SigniTrend: Scalable Detection of Emerging Topics in Textual Streams by Hashed Significance Thresholds

Erich Schubert, Michael Weiler, Hans-Peter Kriegel
Ludwig-Maximilians Universität München
Oettingenstr. 67, 80538 München, Germany
http://www.dbs.ifi.lmu.de/
{schube,weiler,kriegel}@dbs.ifi.lmu.de

ABSTRACT

Social media such as Twitter or weblogs are a popular source for live textual data. Much of this popularity is due to the fast rate at which this data arrives, and there are a number of global events – such as the Arab Spring – where Twitter is reported to have had a major influence. However, existing methods for emerging topic detection are often only able to detect events of a global magnitude such as natural disasters or celebrity deaths, and can monitor user-selected keywords or operate on a curated set of hashtags only. Interesting emerging topics may, however, be of much smaller magnitude and may involve the combination of two or more words that themselves are not unusually hot at that time. Our contributions to the detection of emerging trends are threefold: first of all, we propose a significance measure that can be used to detect emerging topics early, long before they become "hot tags", by drawing upon experience from outlier detection. Secondly, by using hash tables in a heavy-hitters type algorithm for establishing a noise baseline, we show how to track even all keyword pairs using only a fixed amount of memory. Finally, we aggregate the detected co-trends into larger topics using clustering approaches, as often as a single event will cause multiple word combinations to trend at the same time.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.5.m [**Pattern Recognition**]: Miscellaneous—*Emerging Topic Detection*

Keywords

Trend Detection; Hashing; Text-Mining; Outlier Detection

1. INTRODUCTION

Social networks and microblogging services like Facebook and Twitter are known for the large amount of data published every second by their users. But also news agencies like Reuters and Bloomberg publish thousands of articles daily covering a wide range of topics. The information explosion calls for new tools and approaches to process this amount of data, as a single user can no longer read all the information available. Instead, automatic filters are used everywhere: email servers reject two thirds of our email traffic deemed as spam; the Facebook news feed is also heavily filtered: Facebook reported that the average user would have 1500 potential stories each day, out of which around 300 are automatically prioritized and shown by Facebook; yet most users even only read about half of these on average. In addition, some hard coded filters are used to e.g. merge all birthday wishes into a single story to reduce clutter.

Traditional text mining techniques such as clustering and topic modeling cannot be used trivially when processing a live stream of data. They can provide meaningful insight to refine e.g. a search query or to analyze a static text corpus, but are not applicable to fast-flowing, ever changing data streams. To cope with this flow of data, emerging topic detection is a useful tool, yet itself an emerging area. The goal of emerging topic detection is to identify new trends early, to notify the user of the evolving story. Intuitively the goal is to have an automated "breaking news" detection. When the user is able to customize and prioritize the data sources, this will ultimately allow personalized breaking news and trend detection, and corporations could use the additional knowledge to quickly react to future market needs and thus increase their profit. But the detection of trends in such fast-flowing data remains challenging because most documents contain raw unstructured natural language text; often abbreviated such that the interpretation is hard for both humans and computers. Ambiguity, homonyms and synonyms further add to these challenges. Even if some semantic annotations like tags are available, these are not necessarily helpful for the problem at hand, as they may be ambiguous as well and not all of them have an actual meaning [24]. Due to these problems, the resulting analysis quality still does often not meet the expectations in practise.

Scalability of textual analysis continues to be a problem. Many approaches are based on simple preprocessing such as stopword removal and stemming, and then use distributed algorithms to simply count and track word occurrences over time; very few methods seem to be able to extract meaning from sentences, correlate words and model complex information in near real time on high volume data [13]. This may explain why text analytics seems to take off very slowly.[1] As

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623740.

[1] `http://www.kdnuggets.com/2014/02/poll-results-te\xt-analytics-use-shows-no-significant-change.html`

marketing professor Hausman recently noted (commerical) sentiment analysis software often is less accurate than a coin toss [20], it may be necessary to first reduce the amount of data, then analyze this part thoroughly but manually.

In this article, we propose a statistics-based score for evaluating trends as well as scalability improvements to allow tracking arbitrary word pairs. The detected trends are then aggregated into a cluster to be further analyzed by the user.

