TwitInfo: Aggregating and Visualizing Microblogs for Event Exploration

Adam Marcus, Michael S. Bernstein, Osama Badar,
David R. Karger, Samuel Madden, Robert C. Miller
MIT CSAIL
32 Vassar St., Cambridge MA
{marcua, msbernst, badar, karger, madden, rcm}@csail.mit.edu

ABSTRACT
Microblogs are a tremendous repository of user-generated content about world events. However, for people trying to understand events by querying services like Twitter, a chronological log of posts makes it very difficult to get a detailed understanding of an event. In this paper, we present TwitInfo, a system for visualizing and summarizing events on Twitter. TwitInfo allows users to browse a large collection of tweets using a timeline-based display that highlights peaks of high tweet activity. A novel streaming algorithm automatically discovers these peaks and labels them meaningfully using text from the tweets. Users can drill down to subevents, and explore further via geolocation, sentiment, and popular URLs. We contribute a recall-normalized aggregate sentiment visualization to produce more honest sentiment overviews. An evaluation of the system revealed that users were able to reconstruct meaningful summaries of events in a small amount of time. An interview with a Pulitzer Prize-winning journalist suggested that the system would be especially useful for understanding a long-running event and for identifying eyewitnesses. Quantitatively, our system can identify 80-100% of manually labeled peaks, facilitating a relatively complete view of each event studied.

Author Keywords
Twitter, Visualization, Exploration, Event Detection

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Misc.

General Terms
Design, Human Factors

INTRODUCTION
Twitter captures a moment-by-moment pulse of the public consciousness. Unfortunately, the format of the raw data from Twitter is a flood of opinions, information and emotion, and is very difficult to synthesize into a coherent timeline of events or sentiment. To address the limitations of raw Twitter feeds, researchers have built a number of one-off visualizations for a variety of domains, including political events [7, 24], media events [23], and crisis scenarios [27, 29]. Others have focused on building visualization over large static collections of tweets [6]. However, these are generally domain specific, and operate only on archived tweet data.

To address these limitations, we present *TwitInfo*, a platform for exploring Twitter in real-time. Given a search query related to an event, TwitInfo identifies and labels event peaks, provides a focus+context visualization of long-running events, and provides an aggregate view of user sentiment. The system extracts the posts that match keywords in the query and provides a graphical timeline interface that labels peaks of high-volume posts as subevents and highlights important terms and tweets in the conversation around each subevent. Users can zoom into this timeline to visualize more peaks and subevents. TwitInfo also includes a sentiment analysis engine that displays the crowd's sentiment about various subevents, and additionally shows the geographic distribution of posted messages. When there are frequently occurring links in a cluster of tweets which might provide a bigger picture, TwitInfo also displays these links in the user interface.

TwitInfo addresses two major technical problems inherent in scaling such visualizations to realtime performance on an unconstrained dataset: identifying subevents and summarizing sentiment. To identify peaks in conversation around a given topic in realtime, TwitInfo adapts signal processing techniques to social streams. Existing sentiment visualizations have the potential to mislead users because positive and negative sentiment classifiers have differing precision and recall (Figure 2). We introduce a normalization procedure to ensure that these visualizations produce correct results.

In this paper we make the following contributions:

1. We describe streaming algorithms to generate labeled event peaks and visualizations of arbitrary topics in real time from the Twitter stream. This approach allows users to track events as they unfold, like a sporting event or election, rather than restricting users to post-hoc analysis. TwitInfo marries the signal processing literature with social streams to make realtime end-user event tracking feasible.
2. We present an automatic peak detection and labeling algorithm with subevent drill down, which is the first to allow

focus+context exploration for social streams. TwitInfo is the first interface to provide navigable peak labels for social streams; our evaluation shows that these peaks were the main tool users chose to explore the interface. Users used peak labels to understand events, and most identified labels as the most memorable and helpful element. We found that TwitInfo's event detection algorithm, which identifies subevents, has recall and precision of 80-95%.

3. We identify a problem with aggregate visualizations of sentiment analysis over a collection of documents. The problem arises when positive and negative classifiers have different recall rates, which can lead to skew in the reported proportion of documents that are positive or negative. We present a technique for properly normalizing these statistics.
4. We describe a formative evaluation of the interface with 12 users and one professional journalist. We find that non-expert participants successfully used the interface to rapidly summarize complex, nebulous event timelines.

We start with a motivating scenario for TwitInfo. We go on to discuss related approaches, describe the system in detail, and present our algorithms. Finally, we evaluate the algorithms and user interface, and discuss future work.

MOTIVATING SCENARIO

It's June 2009 and Ellen has caught wind of protests in Iran concerning the outcome of the election. A few traditional news media articles have come out based on eyewitness reports, but the media are barred from covering events. Instead, many individuals utilize the Twitter microblogging service to broadcast updates and report on events on the ground. Ellen wants to get a bigger picture. Her key questions are: what were the major events of the protests, what were people located inside the country saying, and how do people generally feel about each event? Performing a Twitter search for the *#iranElection* hashtag (used by many people to identify the subject of their posts) produces a list of thousands of individual messages that are not grouped or organized in any meaningful way.

Instead, Ellen visits the TwitInfo website to view a timeline of tweets containing the term *#iranElection*. Whenever there is a peak in activity, TwitInfo highlights the peak and labels it with commonly occurring words like the protest location. She wants to know how individuals in Israel, whose citizens are interested in Iranian affairs, are reacting to the event. So, she turns to the map interface and zooms in to Tel Aviv, then reads several comments on the events. Ellen hypothesizes that public sentiment about the election will shift over time, so she uses TwitInfo's sentiment aggregation interface to examine overall opinion at each peak in the timeline including a discussion that has just popped up. Ellen centers in on one protest event of interest, and follows a popular link mentioned in tweets for deeper exploration.

RELATED WORK

We now review related work on Twitter analytics, temporal exploration interfaces and topic detection algorithms.

The growth of microblogging has led to an expanding body of research. Java et al. studied the topological and geographical distribution of Twitter [16]. Kwak et al. found that Twitter exhibited network properties more like a news media than a social network [17]. Naaman et al. detailed a division between informers, who spread information, and meformers, who discuss their own affairs [19]. Starbird, Vieweg and colleagues have contributed analyses of microblog usage and information lifecycles during crisis situations [27, 29].

