particular political story/event is *mentioned* by a tweet in a tweet stream. Configuring n and τ parameters depends on the type of stream. However with our formulation we could reduce the number of parameters to one threshold parameter. For *Gamma Ray Bursts* the typical average time-span is 10 seconds [44] and the typical bursty event in social media is more than 15 minutes. We can configure τ with such values depending on the type of bursts we want to capture. We introduce ν as the average inter-event occurrence gap, which is numerically $\nu = \tau/n$.

Let $S_e = \{a_0, a_1, a_2, \ldots, a_{m-1}, a_m, \ldots\}$ be a data stream of microblog article a_i *about the same event* e each with timestamps $t_0 < t_1 \leq t_2 \leq \ldots t_{m-1} \leq t_m \leq \ldots$ respectively. Note that they may occur sporadically along the time dimension, i.e., the gap between t_i and t_{i+1} is an arbitrary values depending on the timestamp value of the occurrences of a_i and a_{i+1}.

We define term surge s_{t,τ_e} of the event e at a timestamp t between two occurrences a_{i-1} and a_i with Holt's Linear method of Exponential smoothing as follows.

$$s_{t,\tau_e} = \begin{cases} \frac{1}{\tau_e} & \text{if } t \text{ is } 0 \\ \alpha(t) s_{t_{i-1},\tau_e} + (1-\alpha^*)\frac{1}{\tau_e}, & \text{if } t \text{ is } t_i \\ \alpha(t) s_{t_i,\tau_e} & \text{otherwise} \end{cases}$$

where $0 \leq \alpha \leq 1$.

In the above definition, α is a the *smoothing parameter*, hence, α is time-variant and a smoothing variable, we define $\alpha(t)$ and α^* as:

$$\alpha(t) = e^{-\frac{t-t_{i-1}}{\nu_e}};$$
$$\alpha^* = e^{-\frac{t-t_{i-1}}{\nu_e}} = e^{-1} \quad \text{where } t - t_{i-1} = \nu_e$$

where t_{i-1} is last timestamp of the occurrence of e *before time* t (i.e., a_{i-1}, the *(i-1)*-th article with mentioning of e), ν_e is average inter-event occurrence gap of event e (which is a constant defined by τ_e/n_e), and t is the current query time. Note that for a given event e, τ_e is a user-defined constant and n_e is also a constant that captures the average expected number of occurrences of e with a timespan of τ_e. Lastly, S_e is extracted from a stream S that has a mixture of microblog articles with mentioning of different events.

Next, we introduce *burstiness* of an event as a continuous measure of burst over time.

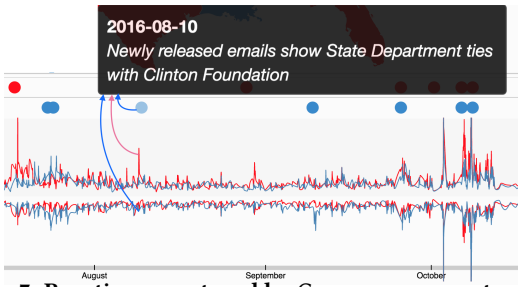


Figure 7: Burstiness captured by Compass**: www.estorm.org.**

Definition 2 (Burstiness). We define *burstiness* of event e at time t as area under the curve of surge over time τ_e. More area under curve indicates substantial bursts:

$$b_{t,\tau_e} = \int_{t-\tau_e}^{t} s_{t,\tau_e}\, dt.$$

The intuition of this definition is to measure the magnitude of bursts of event e in a window of size τ_e defined at time t, as illustrated in Figure 6. This constitutes the core of bursty event detection in Compass. Two types of query can be invoked:

(1) $\mathbb{Q}_{e,[start,end]}$ where e is an event type represented by a keyword and $[start, end]$ defines a query time window.
(2) $\mathbb{Q}_{[start,end],\gamma}$ where γ is a threshold.

Query 1 returns s_{t,τ_e} and b_{t,τ_e} streams for e where $start \le t \le end$. Query 2 returns all elements whose b_{t,τ_e} value has exceeded threshold in interval $[start, end]$.

There is also an *event discovery module* that cluster incoming tweets into different event groups. The details of which are omitted for the interest of space. Lastly, computing s_{t,τ_e} and b_{t,τ_e} values at arbitrary timestamp t *exactly* is expensive, especially in streaming setting. We have designed efficient approximation algorithms based on the idea of sketching, by extending classic sketching algorithms such as Count-Min (CM) sketch [11] and persistent CM sketch [43], which provides high quality approximations with theoretical guarantees. The technical details of which are beyond the scope of this paper and will be presented in a follow-up work.