2. RELATED WORK

The field of emerging topics detection receives much attention, as the daily flood of information available in social media and news agencies is impossible to overlook – in fact, the media are reporting on the increasing volume of data every day. To know what topics are currently "hot" could be an enormous advantage both for private uses and businesses. To solve these demands, commercial systems like Dataminr, Sysomos, Brandwatch, MediaMiser, and Topsy have been established. There are also a number of non-commercial systems like the CMU system [37], the UMass system [4], Meme-Detection System [26] or Blogscope [8]. Most of the published research focusses on Twitter as primary source of information inter alia due to its public streaming API[2] for accessing random samples of all public statuses and the contained hashtags, which help to reduce the dimensionality of the incoming text and could provide some additional semantic information.

First Story Detection (FSD) is a closely related task to emerging topic detection, with the subtle difference that the focus is to identify the actual first report of an event, not necessarily a topic gaining substantially in popularity. The CMU system [37] attempts to cluster continuously arriving news streams into groups that share the same event. They present an on-line method to accumulate events for a retrospective event detection. The UMass system [4] represents documents in a vector space such as a term frequency vector. Incoming documents are first compared to the database of previous documents using a nearest-neighbor search, then added to the corpus. Documents that differ enough from the most similar earlier document are considered to be first stories. This approach however does not scale well to large data sets such as Twitter due to the need to maintain the database corpus [29]. To scale this nearest-neighbor approach to Twitter streams, the use of locality sensitive hashing (LSH [21]) was proposed [29]. To limit database size, buckets are limited in size and the oldest document is removed from the bucket when the bucket would overflow.

Kleinberg [23] uses an infinite state automaton to model frequency bursts of incoming documents such as emails. Different states of the automaton correspond to different message frequencies, and a hierarchy is extracted from the state transitions. In Blogosphere [30], Kleinberg's approach was applied to the titles of blog posts. To improve performance, they had to omit keywords cooccurring with other keywords in the same title. In a post-processing stage they enrich bursty keywords with semantically correlated terms by looking at the five nearest neighbours using an Euclidean-based distance metric on the automatons' state series. Kleinberg's burst modeling was also used in [33], where it was applied to topics estimated by a dynamic topic model (DTM).

Cataldi et al. [12] proposed a method that extracts terms from tweets to model a life cycle inspired by biological processes. They define a term as *emerging*, if it is now frequent but was rare in the past. To improve results, they also consider social relationships (like the followers count) to determine the user authority. Although they do a post-processing step to enrich detected trending terms with semantically related keywords, they cannot detect when two keywords that have already been frequent before now are frequently comentioned such as when *obama* and *merkel* meet.

TwitterMonitor [27] first identifies bursty keywords that have a higher absolute frequency than usual and uses them as a seed to explore them further. Keywords that cooccur in the same tweets (within a small history window) are grouped together. For each such keyword set a singular value decomposition (SVD, [16]) is applied on all tweets containing them. The extracted terms are used to build a better description for each bursty keyword set.

EDCoW [35] applies a wavelet analysis on words to model their frequencies. Trivial words are identified with their cross correlation value. In Hip and Trendy [28], the focus lies on building a taxonomy from trends for a specific geographic area and to categorise them, rather than finding cooccurence trends. Similarly CiteSpace II [14] has a strong focus on visualizing emerging trends.

Density estimation on data streams as performed in stream clustering approaches such as CluStream [3] is also related to our approach. However these algorithms assume you have a stream of coordinates, and want to estimate the resulting coordinate-based density. Such vector data arises for example in sensor networks, where the need of compressing historical information [17] can be satisfied using wavelets, discrete cosine or histograms. Temporal velocity profiles [2] are then applicable to such data, and can be used to predict the positions of dense regions of moving objects. In our approach, we will be estimating a density on the temporal domain only, as if we would split the input stream into separate streams for each word and process them separately. These techniques therefore do not apply here.

The problem of top-k monitoring [7] is related, but it does not take relative significance into account. These approaches, often aimed at detecting denial-of-service (DOS) attacks in network flows, are interested in the most frequently occurring patterns only. In the context of monitoring textual streams, these approaches would likely monitor popular terms such as *justin* and *bieber*.