A particularly relevant branch of microblogging research analyzes reactions to news events on Twitter. Many of the papers in this category involve hand-creating timelines, graphs and visualizations very similar to those that TwitInfo produces automatically and interactively. Diakopoulous and Shamma did inspirational work in this vein, demonstrating the use of timeline analytics to explore the 2008 Presidential debates through Twitter sentiment [7]. Shamma et al. later demonstrated a similar analysis at the President Obama's Inauguration, finding that tweet volume increases and @replies drop during important moments [24]. Starbird et al. tracked tweets from different geographical areas on a timeline as the Red River flood proceeded [27]. Gaffney performed a retrospective analysis of the recent Iran election, in which Twitter played a pivotal role in news reporting [11]. Jansen et al. found that tweet sentiment analysis could provide useful insights into product opinion, an idea we leverage for the sentiment visualization in TwitInfo [15]. Our goal in TwitInfo is to make these kinds of analyses easier and faster, without requiring programming knowledge.

More recently, Diakopoulos et al. presented Vox Civitas [6], a timeline-based visualization of events discussed on microblogs. Vox Civitas displays sentiment over time, and can be used to annotate a video of the event with tweet-generated commentary. Dörk et al. [8] present a timeline-based visualization used as a backchannel for events discussed on microblogs. Like both systems, TwitInfo generates an event-summarization dashboard, but contributes a streaming algorithm for event detection and timeline annotations that our user study finds to be effective for exploring large collections of tweets. TwitInfo also contributes a method for correcting a common problem that arises in displaying aggregate sentiment as estimated by automated classifiers.

TwitInfo builds on recent work on exploratory interfaces for temporal exploration. Most closely related are Statler [23] and Eddi [4], which both provide timeline visualizations of Twitter data. Statler focuses on media events, and Eddi on only one user's feed; our goal with TwitInfo is to extend and generalize these interfaces to an arbitrary topic of interest on Twitter. To do this, we draw on other work exploring temporal visualization (for a survey, see Silva [26]). Leskovec et al.'s Memetracker pursued similar goals, tracking prominent phrases through blogs and news media [18] rather than Twitter. Zoetrope allows the user to graph web content trends like temperature or images over time [1]. TwitInfo also takes inspiration from Continuum's [2] notion of hierarchical timelines, allowing users to expand subevents in the timeline for further exploration. Opinion Space [9] investigated visual layouts for browsing many user comments. In the future, we plan to integrate its ideas for treating tweets with sentiment or opinion separately from information-sharing tweets.

We build on work related to topic and event detection. Swan and Allen developed a χ^2 significance test for identifying bursty extracted noun phrases [28] in a stream, but it is not appropriate for Twitter's needs. We need an online algorithm that can update as tweets stream in, and tweets are so short that noun agreement may not occur naturally. Newsjunkie developed algorithms for users who wish to find novel content in their stream [10]. BlogScope [3] pursues many of the same goals as TwitInfo, but is focused on blogs, and does not go into detail about peak detection algorithms. One body of work that does address such scenarios is TCP congestion control [14], which has minimal memory requirements and works in an online fashion to determine outlier delays in packet transmission. We base our approach on this notion.

Finally, it is worth touching on some related non-academic projects. Twitter Sentiment[1] is a site which displays a message frequency timeline, sentiment pie chart, and sentiment-coded tweets. The site inspired many visual elements in TwitInfo, and we utilize their approach to extract sentiment from tweets, but they fall victim to the aggregate sentiment normalization issues we solve with TwitInfo. Google News Timeline[2] facilitates exploration of news media in given time ranges, but does not visually signal to the user which items to focus on, and does not provide intuition into which articles are selected for a given period. Systems such as Flipboard[3] make news socially relevant by providing a magazine-like interface for exploring news sources extracted from your social networks on Twitter and Facebook, but do not provide context about events of interest. Twitter is planning a real-time analytics framework[4], but this does not seem to be geared at relaying news or informing end-users.

THE TWITINFO SYSTEM

In this section we describe TwitInfo, beginning with the user's initial event query, then detailing the exploration interface and the creation of nested events.

Creating the Event

TwitInfo users define an *event* by specifying a Twitter keyword query. For example, for a soccer game users might enter search keywords *soccer, football, premierleague,* and team names like *manchester* and *liverpool*. Users give the event a human-readable name like "Soccer: Manchester City vs. Liverpool" or "Obama presidency," as well as an optional time window. When users are done entering the information, TwitInfo saves the event and begins logging tweets matching the query. In its current implementation, we only track tweets for a keyword when a user first enters the keyword into the system, but this is not a hard limitation: to get a historical view of an event from its keywords, we could continuously collect a sample of all tweets, and historically index each keyword as users begin tracking them.

Timeline and Tweets

Once users have created an event, they can monitor the event in realtime by navigating to a web page that TwitInfo creates for the event. The main TwitInfo interface (Figure 1) is a dashboard summarizing the event over time. The dashboard displays a timeline for this event, raw tweet text sampled from the event, an overview graph of tweet sentiment, and a map view displaying tweet sentiment and locations.

The event timeline (Figure 1.2) reports tweet activity by volume. The more tweets that match the query during a period of time, the higher the y-axis value on the timeline for that period. So, when many users are tweeting about a topic (for example, *Obama*), the timeline spikes. TwitInfo's peak detection algorithm, described later in the paper, automatically identifies these spikes and flags them as peaks in the interface. Peaks appear as flags in the timeline and appear to the right of the timeline along with automatically-generated key terms that appear frequently in tweets during the peak. For example, in Figure 1.2, TwitInfo automatically tags one of the goals in the soccer game as peak "F" and annotates it on the right with representative terms in the tweets like 3-0 (the new score) and tevez (the soccer player who scored). Users can perform text search on this list of key terms to locate a specific peak. To visually scale from short events to long-running queries, TwitInfo will aggregate tweets on the timeline by minute-, hour-, or day-level granularity depending on the timespan the user is viewing.

The timeline also provides a way to filter the tweets in the rest of the interface: when the user clicks on a peak, the other interface elements (map, links, tweet list, and sentiment graph) refresh to show only tweets in the time period of that peak.