That said, these components form an end-to-end pipeline for efficient bursty event discovery in Compass. It takes tweets stream as input and reports surge and bursts measure of positive and negative sentiment tweets for both US parties with associated events, and it is able to map these bursts to specific geo-temporal regions of interest. For example, Figure 7 shows how news of *newly released emails of Hillary Clintons* on 10th of August creates positive effect on republicans. Similarly in early September when *Clinton Calls Many Trump Backers 'Deplorables'* democrats received harsh criticism. More detailed results are presented in Section 4.

3.6 Support for Spatio-temporal Analytics

It is critical for Compass to provide a fast online spatio-temporal analytical processing unit, since large social media data with spatio-temporal features is ubiquitous. Thus, it is important to provide fast, scalable, and high-throughput spatial and temporal queries and analytics for location-based services (LBS). Traditional spatial databases and spatial analytics systems are disk- based and optimized for IO efficiency. But increasingly, data are stored and processed in memory to achieve low latency, and CPU time becomes the new bottleneck. Furthermore, as data size continues to grow, scale-out to a cluster of nodes is almost a required step towards realizing these goals. Compass integrates and uses Simba (Spatial In-Memory Big data Analytics) [45] that offers scalable and efficient in-memory spatial query processing and analytics for big spatio-temporal data.

Simba is a distributed in-memory spatial analytics engine based on Apache Spark. It extends the Spark SQL engine across the system stack to support rich spatial queries and analytics through both SQL and DataFrame query interfaces. Compass use HTTP RESTful API interface interacting in SQL with Simba. We leverage Simba's native distributed indexing support over RDDs and efficient spatial operators to do analytics. Simba extends Spark SQL's query optimizer with spatial-aware and cost- based optimizations to make the best use of existing indexes and statistics. Simba enables Compass with the flexibility and efficiency for analyzing geomapped data. Compass extends Simba with the support for temporal queries, and can process millions of tweets in some mini-seconds.

An interactive web visualization system build with D3.js for election sentiment map provides user a nice and intuitive visualization interface to analyze US election in the limelight of tweets (http://www.estorm.org). Let ρ be a probability threshold value for relevance/affiliation of a tweet with a particular party, β be a sentiment score threshold value, s and e be the start and end timestamps for a query time range, C_D and C_R be the class of tweets related to democrat and republican respectively, b is flag to indicate democrat or republican, and B is a query spatial region (represented by a bounding box B). Through the visualization layer, users are able to execute spatio-temporal analytical queries like :

(1) $\mathbb{Q}_+(\rho, \beta, s, e, b, B)$. This query finds all tweets from region B occurred within the time range $[s, e]$, and each tweet a satisfies $\Pr[a \in C_D] > \rho$ if $b = 0$ or $\Pr[a \in C_R] > \rho$ if $b = 1$, and $\text{sen}(a) > \beta$ ($\text{sen}(a)$ is the sentiment score of a). An aggregate version of this query returns the count and the average sentiment score of such tweets, *grouped by each county falling within B*.
(2) $\mathbb{Q}_-(\rho, \beta, s, e, b, B)$. This query is the same as $\mathbb{Q}$ except that it finds tweets with "negative" sentiments, i.e., all tweets such that $\text{sen}(a) < \beta$. A similar aggregation version also exists.
(3) $\mathbb{Q}_\pm(\rho, \beta_+, \beta_-, s, e, b, B)$. This query combines $\mathbb{Q}_+$ and $\mathbb{Q}_-$ using two sentiment score thresholds β_+ and β_- respectively, and its aggregation version estimates the popularity of all parties by weighted average method for all counties after applying the filtering criteria mentioned in previous queries.

4 EXPERIMENT AND RESULT

Using its twitter geo data collection module as described in Section 3.1, Compass has collected 286 million geotagged tweets from June 3rd, 2016 to October 30, 2016 which amount to 989GB. Among these tweets, Compass has identified approximately 2 million *geo-tagged* tweets *related to US politics* using its tweet classification module as described in Section 3.2. Among them, 822,062 tweets are labeled republican related, and 702,042 tweets are labeled democrat related, and the rest has no strong relevance with either party. A sentiment score is assigned to each tweet based on the sentiment module presented in Section 3.3 and all tweets are mapped to a US county

(or a country if it is outside the US) based on the geo-mapping module in Compass (Section 3.4). Spatio-temporal bursty detection over these tweets is supported by the bursty model and computation detailed in Section 3.5. Lastly, a visualization that supports spatio-temporal analytics as introduced in Section 3.6 is presented through a web interface, available at http://www.estorm.org.