Exponential histograms [15] have been proposed for probabilistic counting as well as for weakly additive functions. This was then extended to continuously monitor variance over data streams [6] similar to our approach. However, we use a simpler yet effective approach: their techniques are designed with the requirement to take exactly the previous N observations into account. Our approach uses exponential weighting, which will never fully forget data, but aggregates historical values into a single figure, in which old results will eventually have a weight of effectively zero.

A rather similar approach to our system is enBlogue [5]: Like our approach, they are also interested in the cooccurrence of words, instead of relying on single terms. The central measure of their approach is Jaccard similarity [22] of two words in relation to their overall popularity. Like most other approaches processing Twitter data, they first preprocess the data, in order to extract hashtags and named entities as tokens. In order to find cooccurrences, they start

[2] `https://dev.twitter.com/docs/streaming-apis`

with a reduced seed list of frequent tokens; then track all token combinations that contain at least one of these seeds. For each token pair, a short history of ρ recent occurrence counts and popularities is kept; token pairs not seen during the last ρ epochs are discarded. Similar to our method, exponential smoothing is applied to predict future values from history. The score is computed based on the relative deviation from the estimate, weighted by logarithmic popularity:

$$score(t) = \frac{\big(x(t) - E[x]\big)/x(t)}{|\log\left(popularity(t)\right)|}$$

A key limitation of enBlogue is the need for controlling the memory usage. A number of mechanisms are employed to reduce the data volume: only hashtags and named entities are used for the analysis. Then seed tags are chosen based on a minimum occurrence threshold as well as a top-k filter. However, the experiments indicate that the accurracy drops linearly with the seed tags parameter, so these thresholds do not appear to be beneficial. Last but not least, since enBlogue keeps a history of 2ρ historical values for any pair of tags tracked, it scales badly with respect to memory usage. In our experiments, we observed as much as 700 million word and word pairs (see Table 1). While not all of them will be kept in memory at the same time, this requires a substantial amount of memory and update cost.

The method we propose improves over enBlogue in multiple ways. Instead of keeping a history, we only need two floating point values per record. Secondly, by storing this data only approximately in a hash table, we are not as much affected by the large number of potential word pairs. As we cleverly exploit hash collissions, we only need the hash table to be large enough to store all frequent word pairs, which due to the long-tailedness of the distribution is a substantially smaller number (as seen from the quantiles in Table 2). Petrović et al. [29] proposed a method for first story detection based on locality sensitive hashing. While our approach borrows on ideas from LSH and MinHash, we do not use this for nearest neighbor search, and do not need to store the individual documents for similarity search.

3. SIGNIFICANT TREND DETECTION

In information retrieval and text mining, a popular similarity measure for text is cosine similarity on the term frequency (TF) vectors, weighted by the inverse document frequency (IDF). This model is commonly referred to as TF-IDF vector model. Depending on the definitions used, term frequency can either be the absolute counts of each term, or the relative frequency. In the context of trend detection, such similarity measures are difficult to use: they are designed to quantify the similarity of two documents, but trending topics may span many documents, and documents will often cover more than one topic. In particular, topics may have subtopics; and within a larger topic (such as pop music) a subtopic (such as a particular artist) may be trending, even when the larger topic is not.

A naïve approach to perform topic detection would be the use of cluster analysis. However, cluster analysis on streaming data is far from a solved problem. Often, these clustering algorithms are unable to recognize hierarchies of clusters, and may need parameter fine tuning. Because of these limitations, the algorithms are mainly useful to cluster the result set of an information retrieval task for user presentation. Another similarly naïve approach maintains a database of recent documents, and searches the nearest neighbor (i.e. the most similar earlier document) for each new document. While this works reasonably well in a controlled corpus, it will fail on many noisy real data sets: while a low nearest-neighbor distance is indicative that the document is a near duplicate, the contrary does not hold: not every document will be the start of an interesting emerging topic, but it may also be just noise. In fact, nearest-neighbor distances are a popular measure of outlierness [31].

When detecting emerging topics in data streams, we cannot assume that we already have a cluster for the topic. In fact, it is desired to detect the topic as soon as possible. Yet at the same time, we are only interested in topics that both have a minimum size, but also show an "unusual" growth rate. When looking for trending subtopics, this becomes even more challenging – here, we may not be interested in the main topic, but only in the restriction to a subdomain.