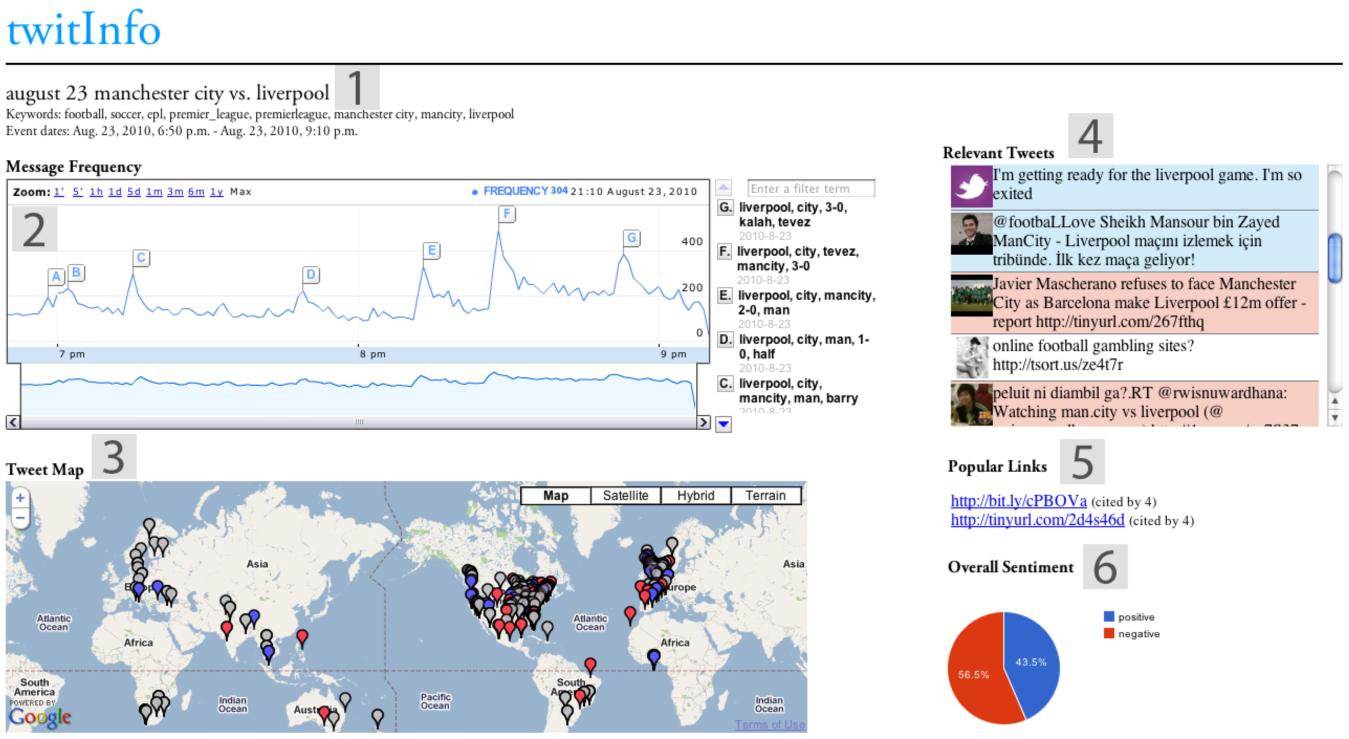
While derived information is useful at a glance, users need to have access to the tweets behind the event. The Relevant Tweets panel (Figure 1.4) lists tweets that fall within the event's start and end times. These tweets are sorted by similarity to the event or peak keywords, so that tweets near the top are most representative of the selected event. Tweets are colored red, blue, or white depending on whether their detected sentiment is positive, negative, or neutral.

Aggregate Metadata Views

In addition to skimming sentiment for individual tweets, users may wish to see the general sentiment on Twitter about a given topic. The Overall Sentiment panel (Figure 1.6) displays a piechart representing the total proportion of positive and negative tweets during the event. We discuss the difficulties in accurately depicting aggregate sentiment and other aggregated classifier output in the Algorithms section.

Twitter users will share links as a story unfolds [17]. The Popular Links panel (Figure 1.5) aggregates the top three URLs extracted from tweets that occurred during the selected subevent if users want sources for further exploration.

[1]http://twittersentiment.appspot.com/
[2]http://newstimeline.googlelabs.com/
[3]http://www.flipboard.com/
[4]http://gigaom.com/2010/09/22/twitter-to-launch-free-real-time-analytics-this-year/


twitInfo

august 23 manchester city vs. liverpool

Keywords: football, soccer, epl, premier_league, premierleague, manchester city, mancity, liverpool

Event dates: Aug. 23, 2010, 6:50 p.m. - Aug. 23, 2010, 9:10 p.m.

Message Frequency

Zoom: 1' 5' 1h 1d 5d 1m 3m 6m 1y Max

● FREQUENCY 304 21:10 August 23, 2010

G. liverpool, city, 3-0, kalah, tevez 2010-8-23

F. liverpool, city, tevez, mancity, 3-0 2010-8-23

E. liverpool, city, mancity, 2-0, man 2010-8-23

D. liverpool, city, man, 1-0, half 2010-8-23

C. liverpool, city, mancity, man, barry 2010-8-23

Relevant Tweets

I'm getting ready for the liverpool game. I'm so exited

@footbaLLove Sheikh Mansour bin Zayed ManCity - Liverpool maçını izlemek için tribünde. İlk kez maça geliyor!

Javier Mascherano refuses to face Manchester City as Barcelona make Liverpool £12m offer - report http://tinyurl.com/267fthq

online football gambling sites? http://tsort.us/ze4t7r

peluit ni diambil ga?.RT @rwisnuwardhana: Watching man.city vs liverpool (@

Tweet Map

Popular Links

http://bit.ly/cPBOVa (cited by 4)

http://tinyurl.com/2d4s46d (cited by 4)

Overall Sentiment

Figure 1. The TwitInfo user interface. 1) the user-defined name of the event, as well as keywords sent to the Twitter Search API to log the tweets. 2) The central timeline interface, with the y-axis corresponding to tweet volume. Automatically-detected peaks (subevents) are labeled with lettered flags. When the user clicks on a peak, all the other interface elements filter to tweets in that time period. 3) Event-related tweets with geolocation are displayed on a map. 4) Tweets for the currently selected event or peak (subevent), colored red if TwitInfo detects negative sentiment or blue if TwitInfo detects positive sentiment. 5) The most popular links in the currently-selected event or subevent. 6) Aggregated sentiment of all tweets in the event or subevent.