To validate the design of Compass and investigate the effectiveness of its core components, we have conducted an extensive experimental evaluation using the above data set.

4.1 Tweet Classification

4.1.1 Political vs Non-Political tweet classification. As the bootstrap method in Section 3.2 shows, we obtained enriched Keywords with LDA and Word2Vec. Given these keywords, we build a *basic filter* to distinguish Political tweets and Non-Political tweets. A tweet which contains any keyword is regarded as a political tweet. However this basic filter would fail when a new political tweet arrives without any previously identified political keyword in it. To solve this problem, we build a binary classifier to obtain better performance in general.

With the basic filter, we obtain around 1.5 million of geo-tagged political tweets. To make the dataset balanced, we pick around 1.5+ million of non-political tweets randomly and add them to the data. We created training and testing data used an 80-20 split.

We instantiate and develop the classifier in Section 3.2 based on text classifiers like SVM and Logistic Regression (LR) to detect whether a tweet is political or non-political; they are denoted as *POLM1 and POLM2* (POLM stands for POLitical Model) respectively. For each model, two sets of features are tried: one is unigram and the other is the combination of unigram and bigram. As shown in the Table 2 both of Compass' linear classifiers performed very well. Especially, the combination features can improve the model.

From the previous analysis on LDA model, the vocabulary of high frequency words in political text is obviously different with other topics. Here we can also attribute this high accuracy to the presence of special vocabulary that is present in the political tweets. Furthermore, we can demonstrate that this binary classifiers can learn some politic-related features other than keywords. When training the models, we remove all keywords from the tweets which are labeled as political ones. As the result shown in Table 3, the resulting models without keywords performed as well as the previous models. From this observation, it demonstrates that Compass' binary political classification model can *predict well for new political tweets which does not contain any political keywords that were previously identified.*

	Accuracy	Precision	Recall	F1-Score
POLM1 + unigram	0.930	0.948	0.906	0.927
POLM2 + unigram	0.927	0.944	0.903	0.923
POLM1 + (uni + bi)-gram	0.933	0.951	0.910	0.930
POLM2 + (uni + bi)-gram	0.931	0.948	0.908	0.927

Table 2: Effectiveness of Political vs Non-Political classification.

4.1.2 Democratic vs Republican tweet classification. As described in the previous section, we collected around 1.5 million political tweets based on democratic and republican keywords. Since, it is not feasible to keep updating this list of keywords, Compass uses this dataset to create a machine learning classifier that could generalize to new tweets that might not contain these keywords as described in Section 3.2. In order to make the ML model to not just remember the keyword that caused the label to be democratic or republican, we followed the below two approaches.

	Accuracy	Precision	Recall	F1-Score
POLM1 + (uni + bi)-gram +tfidf	0.934	0.9494	0.914	0.931
POLM2 + (uni + bi)-gram + tfidf	0.932	0.9487	0.910	0.929

Table 3: POLM1-2 without political keywords in the training data.

A0: No keywords: We removed all the keywords from the dataset and trained the models to make sure that the model generalize to tweets without the keywords.

A1: Keeping 2 keywords: We kept only two keywords, i.e Trump and Hillary, in the model vocabulary to see how these two words impact the performance.

A model under each approach is then developed based on either SVM or LR following the discussion in Section 3.2. These variations are denoted as *PARTYM1 and PARTYM2* respectively (where PARTYM stands for Party Model). The results are shown in Table 4.

	Accuracy	Precision	Recall	F1-Score
PARTYM1 + A0	0.734	0.733	0.653	0.690
PARTYM2 + A0	0.741	0.756	0.634	0.690
PARTYM1 + A1	0.886	0.853	0.905	0.878
PARTYM2 + A1	0.889	0.844	0.927	0.884

Table 4: Performance for Democratic vs Republican classification.

Manual labeled tweets. We further collected 412 tweets that were manually labeled using a crowdsourcing module we built for Compass, and evaluated our models on this dataset to ensure the models are learning generalizable patterns. Table 5 has clearly validated and demonstrated the effectiveness of our models in Compass.