In the following subsections, we will discuss some important aspects for emerging trend detection. First of all, we discuss how to measure *significance* of trends, as opposed to just some deviation score. Secondly, we will elaborate briefly on the *relationship to outlier detection*. Then we will discuss the importance of *word cooccurrences* for detecting trends that may be masked otherwise and additional challenges for *early detection* when using the suggested statistics. Finally, we will discuss *hashing* approaches for *scalability* to monitoring all pairwise cooccurrences.

3.1 Emerging and Trending Topics

When users see "trending topics", they usually assume that these are simply the most popular terms. However, users do not only want popularity, but they also expect *novelty*. Therefore, we must not simply look at the most popular tags, but we must take this popularity into its historical context. Furthermore, the absolute popularity is subject to seasonal trends. Depending on the data source, working hours and weekdays result in "seasonal" patterns that need to be accounted for. Relative popularity – normalized by the total number of documents for the day – proved much more robust in our experiments.

To evaluate the significance of a trend, we suggest to make use of statistical best practice such as the z-score (formally, the z-score assumes a normal distribution; while this will likely not hold, it nevertheless can serve as a reasonable heuristic for our purposes):

$$z(x) := (x - \mu)/\sigma$$

To use this on data streams, we need a moving average and moving standard deviation such as the exponentially weighted average ($EWMA$) and the associated standard deviation or variance ($EWMVar$):

$$sig(x) := \frac{x - EWMA}{\sqrt{EWMVar}} \tag{1}$$

in order to increase stability (the variance could be 0) as well as to account for non-interesting fluctuations of rare terms, we will ensure a minimum average and minimum standard deviation of β, which we call the bias term:

$$sig_\beta(x) := \frac{x - \max\{EWMA, \beta\}}{\sqrt{EWMVar} + \beta} \tag{2}$$

This bias term β not only avoids a division by 0, but also serves as a noise filter. For low volume data $EWMA < \beta$

with a small value of β we cannot statistically argue about trends. Intuitively, we should set β to the expected background noise level, i.e. how often rare terms occur naturally in the data stream, without trending. This bias term also takes into account that our input data is discrete – there are no "half occurrences" of words.

In order to estimate the average *EWMA* and the variance *EWMVar* for a data stream and a learning rate α, we can rely on earlier work by Welford [34] and West [36] on incremental mean and variance. The update equations given by Finch [18] for the exponentially weighted variants allow these values to be efficiently maintained on a data stream:

$$\Delta \leftarrow x - EWMA$$

$$EWMA \leftarrow EWMA + \alpha \cdot \Delta \tag{3}$$

$$EWMVar \leftarrow (1-\alpha) \cdot (EWMVar + \alpha \cdot \Delta^2) \tag{4}$$

The learning rate α can be set using the half-life time $t_{1/2}$; a parameter a domain expert will be able to choose easily based on his experience and needs:

$$\alpha_{\textit{half-life}} = 1 - \exp\left(\log\left(\tfrac{1}{2}\right)/t_{1/2}\right) \tag{5}$$

This update equation is best used with a fixed update cycle. While we could adjust the update cycle dynamically by adjusting α or $t_{1/2}$ accordingly, there are good reasons not to update too often: first of all, a fixed recomputation interval gives better performance guarantees, and secondly we have more control over the statistical validity of our estimates. We cannot use above equations to update the statistics on every arriving record, but we must perform some aggregation to estimate the popularity x reliably. When using a too high update rate, we will have more variance in our estimation of x, which will in turn harm the estimation of *EWMA* and *EWMVar*. When using a data source with a natural cycle, such as a news ticker having a daily pattern, we can expect best results aligning our updates with this cycle.

Note that we avoid the popular equation $E(X^2) - E(X)^2$ for estimating the variance, because this equation is prone to numerical instability due to catastrophic cancellation with floating point arithmetic. This instability may be the reason why variance on data streams seems to be rarely used yet. Equation 4 is numerically stable [18], and thus preferable.

3.2 Emerging Topics are Outliers

Emerging topics, when defined as "a set of documents which grows faster than expected from comparable other document sets", essentially are a specific kind of outliers. Schubert et al. [32] discuss a generalized local outlier detection model, and demonstrate its applicability to video streams. A similar interpretation for outlier detection is possible for our approach: for each topic (approximated as word or word pair), we compute a reference model consisting of an average frequency and variance based on an exponentially weighted *temporal neighborhood* set. We then compare the current frequency to this reference, and measure the trend by the deviation from this model.