Often, opinion on an event differs by geographic region. A user should be able to quickly zoom in on clusters of activity around New York and Boston during a Red Sox-Yankees baseball game, with sentiment toward a given peak (e.g., a home run) varying by region. The Tweet Map (Figure 1.3) displays tweets that provide geolocation metadata. The marker for each tweet is colored according to its sentiment, and clicking on a pin reveals the associated tweet.

Creating Subevents

An important contribution of TwitInfo is its interactive support for hierarchically nested events: users can start by viewing several months of the Obama presidency on Twitter, zoom in to a subevent to see a timeline of just a single speech, then zoom in again to do a detailed analysis of reactions to a single remark the President made. The user can transform any peak on the timeline into its own first-class event with a dedicated page. To do so, users are asked to "Label this event" in the timeline and provide a human-readable name for the event. This serves the purpose of providing a better event label, and for allowing future users to navigate deeper into the subevent structure.

Realtime Updating

Diffusion speed is one of Twitter's competitive advantages in tracking the news, so it is important for Twitter interfaces to be able to handle ongoing events. Because TwitInfo captures realtime streams from Twitter, its interface reacts accordingly. The interface refreshes at regular intervals to include any new tweets that have entered the system. In this way, a user might track a speech, sporting event, or media event in realtime as it occurs.

ALGORITHMS FOR DISPLAYING EVENTS

We now turn to the algorithms we use for detecting and displaying subevents in a collection of tweets.

Event Detection

A key technical contribution of this paper is identifying temporal peaks in tweet frequency. Our algorithm first bins the tweets into a histogram by time, for example by minute. This binning allows us to count the tweet-arrival rate in each time period. We then calculate a historically weighted running average of tweet rate and identify rates that are significantly higher than the mean tweet rate. For these rate spikes, we find the local maximum of tweet rate and identify a window surrounding the local maximum. Finally, we collect tweets within this window that contain the desired keywords, and select frequent terms to provide an automated label of each peak. We now describe each of these steps in detail.

Given a time-sorted collection of tweets, we first group the tweets that were posted within a one-minute time window of one-another. Increasing bin size smooths out small spikes in longer events, and can sometimes aid in finding larger trends. Users can adjust bin size to hours or days.

After our binning process, we are left with a list $[C_1, ..., C_N]$ of tweet counts, where C_i is the number of tweets in bin i. For example, a list $[25, 50, 13]$ with minute-sized bins would mean that there were 25 tweets in the first minute of an event, 50 tweets in the second, and 13 tweets in the third. We wish to determine when there is a peak in conversation about some topic. A naïve approach would be to find local maxima amongst the C_i's, but this would mis-identify peaks which are marginally taller relative to their neighbors. Instead, we wish to identify each bin i such that C_i is large relative to the recent history $C_{i-1}, C_{i-2}, ...$ A similar problem is addressed in TCP's congestion control mechanism [14], which must determine whether a packet is taking unusually long to be acknowledged and is thus an outlier. The analogue in our system is that the algorithm must determine whether a bin has an unusually large number of tweets in it. We take inspiration from TCP's approach, which uses a weighted moving average and variance.

Algorithm 1 Offline Peak-Finding Algorithm

```
 1: function **find_peak_windows**($C$):
 2: windows = []
 3: mean = $C_1$
 4: meandev = variance($C_1$, ..., $C_p$)
 5:
 6: for $i = 2$; $i <$ len($C$); $i + +$ do
 7:     if $\frac{|C_i - mean|}{meandev} > \tau$ and $C_i > C_{i-1}$ then
 8:         start = $i - 1$
 9:         while $i <$len($C$) and $C_i > C_{i-1}$ do
10:             (mean, meandev) = update(mean, meandev, $C_i$)
11:             $i + +$
12:         end while
13:         while $i <$ len($C$) and $C_i > C_{start}$ do
14:             if $\frac{|C_i - mean|}{meandev} > \tau$ and $C_i > C_{i-1}$ then
15:                 end = $- - i$
16:                 break
17:             else
18:                 (mean, meandev) = update(mean, meandev, $C_i$)
19:                 end = $i + +$
20:             end if
21:         end while
22:         windows.append(start, end)
23:     else
24:         (mean, meandev) = update(mean, meandev, $C_i$)
25:     end if
26: end for
27: return windows
28:
29: function update(oldmean, oldmeandev, updatevalue):
30: diff = |oldmean − updatevalue|
31: newmeandev = $\alpha$*diff + (1−$\alpha$)*oldmeandev
32: newmean = $\alpha$*updatevalue + (1−$\alpha$)*oldmean
33: return (newmean, newmeandev)
```

An offline version of peak detection is described in Algorithm 1. The function **find_peak_windows(C)** takes binned tweet counts as input $[C_1, ..., C_N]$. The function returns a list $[W_1, ..., W_N]$ of peak windows, where window $W_i = (S_i, F_i), S_i < F_i$ is represented by S_i, the start bin of the window, and F_i, the final bin of the window. After initializing the mean to the first bin count (line 3) and the mean deviation to the variance of the first p (we use $p = 5$) bins (line 4), we loop through the subsequent bin counts (line 6). Line 7 contains our peak detection logic: if the current bin count (C_i) is more than τ (we use $\tau = 2$) mean deviations from the current mean, and the bin counts are increasing, we say that the increase is significant, and begin a new peak window (line 8). The loop in lines 9-12 is performing hill-climbing to find the peak of this window: we loop until we reach a bin count smaller than the previous one. Every time we iterate over a new bin count, we update the mean and mean deviation (lines 10, 18, 24). After a peak C_i has been found, we enter the loop of lines 13-21. Here we follow the peak to its bottom, which occurs either when the bin counts are back at the level they started (line 13) or another significant increase is found (the *if* statement on line 14).

To summarize the algorithm: when the algorithm encounters a significant increase in bin count relative to the historical mean, it starts a new window and follows the increase to its maximum. The algorithm ends the peak's window once the bin count returns to the same level it started at, or when it encounters another significant increase.

The implementation of **find_peak_windows(C)** in TwitInfo is different than the one in Algorithm 1 to facilitate streaming interactivity. In order to make peak-finding an online algorithm, we make the iteration between lines 6 and 26 reachable using a continuation-passing style, and make the function take only the latest bin count with each call.