	Accuracy	Precision	Recall	F1-Score
PARTYM1 + A0	0.704	0.711	0.630	0.668
PARTYM2 + A0	0.672	0.674	0.595	0.632
PARTYM1 + A1	0.888	0.878	0.887	0.883
PARTYM2 + A1	0.888	0.867	0.902	0.884

Table 5: Performance for Democratic vs Republican classification on manual labeled tweets (through crowdsourcing).

4.2 Sentiment analysis

We used the 1.6 million tweets from the Stanford Twitter Sentiment (STS) corpus to train Compass' sentiment classifiers as shown in Section 3.3 and reported the scores on the manually labeled tweets. Compass' sentiment model (SENT) can be instantiated with different classifiers as discussed in Section 3.3 and we have tested the following models: SENT1: SVM based, SENT2: LR based, SENT3: MNB based (Multinomial Naive Bayes), SENT4: LSTM based as detailed in Section 3.3.1, and SENT5: FastText based as detailed in Section 3.3.2. We report the scores of these models on the dataset in Table 6.

SENT5 based on FastText is our best performing model with an accuracy of 84.4% on the test set. While it is possible to get accuracy above 95% if we include emoticons data, we removed emoticons during preprocessing to make sure the model learns from the text and not just memorizing emoticons.

Method	Accuracy
SENT1 + unigram	0.819
SENT2 + unigram	0.813
SENT3 + unigram	0.813
SENT1 + (uni + bi)-gram	0.808
SENT2 + (uni + bi)-gram	0.827
SENT3 + (uni + bi)-gram	0.825
SENT4	0.828
SENT5	0.844

Table 6: Comparison of models for sentiment analysis.

4.3 Bursty Event Detection

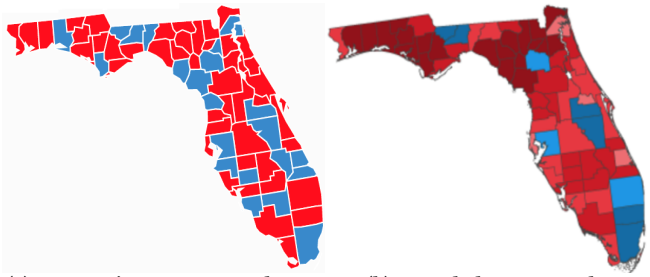
Our bursty model has made successful *early detection* of significant events throughout the election season. We compared our results with the original events happened [5]. An event is bursty when its bursty score is more than 1. Table 7 lists event dates, bursty score and keywords related to events. The measured parameters b_{r+}, b_{r-}, b_{d+}, b_{d-} are bursty scores for republican positive tweets, republican negative tweets, democrats positive tweets and democrats negative tweets respectively. Due to space constraint, we present only a sample of events as result. The bursty score from Table 7 suggests that our system successfully detected these events and keywords. It also shows that cumulatively there is more surge of Republican tweets than the Democrats.

Date	Keywords	b_{r+}	b_{r-}	b_{d+}	b_{d-}
		Event Description			
19th Oct	debatenight, debate Trump, answer the question, power transfer	11.8	8.3	5.6	7.4
		Flash Poll Trump Wins Final Presidential Debate			
12th Oct	sexual assault, sexual predator, grabbing, dressing rooms	2.2	2	1.8	1.5
		Women Accuse Trump of inappropriately touching them			
9th Oct	debate tonight, jumping ship, tape	15.8	10.0	5.9	7.7
		Trump and Clinton face off second presidential debate			
7th Oct	Women, sexual assault, Respect, abuse, hated	3.4	4.8	1.6	2.02
		Tape leaks showing Trump talking about groping women			
4th Oct	Pence, Kaine, crazy system, Poor Black, tax returns	4.0	2.6	2.2	2.7
		Kaine and Pence face off in vice presidential debate			
28th Sep	Clinton, answer questions, FBI, won the debate	2.3	1.9	1.9	1.72
		Clinton slams Trump in blistering presidential debate			
10th Sep	BasketOfDeplorables, Trump supporters, voters	1.72	1.3	1.3	1.7
		Clinton called out for Trump supporter deplorable			
29th Aug	Huma Abedin, Separate, Joe, Breitbart	1.85	1.4	0.7	0.8
		Clinton aide Abedin separates from Weiner			
25th Jul	DNC convention, Democratic, Hillary, Michelle, emails	1.3	1.1	2.3	2.0
		Democratic National Convention Philadelphia			
21th Jul	Make America Great, trumps speech, supporters, jobs	2.2	1.1	2.0	2.3
		Trump accepts Republican presidential nomination			
18th Jul	GOP convention, Trump, RNC, RNC in CLE	2.5	1.4	1.1	0.97
		Republican National Convention in Cleveland			

Table 7: Sample of bursty events detected with burst scores.