The relationship to outlier detection not only exists on a formal level, but we are also seeing many problems typical to this domain [38]: we do not know beforehand what data we are searching for, and we do not have labeled training data so popular machine learning approaches such as random forests cannot be trained for this problem. Instead, we have to fight the problems of masking and swamping [9, 19], where for example a continuously popular word such as *obama* may prevent the detection of related trends, or a single strong trend may mask other trends in the data set.

3.3 Trend Detection on Cooccurrences

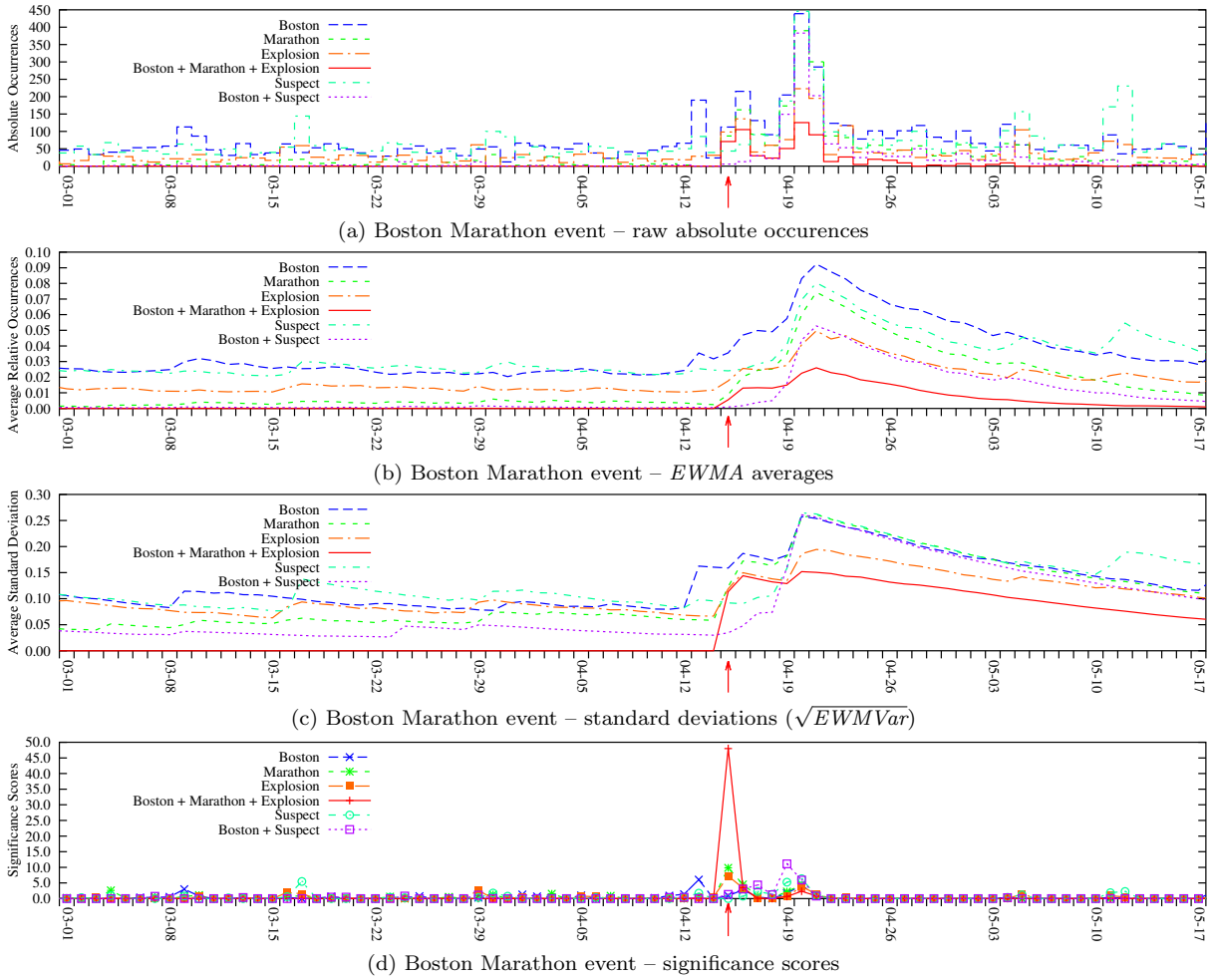
Equation 3 and Equation 4 can be applied to any numerical variable X. In order to apply this approach to trend detection in textual data, we need to represent the incoming data stream appropriately. We will not only build one variable for each word, but also for each word coocurrence. The reason is that we may be unable to reliably detect or analyze some trends on single words only. In Figure 1 we visualize the Boston Marathon bombing on news data. In Figure 1a the raw word occurrences are presented. The word *boston* shows a comparable peak three days before the Marathon. Figure 1b visualizes the EWMA moving average. In this figure, it becomes evident that *boston* is (unsurprisingly) frequently mentioned, but even *explosion* (as e.g. "the explosion of mobile traffic", "stock explosion" and "Dreamliner battery explosion") is occurring quite often. The exact combination however of these three words first occurs on April 15. Using the proposed significance measure, as visualized in Figure 1d, shows a 48σ event for this combination. For other related words, this event of global media interest also shows a peak, but not quite as strong. On April 19, we observe a similar peak for the combination *boston*, *suspect*; on this day the suspects were identified and the manhunt began. This example demonstrates the benefits of tracking word cooccurrences instead of single words.