The **update(oldmean, oldmeandev, updatevalue)** function is the update step: given a new bin count (*updatevalue*), it updates *oldmean* and *oldmeandev* and returns their new values. Because we are doing this in a streaming context (we do not want to maintain a list of historical values) and because we want to eventually forget old bin counts to adjust to changing message frequency trends, we maintain an exponentially weighted moving mean of bin counts. Similarly, we maintain exponentially weighted moving mean deviations. The formulas for these update steps are on lines 31 and 32, and require that $\alpha < 1$. We have found that $\alpha = .125$ captures a reasonable amount of historical information.

The significance tests of lines 7 and 14 are similar to outlier detection criteria which identify outliers if they are a certain number of standard deviations from the mean. We utilize mean deviation instead of standard deviation because the update step for mean (line 31) does not require keeping historical bin count values. The update steps of lines 31 and 32, as well as the use of exponentially weighted moving averages and mean deviations are based on ideas from TCP congestion control.

After identifying a window $W = \{S, F\}$, we generate a label for W. We consider all tweets containing the event's keywords that were posted between the start of bin S and the end of bin F. We then tokenize all tweets into unigrams and rank the unigrams by their Term Frequency / Inverse Document Frequency (TF-IDF) [22]. TF is calculated across all tweets in W, and IDF is calculated for tweets in all windows. We present the top five ranked unigrams as the label for W.

Removing Noisy Query Terms

If a user includes a tangential but high-volume keyword in the list of event keywords, the resulting timeline can lose its meaningful signal. For example, suppose the user wanted to track music news using band keywords like *Journey*, *T-Pain*, and *Justin Bieber*. Justin Bieber is a Twitter celebrity and perpetually trending — the volume of Bieber-related tweets would overwhelm signal about the other musicians.

We employ an IDF-normalizing technique for avoiding this situation. Rather than count the number of tweets which contain any of the desired keywords in a given timeframe, we normalize the count of tweets containing always-popular terms by the terms' global popularity (IDF). Thus, because *Bieber* frequently appears in the Twitter stream, the contribution of tweets containing that term to the volume of a period is dampened.

Identifying Relevant Tweets

Because there are typically more tweets than can be digested by a user for any subevent, we try to identify relevant tweets for the user in the Relevant Tweets list (Figure 1.4). Given our 5-term label for each identified event, we rank tweets by the number of terms from the subevent label that the tweet contains. Since retweets contain the same words as a tweet and we want to promote diversity, we halve the matching term count of retweeted tweets.

Representing Aggregate Sentiment

Our sentiment analysis algorithm places tweets into *positive* and *negative* classes. It uses a Naïve Bayes classifier trained on unigram features. We built the algorithm around an idea described by Go et al. [12], generating training sets for the positive and negative classes using tweets with happy and sad emoticons.

Traditionally, twitter-based sentiment analysis algorithms are evaluated tweet-by-tweet, using standard precision/recall metrics. In practice, however, they are often used in aggregated form. The standard means of displaying aggregate sentiment can pose a biased view. We illustrate the issue in Figure 2. TwitInfo contributes a correction for this bias.

Our classifier predicts the probability that a tweet is a member of the *positive* class (p_p) and the probability that it is a member of the *negative* class (p_n). Not all tweets express a strong sentiment, so we need a confidence threshold τ on the classifiers below which we predict a *neutral* category. Increasing the threshold for a class on a well-trained classifier has the effect of increasing the precision and decreasing the recall of the classifier. For example, we can increase our precision on the positive class by increasing threshold τ_p for which we will accept tweets with p_p values as positive.

Our positive and negative classifiers will have different precision/recall curves as we adjust their thresholds. It is relatively unlikely that the precision/recall curves will be identical for both classes. This may cause the two classes to have different recall values at the same precision.

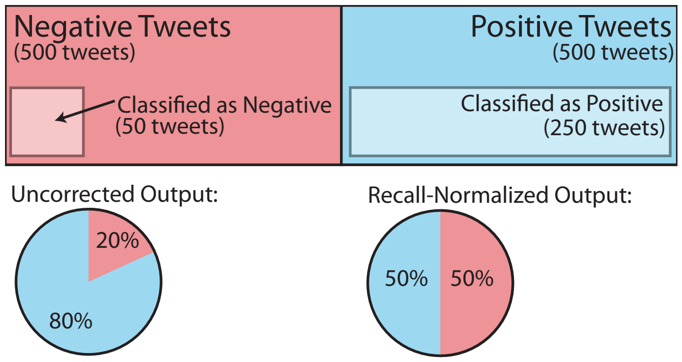


Figure 2. Positive and negative sentiment classifiers often have different performance. Suppose both the positive and the negative class contain 500 tweets. If the negative classifier is conservative and only identifies 50 tweets while the positive classifier is more liberal and identifies 250, the relative proportions will be biased (bottom left). If wedivide by an estimate of the classifier's recall (as a number in $[0 \dots 1]$), however, we produce a more accurate visualization.

This means that one classifier may be conservatively ignoring tweets while the other is accepting them. Following the example of Figure 2, if we have 500 positive and negative tweets, but the negative classifier has a recall of .1, whereas the positive classifier has a recall of .5, then the positive classifier will identify 5 times as many positive tweets as negative ones in aggregate. If we do not correct for this problem, the user will see an aggregate visualization that grossly overestimates the number of positive tweets in the data.

Our solution to this problem is to *recall-normalize* the aggregate numbers. We first adjust both classifiers so that they have equivalent precision values on a test dataset. We then measure their recalls on this set. When we count the number of positive tweets, we divide the count by the positive recall, and similarly normalize the negative count. For example, if we know recall is .1 and we observed 50 negative tweets, we estimate $50/.1 = 500$ negative tweets. This correction assumes that the precision/recall characteristics of our sentiment analysis algorithm is similar in practice as it is on the test dataset. As aggregated sentiment is important in many domains, we think that researching a principled solution to this problem will be important future work.

IMPLEMENTATION

TwitInfo layers an interface and infrastructure over the tweet stream. When an event is created, TwitInfo searches for the keywords for that event using the Twitter Streaming API. It associates tweets containing relevant keywords with events, counts tweet volume over time, and extracts metadata like sentiment and URLs. It records location metadata by harvesting geolocation tags on the tweet or, failing that, attempting to convert the freetext user location field into a latitude/longitude pair using a geolocation service.