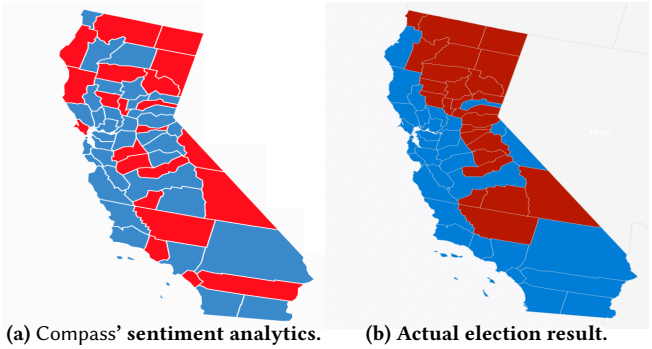
4.4 Spatio-temporal sentiment analysis

We compared the actual results of the election with the spatio-temporal sentiment maps produced by Compass. Figures 8 and 9 show how Compass' sentiment map matches to the election result to an extent. These sentiment maps are based on tweets from 21st July (*Republican Convention*) to 20th October (*Final Presidential Debate*). Note that we received predominantly less tweets from



(a) Compass' **sentiment analytics.** **(b) Actual election result.**

Figure 8: Results from the state of Florida.



(a) Compass' **sentiment analytics.** **(b) Actual election result.**

Figure 9: Results from the state of California.

non-urban areas which hinders Compass' sentiment analytics performance. There are counties from North Dakota, South Dakota , Montana, Nebraska, Kansas for which we received very little or no tweets. The sentiment map does suggest popularity but does not guarantee an actual win due to the US Electoral College voting process. Various other last-minute factors have also influenced the 2016 election result.

5 CONCLUSION

Twitter is still predominant among younger generations [34, 36]. Hence it reflects the voice of the youth generation mostly. It is not possible to get a complete picture until Twitter user becomes popular among all generation. Exit polls indicates former generation and non-graduate leans toward republican party and their percentage is statistically significant [40]. Beyond election, Compass is a useful and generic end-to-end framework for spatio-temporal sentiment analysis over any topic of interest. The framework can be easily extended with pluggable features like bursty event detection. Our ongoing and future work include adding more statistical tools and the ability to incrementally update various models within Compass. To address the multilingual solution on sentiment analysis Compass's default sentiment model can be replaced with other language say, *French* sentiment classification model to achieve the purpose. We are thankful to the reviewers for bringing this into our notice. Compass will be released as an open source project on Github.

6 ACKNOWLEDGMENT

The authors appreciate the valuable feedback provided by the anonymous reviewers, and the valuable feedback and discussion provided by Jian Li from the Tsinghua University that help improve this

work. Debjyoti Paul, Feifei Li, Murali Krishna Teja, Xin Yu, and Richie Frost were supported in part by NSF grants 1200792, 1251019, 1443046, 1619287 and the associated REU grants. Feifei Li was also supported in part by NSFC grant 61428204, a Google Faculty award, and a Huawei gift award.