3.4 Early Detection of Trends

Equation 3 and Equation 4 are not designed for continuous updating of the *EWMA* and *EWMVar* estimates, but we first need to have a robust estimate of the value of x. While we can easily adjust $\alpha_{\textit{half-life}}$ to dynamic time windows, the value of x will become noisy and exhibit a too high variance to be useful for trend detection. A simple but reliable approach estimating x is to use temporal slicing on natural cycles and volume of the data source (which may be a day for a news feed, or an hour for Twitter). Even when not updating *EWMA* continuously, we can still perform an online detection of trends. In order to evaluate the significance of a trend, it is even desirable to compare the current estimate of x with estimations based on delayed data only, i.e. we want to compare the observed value x with the predicted value *EWMA* based on the previous day, and not including the latest data in the prediction yet. By solving Equation 1 for x, we can obtain an alerting threshold τ if we fix a desired significance niveau s.

$$x > EWMA + s \cdot EWMVar =: \tau \tag{6}$$

Intuitively, when $x > \tau$, we are observing an $s \cdot \sigma$ significant increasing trend in the data. We do however need to acknowledge that this is not based on proper statistical hypothesis testing, so we cannot assume that 3σ events are rare: the $66 - 95 - 99.7\%$ rule (the three sigma rule) does not apply here, as we will be monitoring thousands of trends in parallel and have some margin of error; on the other hand we are interested in seeing multiple event every day, and not just the most significant events of the year. This approach can be seen as a variant of Bollinger Bands as used in stock



(a) Boston Marathon event – raw absolute occurences

(b) Boston Marathon event – $EWMA$ averages

(c) Boston Marathon event – standard deviations ($\sqrt{EWMVar}$)

(d) Boston Marathon event – significance scores

Figure 1: Visualization of selected word occurrences for the Boston Marathon bombing on news data

market analysis, except that our data sources do not exhibit the fast negative feedback loop driving the stock market.[3]

The main difficulty with this approach is getting a robust estimate of x while the epoch is not yet complete. On the news data set, for example, it is common not to see any news posted before 6am in the morning. The first news item each day will naturally produce terms with a naïve document frequency of 1; yet they do not constitute a reliable trend so far. However, we also do not know how many news will be posted during the day in total. Therefore, in order to estimate x, we need to learn to predict the number of documents to be posted during the epoch; we can then compare the absolute number of occurrences, the number of documents seen so far, and the number of documents expected to obtain a better estimate of x.