These tweets are serialized to a database using Django, indexed by keywords that matched the streaming query. When the user requests a TwitInfo page, Django looks up relevant tweets using indexes on keyword and timestamp. To power

Data Source	Precision	Recall
Soccer Events	$\frac{17}{22} \Rightarrow 77\%$	$\frac{17}{22} \Rightarrow 77\%$
Soccer Events & Discussion	$\frac{21}{22} \Rightarrow 95\%$	$\frac{21}{26} \Rightarrow 81\%$
Major Earthquakes	$\frac{6}{44} \Rightarrow 14\%$	$\frac{5}{5} \Rightarrow 100\%$
All Earthquakes	$\frac{29}{44} \Rightarrow 66\%$	N/A
All Earthquakes & Discussion	$\frac{39}{44} \Rightarrow 89\%$	N/A

Table 1. Event detection performance on our data sets. Without a reasonable ground truth labeling of all earthquakes, we omit recall values for them. Note that there were five major earthquakes in the period of interest, but one event had two distinct peaks on Twitter. Precision and recall were equal for Soccer Events.

the visualization, we use the Google Visualization API [5] for our annotated timeline and pie charts, and generate a map using the Google Maps API.

EVALUATION

We now evaluate the performance of the algorithm for detecting events in the timeline, and the effectiveness of the user interface at visualizing events using Twitter data. We analyze the algorithm through a quantitative analysis of its ability to identify events, and analyze the user interface through a user study on twelve users and a semi-structured interview with a Pulitzer Prize-winning former Washington Post investigative reporter.

Algorithm Evaluation

We now determine if our automated event detection approach matches human intuitions about correct behavior. To support the evaluation, we gathered tweets from three soccer games and one month of earthquakes. To create ground truth for the soccer data, one researcher annotated major events in the soccer game using the game video and web-based game summaries, without looking at tweets. For ground truth on the earthquake data, we gathered data from the US Geological Survey on major earthquakes during the time period.

We tested how many of the events our algorithm detected were part of this ground truth set (*precision*), and how many events in the ground truth set our algorithm detected (*recall*). We used the default threshold cutoff for determining events — we can trade off precision and recall by adjusting this threshold, but we chose to use the single value that users will experience. Results are in Table 1.

The algorithm has high recall, finding all major earthquakes and most soccer events. Figure 3 displays several ground truth timelines and events that the algorithm identified. If the algorithm failed to catch an event, it was typically because Twitter volume did not peak when it occurred. For example, the only uncaught ground truth events in the soccer data set were yellow cards, which did not appear in large quantities in the stream, even though our researcher identified them as interesting events in his initial analysis of the game.

[5]http://code.google.com/apis/visualization/documentation/gallery.html

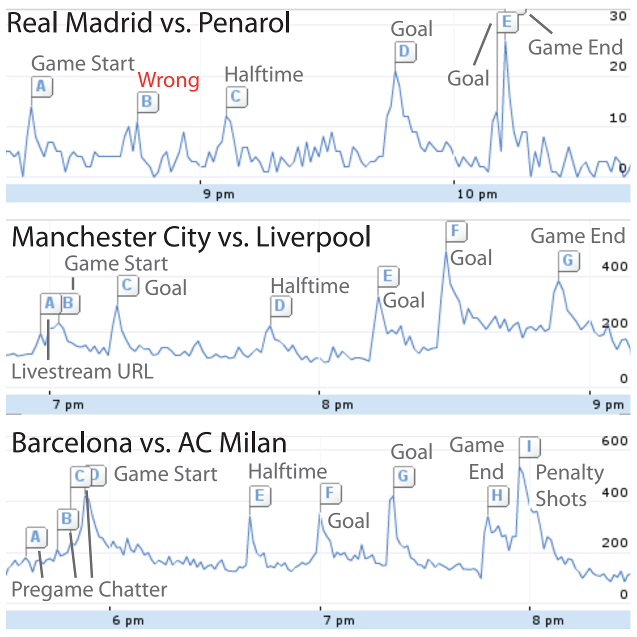


Figure 3. The TwitInfo timeline displaying three soccer games. The peak detection algorithm finds only 1 false positive, labeled red. The algorithm did not find Yellow Card events, but there was no spike in Twitter usage to suggest it. The algorithm occasionally flagged pregame chatter like retweeting a livestream URL.

Precision depended on activity type: it was high for soccer, but there were several false positives for major earthquakes because the algorithm also flagged minor earthquakes. If we include in our ground truth dataset all minor earthquakes, precision rises from 14% to 66% (Table 1). It still produces false positives, because Twitter spikes when its users discuss an earthquake or share information about donating aid money. If we include all earthquake-related discussion, precision rises to 89%. Likewise, including soccer-relevant discussion (like conversations identifying live video stream URLs for a game) leads to 95% precision in the soccer dataset. The remaining false positives come from high-volume topics that happened to mention the word earthquake or a soccer term. For example, one Twitterer asked celebrity Justin Bieber to follow her because she survived Chile's earthquake, and she was retweeted around Bieber's fan base.

The algorithm can be biased by Twitter's interests. If there is a medium-sized earthquake in the middle of the ocean, nobody will notice or tweet about it, so TwitInfo will miss the event. Conversely, if there is a minor earthquake in a densely populated area, lots of locals will tweet about it and TwitInfo may flag it as an event.

We also note two artifacts that occur because of a combination of our algorithm, timeline visualization, and use of Twitter as a datasource. First, a well-defined event, such as the September 2010 earthquake in Christchurch, NZ, often appears as multiple peaks on Twitter. In the Christchurch earthquake, we noticed three distinct peaks, all of which were identified by the algorithm: 1) news about the earthquake, 2) thoughts and prayers to those in Christchurch, and

3) information about aid and scams. The second artifact occurs when multiple events overlap in time. For example, two earthquakes in different locations might result in a single larger peak on the timeline. Our algorithm does not distinguish overlapping events, and our visualization, which does not provide stacked layers like MemeTracker [18] does not identify the two events as distinct.

User Interface Evaluation

Our next goal was to understand if TwitInfo users can understand an event's structure via the interface, and if some elements of the interface are more helpful for event exploration than others.

Method

We recruited twelve participants, six of whom had Twitter accounts. Ten had been to the Twitter website before. They could all explain Twitter's mechanics, including tweet length and broadcast messaging. Four participants were female. We compensated participants with $10 gift certificates.

We asked the participants to perform directed search tasks for the first half of the study. We chose tasks that exercised all the elements of the interface and allowed us to gather feedback on a variety of usage scenarios. Some example tasks on the earthquake dataset included:

- Find the earthquake that happened in Vanuatu. When did it occur, and what was the magnitude?
- How did sentiment about the earthquake differ between Vanuatu and New Zealand?
- What are people who live in Christchurch saying about the earthquake there?