REFERENCES

[1] Deepak Agarwal and Bee-Chung Chen. 2010. fLDA: matrix factorization through latent dirichlet allocation. In *WSDM*.
[2] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. 2008. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*. IEEE.
[3] Anima Anandkumar, Dean P Foster, Daniel J Hsu, Sham M Kakade, and Yi-Kai Liu. 2012. A spectral algorithm for latent dirichlet allocation. In *NIPS*.
[4] David Anuta, Josh Churchin, and Jiebo Luo. 2016. Election Bias: Comparing Polls and Twitter in the 2016 US Election. *arXiv:1701.06232* (2016).
[5] AOL. 2016. 2016 Presidential Election Timeline. (2016). https://www.aol.com/2016-election/timeline/ [accessed 08-02-2017].
[6] David M Blei. 2012. Probabilistic topic models. *CACM* 55, 4 (2012), 77–84.
[7] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR* 3, Jan (2003), 993–1022.
[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).
[9] Alexandre Bovet, Flaviano Morone, and Hernán A Makse. 2016. Predicting election trends with Twitter: Hillary Clinton versus Donald Trump. *arXiv:1610.01587* (2016).
[10] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. 2010. Emerging topic detection on twitter based on temporal and social terms evaluation. In *MDM/KDD*.
[11] Graham Cormode and S. Muthukrishnan. 2004. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. In *LATIN*.
[12] Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts.. In *COLING*.
[13] Adnan Duric and Fei Song. 2011. Feature selection for sentiment analysis based on content and syntax models. *Decision Support Systems* 53, 4 (2011), 704–711.
[14] Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R Voss, and Jiawei Han. 2014. Scalable topical phrase mining from text corpora. *PVLDB* 8, 3 (2014).
[15] Alexander Genkin, David D Lewis, and David Madigan. 2007. Large-scale Bayesian logistic regression for text categorization. *Technometrics* 49, 3 (2007).
[16] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project, Stanford* 1, 12 (2009).
[17] Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *WWW*.
[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[19] Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Uncertainty in artificial intelligence*. 289–296.
[20] IETF. 2017. RFC 7946 - The GeoJSON Format. (2017). https://tools.ietf.org/html/rfc7946 [accessed 08-Feb-2017].
[21] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target-dependent twitter sentiment classification. In *ACL HLT*. 151–160.
[22] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *ECML*.
[23] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759* (2016).
[24] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
[25] Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery* 7, 4 (2003), 373–397.
[26] Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! *Icwsm* 11, 538-541 (2011).
[27] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification.. In *AAAI*, Vol. 333. 2267–2273.
[28] Quanzhi Li, Sameena Shah, Xiaomo Liu, Armineh Nourbakhsh, and Rui Fang. 2016. TweetSift: Tweet Topic Classification Based on Entity Knowledge Base and Topic Enhanced Word Embedding. In *CIKM*.
[29] Rong Lu and Qing Yang. 2012. Trend analysis of news topics on twitter. *IJMLC* 2, 3 (2012).
[30] Andrew McCallum, Kamal Nigam, and others. 1998. A comparison of event models for naive bayes text classification. In *AAAI*, Vol. 752. 41–48.
[31] Larry Medsker and Lakhmi C Jain. 1999. *Recurrent neural networks: design and applications*. CRC press.
[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781* (2013).
[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[34] Dong-Phuong Nguyen, Rilana Gravel, RB Trieschnigg, and Theo Meder. 2013. " How old do you think I am?" A study of language and age in Twitter. (2013).
[35] Bo Pang, Lillian Lee, and others. 2008. Opinion mining and sentiment analysis. *FTIR* 2, 1–2 (2008), 1–135.
[36] PRC. 2016. Demographics of Social Media Users in 2016. (2016). http://www.pewinternet.org/2016/11/11/social-media-update-2016/ [accessed 08-Feb-2017].
[37] David A. Shamma, Lyndon Kennedy, and Elizabeth F. Churchill. 2011. Peaks and Persistence: Modeling the Shape of Microblog Conversations. In *CSCW*.
[38] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics* 37, 2 (2011), 267–307.
[39] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification.. In *EMNLP*. 1422–1432.
[40] New York Times. 2016. Election 2016: Exit Polls. (2016). https://www.nytimes.com/interactive/2016/11/08/us/politics/election-exit-polls.html
[41] Soroush Vosoughi, Helen Zhou, and Deb Roy. 2016. Enhanced twitter sentiment classification using contextual information. *arXiv:1605.05195* (2016).
[42] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*.
[43] Zhewei Wei, Ge Luo, Ke Yi, Xiaoyong Du, and Ji-Rong Wen. 2015. Persistent Data Sketching. In *SIGMOD*.
[44] Wikipedia. 2016. Swift Gamma-Ray Burst Mission — Wikipedia, The Free Encyclopedia. (2016). https://en.wikipedia.org/wiki/Swift_Gamma-Ray_Burst_Mission#Notable_detections [accessed 08-Feb-2017].
[45] Dong Xie, Feifei Li, Bin Yao, Gefei Li, Liang Zhou, and Minyi Guo. 2016. Simba: Efficient in-memory spatial analytics. In *SIGMOD*.
[46] Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim, and Ke Wang. 2013. Topicsketch: Real-time bursty topic detection from twitter. In *ICDM*. 837–846.
[47] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing twitter and traditional media using topic models. In *ECIR*.
[48] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A C-LSTM neural network for text classification. *arXiv:1511.08630* (2015).
[49] Yunyue Zhu and Dennis Shasha. 2003. Efficient Elastic Burst Detection in Data Streams. In *SIGKDD*.