3.5 Scalability by Hashing

Although Equation 3 and Equation 4 are efficient to compute, we would still need to perform this for any word pair. As seen in Table 1, this is not scalable. Instead of restricting the set of candidates to monitor beforehand, we propose to use a probabilistic approach based on hash tables. The popularity of words follows a long-tailed distribution: the majority of words has a low occurrence rate in the data. This property can be exploited using clever hashing techniques. Our approach is related to heavy-hitters algorithms such as Bloom filter [10] and MinHash [11]. Instead of probabilistic set membership testing, our hashing scheme is designed for providing an upper bound of the $EWMA$ and $EWMVar$ values. We use 2^ℓ buckets (each storing an $EWMA$ and $EWMVar$ value), and k hash functions, so that each word is mapped to at most k buckets. When updating the $EWMA$ hash tables – when transitioning from one epoch to the next – we first hash each word (with a frequency larger than β) into its k buckets. For each bucket we track the *maximum* popularity x observed in the current epoch. Next, we update the $EWMA$ and $EWMVar$ values in the hash table with the *maximum* x observed for all words in this bucket only. When computing the alerting threshold τ for an observed word w, we again inspect the k buckets given by the hash functions. The threshold is then obtained by taking the *minimum* alerting threshold of all buckets (Equation 6). Assuming there exists at least one hash bucket where the candidate word w has the maximum popularity x_w of all words in this bucket, then the $EWMA$ value stored in this bucket will not be overwritten by a different word w'; and at the same time none of the k buckets was last updated with a lower popularity

[3]It is therefore unlikely that the proposed method is beneficial for stock market analysis.

than x_w. Otherwise, i.e. if in each of the k buckets there was a hash collission with a more popular word w', we will overestimate the *EWMA* and *EWMVar* values, and we may miss an emerging trend. To avoid this, we need to choose ℓ large enough, such that the majority of bins are not filled with frequent keywords.

Using this hashing strategy, we can control the amount of memory and computation needed for maintaining the trend statistics very well. In fact, this allows us to scale our approach beyond tracking single word (or hashtag) occurrences to monitoring even word-pair occurrences in large data streams. Word-pair occurrences prove very effective in our experiments at detecting subtopics. For example when Edward Snowden traveled from Hong Kong to Moscow, all of the individual words {*edward*, *snowden*, *hong*, *kong*, *moscow*} themselves have not been trending (events surrounding Edward Snowden have continuously been in the media at that time), but various combinations of these words such as *snowden* and *moscow* exhibit a significant peak.

The use of hashing yields a number of benefits. First of all, updating the statistics becomes a vectorized bulk operation. Secondly, the memory usage is constant, and the data can easily be serialized either for transmission to a different system, but also for checkpointing and crash recovery. After loading the last checkpoint, either a replay of the latest data can be used to restore the exact state, or the system can even decide to only process new data, and rely on the statistics to recover (it may then be desirable to disable alerting for this epoch).

3.6 Trend Redundancy and Refinement

A secondary challenge is the redundancy of trends. Assuming that a word such as *snowden* is trending, then we will likely see other related words such as *edward* also trend at the same time. But we may also see uninteresting combinations such as {*snowden*, *the*} trend significantly compared to previous usage of this word combination. Probably the most important step here is to perform stopword removal: by not including known stopwords in the trend analysis, we both have to analyze much less pairs, but we will also remove a large share of uninteresting co-trends. In addition to a language-specific set of stopwords, it is also beneficial to include domain specific stopwords, such as *follow*, *tweet*, *retweet* (*rt*), *lol* and *twitter* for Twitter.

In the refinement phase, we can use an inverted index to both verify observed trends (as the hash table may have had a collission), but also compute the overlap between any two words involved in the detected trends. For this we employed the clustering toolkit ELKI [1], and used hierarchical clustering with Ward linkage. Similarity is measured by how significant the two words trend together. We cannot assume that related words form a clique: consider for example *fiscal*, *cliff*, *barack* and *obama*. The last two will usually not qualify as a trending topic, but the names are in fact one of the most mentioned words in news on any day; however all four together form a known topic.

Alternatives at this stage – which we plan to investigate in future work – include finding maximum-weight cliques (as used in [25]) and topic modeling techniques such as pLSI and LDA. However, topic modeling is not guaranteed to produce a more meaningful output either [13].

In Algorithm 1 we give a pseudocode for the overall detection process, but we have to omit some details for brevity.