During this process, we gathered usability feedback and observed which interface elements were useful or ignored.

The second half of the study was a time-limited exploration task. We gave participants up to five minutes to understand an event using TwitInfo and five minutes to dictate a news report on the event to the experimenter. To gain a variety of observations, we gave half of the participants a dataset of a soccer game between Manchester City and Liverpool, and half a dataset containing sixteen days of tweets about President Barack Obama.

Finally, we guided participants in a semi-structured interview. We asked about the best and worst parts of the TwitInfo interface. We also tried to understand the situations in which our participants would want to use TwitInfo, and asked them for alternate modes of performing their assigned reporting task in the absence of TwitInfo.

Results

Here is one participant's summary of President Obama's activities over a 16 day period:

> *After the peace talks, Obama traveled to the ASEAN conference and to NATO to work on issues in those parts of the world. He then spent the week dedicated to domestic economic issues. First he proposed a research tax break, then a $50 billion investment in infrastructure, then the issue came up about whether he should keep some tax breaks that Bush had implemented, and he's asking for some tax breaks from business, and these are generating some controversy because [. . .]*

Participants such as this one successfully reconstructed events from the Twitter timeline within five minutes, even without previous knowledge. TwitInfo gave quick, high-level understanding, with one user describing some of the visual elements as "at-a-glance" interfaces, and another praising its speed of distilling information, but explaining that the extracted information was "shallow."

Common Usage Patterns. When performing freeform exploration, users picked the largest peak and explored it carefully: they read the tweets thoroughly, drilled in on the map, and followed links to articles. Most users relied on tweets to confirm event details, though a few skimmed the automatically-generated labels. Under time pressure, users instead skimmed all peak labels to get a broad sense of the chronology. Self-reported "news junkies" tended to follow at least one link, either from a tweet citing an interesting news source, or from the popular links section. These users cited the lack of detail of the tweets as their reason for going outside of the system.

The Timeline Focused User Activity. When providing a high-level summary of the interface in the semistructured interview, most users cited the timeline and event labels as the most memorable and helpful elements. The timeline was also the most actively used component in the interface.

Mapping Requires Aggregation. The map provided helpful filtering features, but could have been even more useful if it offered aggregation. Users were excited to find that tweet density increased in areas affected by earthquakes, but this fact was not discoverable while the map was zoomed out. So, these users requested a map indicator for geographic areas with unusually high volume, for example a heatmap. Other participants wanted the bounds and zoom level of the map to act as a filter for the rest of the interface, for example re-sampling the timeline based only on tweets from Iran.

Users Do Not Trust Sentiment Analysis. We tested our sentiment classifier against previous work investigating tweet sentiment [15], finding similar performance. However, users did not agree with its model of sentiment. For example, users were often surprised to find positive overall sentiment for earthquakes. After examining the positive and negative sentiment tweets, they realized that many tweets contained positive messages such as "sending best wishes to those recovering from the earthquake in haiti." They would decide that the sentiment classifier was working correctly, but that it was not giving them the information they wanted. In essence, the sentiment of individual tweets did not summarize the overall sentiment of the topics they mentioned.

What Alternatives Did Users Suggest? We asked users how they would have performed their reporting task had TwitInfo not been provided to them. Users who successfully reported on the soccer match suggested, correctly, that they could have searched for an article about the game and extracted much of the same information. Users reporting on two weeks of Obama's activities had a harder time arriving at solutions. One user explained that he would have collected news weeklies (e.g., *Newsweek*), identified key events from each of President Obama's weeks, and then searched for articles on those events. Most of the other users explained that they would have searched Google News for historical day-by-day coverage, and compiled a set of the most interesting stories. All of the users had difficulty determining how, across the sets of daily search results, they would identify which results to pay attention to.

A Journalist's Perspective

Twitter has considerable buzz as a news medium [17]. We wanted to understand if TwitInfo would be a useful tool for journalists exploring a new topic. We conducted a semi-structured interview with a Pulitzer Prize-winning former Washington Post investigative reporter who is now a Knight Professor of Journalism. We asked the journalist to interact with the soccer, earthquake, and Obama examples, and to reflect on how journalists might use TwitInfo.

When presented with TwitInfo, the journalist initially focused on the labeled timeline view. She explained that the timeline with labeled events would be useful for *backgrounding* on a longer-running topic such as the sixteen-day Obama example. The journalist expanded: "When I do long stories, they are on a very arcane subject. Being able to see a timeline on an arcane subject could be really helpful." Specifically, she felt that the automated timeline would be useful to understand which substories to explore further. Additionally, she recommended blending the extracted URLs from the tweet stream with stories from traditional sources of news for an expanded understanding. In order to navigate longer timelines covering multiple subevents, she recommended a topic-based drill-down interface along the lines of what Eddi [4] provides.

The journalist was also interested in a separate use case for the map view of geolocated tweets, which could be used by reporters seeking to contact on-the-ground sources. The map view was compelling because "from the perspective of us covering a breaking event, you want to know what the people on the ground are saying," and zooming in on the location of interest could result in eyewitnesses to follow up with. The journalist drew an analogy to the *#iranElection*, which was billed as a Twitter revolution, but left reporters wanting to talk to the 200-odd people in Iran who were the eye-witness sources of on-the-ground information.

For the most part, the journalist's view on sentiment analysis was similar to that of the user study participants. She said that journalists would be skeptical of the quality and accuracy of the algorithm as well as the sample population expressing the sentiment. While aware of the research suggesting that extracted Twitter sentiment matches sentiment measured through other polling measures [20], she expressed skepticism that a journalist would trust these numbers.

Overall, the journalist found TwitInfo to be a compelling first version of a tool to make sense of the tweet stream from a reporter's perspective. She expressed that with her recommended interface modifications, she could imagine the tool being useful for reporting on breaking news, following real-time events, and familiarizing herself with overall interest and current opinions on an arcane topic.

DISCUSSION

TwitInfo successfully allows individuals to aggregate thousands of short messages into a coherent picture of an event. If a user or journalist wishes to receive a high-level background on a new topic, or if a reporter wishes to identify eyewitnesses to follow up with, our study and interview suggest that TwitInfo would be a useful tool.

We offer designers and developers of similar systems several takeaways from our experience. TwitInfo utilizes algorithms which enable configuration-free real-time social stream analysis. Prior work focuses on building useful dashboards on such data, but displays prototypes built on static datasets. To fully benefit from social streams, systems designers need to consider the streaming nature of the data when designing the algorithms behind their visualizations.

Our application of signal processing and streaming techniques to social streams opens the door to other uses in interfaces for social computing systems. Some examples include event-based notification (e.g., your tweet is suddenly getting lots of retweets), large text corpus summarization techniques (e.g., extending Diakopoulos et al. [6] to realtime scenarios), and smarter trend detection in newsfeed interfaces (e.g., extending systems like Eddi [4]).

Our evaluation presents a cautionary experience against some of the excitement in using sentiment analysis in tweet dashboards. Using sentiment analysis techniques from the literature, we showed that users are wary of sentiment-enabled exploratory tools of informative tweets due to a mental model mismatch. While this does not reduce the value of such signals in visualizations, it does suggest that sentiment is not an easy drop-in for such interfaces.

As a news source, our users found that Twitter was often only a shallow window into world events. Though analysis and conversation does appear on Twitter [13, 5], for many events Twitter is a medium for quick reactions and information sharing. Even the summary tweets sometimes could not explain the headline news: statistical summarization interfaces could help convey more meaning [25, 4, 21].

Our evaluation had several limitations. First-use studies make it difficult to understand how a person might use the interface longitudinally to track a topic. Such studies also have low external validity: could a citizen journalist use TwitInfo for real investigative reporting? We plan to release TwitInfo publicly to answer some of these questions.

CONCLUSION

Twitter is fast becoming a critical source of information about world events large and small. However, it is difficult to translate this information into a format allows users to draw higher-level conclusions. In this paper we present TwitInfo, a novel microblog-based event tracking interface that can collect, aggregate, and visualize tweets about user-specified events as they unfold on the stream. TwitInfo embeds a novel algorithm for peak detection and labeling, as well as a new technique for correcting aggregate sentiment displays. The peak detection algorithm identifies 80-100% of manually labeled peaks. TwitInfo users were able to understand several weeks' worth of Twitter data in a matter of minutes, on diverse topics like earthquakes, politics, and sports. A professional journalist reflected on the interface as a means of exploring complex topics and identifying eyewitness accounts. We discuss some implications of our approach for future designers and researchers to consider.

Aggregate Twitter interfaces promise a new set of exploratory data analysis tools. These tools are already being used for social science [27, 29] and augmented media experiences [23]. We envision a future where citizen journalists, citizen scientists and curious minds can use tools like TwitInfo to explore and understand their world.

REFERENCES

1. E. Adar, M. Dontcheva, J. Fogarty, and D. S. Weld. Zoetrope: interacting with the ephemeral web. In *UIST '08*. ACM Press, 2008.
2. P. André, M. L. Wilson, A. Russell, D. A. Smith, A. Owens, and m. schraefel. Continuum: designing timelines for hierarchies, relationships and scale. In *UIST '07*. ACM Press, 2007.
3. N. Bansal and N. Koudas. Blogscope: a system for online analysis of high volume text streams. In *VLDB '07*. VLDB Endowment, 2007.
4. M. S. Bernstein, B. Suh, L. Hong, J. Chen, S. Kairam, and E. H. Chi. Eddi: Interactive Topic-based Browsing of Social Status Streams. In *UIST '10*, 2010.
5. danah boyd, S. Golder, and G. Lotan. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *HICSS*. IEEE, 2010.
6. N. Diakopoulos, M. Naaman, and F. Kivran-Swaine. Diamonds in the rough: Social media visual analytics for journalistic inquiry. In *VAST 2010*.
7. N. Diakopoulos and D. A. Shamma. Characterizing debate performance via aggregated twitter sentiment. In *CHI 2010*.
8. M. Dörk, D. Gruen, C. Williamson, and S. Carpendale. A visual backchannel for large-scale events. *IEEE Transactions on Visualization and Computer Graphics*, 16:1129–1138, 2010.
9. S. Faridani, E. Bitton, K. Ryokai, and K. Goldberg. Opinion space: a scalable tool for browsing online comments. In *CHI '10*. ACM Press, 2010.
10. E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *WWW '04*. ACM Press, 2004.
11. D. Gaffney. #iranelection: Quantifying online activism. In *Web Science*, 2010.
12. Go, Alec and Bhayani, Richa and Huang, Lei. Twitter sentiment website, September 2010. `http://twittersentiment.appspot.com`.
13. C. Honeycutt and S. C. Herring. Beyond microblogging: Conversation and collaboration via twitter. In *HICSS '09*, 2009.
14. IETF Network Working Group. Rfc 2988: Computing tcp's retransmission timer, November 2000. `http://tools.ietf.org/html/rfc2988`.
15. B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Twitter power: Tweets as electronic word of mouth. *JASIST*, 60(11), 2009.
16. A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: Understanding the microblogging effect in user intentions and communities. In *WebKDD*, 2007.
17. H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW 2010*, 2010.
18. J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD '09*. ACM Press, 2009.
19. M. Naaman, J. Boase, and C.-H. Lai. Is it really about me?: message content in social awareness streams. In *CSCW '10*, New York, NY, USA, 2010. ACM.
20. B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *ICSWSM 2010*.
21. D. Ramage, S. Dumais, and D. Liebling. Characterizing microblogs with topic models. In *ICWSM '10*, 2010.
22. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
23. D. Shamma, L. Kennedy, and E. Churchill. Tweetgeist: Can the twitter timeline reveal the structure of broadcast events? In *CSCW 2010 Horizon*, 2010.
24. D. A. Shamma, L. Kennedy, and E. F. Churchill. Conversational shadows: Describing live media events using short messages. In *ICWSM '10*, 2010.
25. B. Sharifi, M.-A. Hutton, and J. Kalita. Summarizing microblogs automatically. In *NAACL '10*. ACL, 2010.
26. S. F. Silva and T. Catarci. Visualization of linear time-oriented data: A survey. *Web Information Systems Engineering*, 1:0310, 2000.
27. K. Starbird, L. Palen, A. L. Hughes, and S. Vieweg. Chatter on the red: what hazards threat reveals about the social life of microblogged information. In *CSCW '10*. ACM, 2010.
28. R. Swan and J. Allan. Extracting significant time varying features from text. In *CIKM 1999*.
29. S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *CHI '10*. ACM Press, 2010.