Algorithm 1: Document Processing

```
Data: epoch Epoch identifier
Data: EWMA[c] Averages hash table
Data: EWMVar[c] Variance hash table
Data: h_i Hash functions
Input: s Threshold for refinement / alerting

open index[epoch] shard for writing
initialize frequency map
initialize stats map
/* Index a new document                                   */
foreach doc in current epoch do
    foreach unique word and word-pair in doc do
        add doc.id to index[epoch][word]
        increment frequency[word]
        /* Get word statistics from hash table            */
        if not stats[word] then
            (μ, σ) ← (∞, ∞)
            foreach hash function h_i do
                c ← h_i(word)
                if EWMA[c] < μ then
                    μ ← EWMA[c]
                    σ ← sqrt(EWMVar[c])
            stats[word] ← (μ, σ)
        (μ, σ) ← stats[word]
        /* Test for significance threshold                */
        x ← estimate frequency of word
        if (x − max(β, μ))/(σ + β) > s then
            send to refinement for early alerting

/* Perform end-of-day analysis                            */
initialize trending topics list
foreach unique word and word-pair in frequency do
    (μ, σ) ← stats[word]
    /* Test for significance threshold                    */
    x ← frequency[word]/|documents|
    if (x − max(β, μ))/(σ + β) > s then
        add word to trending topics
close index[epoch] shard for writing
Refine trending topics
Cluster trending topics
Produce end-of-day report

/* Update the statistics table for next epoch             */
/* Aggregate into maximum for each bucket                 */
initialize update-table
foreach word and word-pair in frequency do
    frequency ← frequency of word
    foreach hash function h_i do
        c ← h_i(word)
        if frequency > update-table[c] then
            update-table[c] ← frequency
/* Update statistics table                                */
foreach hash code c do
    freq ← update-table[c]/number of documents
    Δ ← freq − EWMA[c]
    EWMA[c] ← EWMA[c] + α · Δ
    EWMVar[c] ← (1 − α) · (EWMVar[c] + α · Δ^2)
/* End epoch                                              */
```

4. EXPERIMENTS

We demonstrate the scalability of our system as well as its ability to detect statistically significant trends in real data sets, without the need to reduce the candidate set as required in related work such as enBlogue [5]. For all data sets, we employed best practises for text mining such as language-specific stemming using the Xapian search engine library.[4] We also removed a standard set of English stopwords as well as domain specific stopwords (e.g. *retweet* for Twitter).

The implementation of the main analysis (not including the web spider and preprocessing) was done in Java using a single thread and Apache Lucene[5] as backing index. To save disk space and comply with copyright requirements, we did not store complete documents for the news data set, but only inverted lists, the URL and the headline.

4.1 Data Sets

For our experiments we focused on three data sets that exhibit very different charactersistics, as it can be seen from Table 1 and Table 2. The first data set, consisting of news articles, contains much fewer documents, but the documents obviously are much longer than those of the Twitter data set. The longer paragraphs of the news articles then yield even more word pairs than the Twitter data. But there is also a more subtle difference than the size. Where Twitter contained 25 million unique tokens, 99% of these occurred less than 14 times in the data set. In the news data, the vocabulary was much more evenly used, with the top 1% of words occurring 3476 times or more.

4.1.1 News Articles

We used a web spider to index news articles from popular news agencies such as Reuters and Bloomberg as our first data source. We limited the index process to the year 2013 plus a window of 10 days. As these news contents have restrictive copyright requirements, we must not store the full documents, but only use the data to perform trend detection and construct a search index for refinement; but the original article can be accessed on the publisher's website.

Nevertheless, the news article corpus is very interesting, because it is editorially managed, and of a different nature than Twitter: where Twitter users tend to re-share the exact same text to their followers, news agencies try to avoid duplication. For some trending topics, we do however see "update" documents in this data source. For the use case of a corporate user interested e.g. in financial trends, news agencies may be the more interesting data source than Twitter. Methods such as enBlogue [5] cannot be applied on this data set without modifications, as it relies on hashtags to seed its search process. Our method does not have such restrictions, but monitors all frequent word pairs.

4.1.2 Twitter

Because of the low latency and high volume, Twitter has become one of the most popular data sources for trend detection on the Internet. Events such as the Arab Spring led to a wide acceptance of Twitter as a medium capable of real-time monitoring. As such, this data source is an interesting complement to the news organization data sources we used for most of our experiments: Twitter data is even noisier and larger in volume, but it is also prone to bias due to its non-representative user base. We used Twitter's public streaming API at the "Spritzer" sampling rate (approximately 1% of tweets) to process about 279 million tweets over a period of 114 days. For analysis purposes we only used English language tweets, removed all retweets (about 10%) and used a simple near-duplicate detector to remove obvious spam and bulk (about 1.5%), which yields 94 million tweets to analyze. When including *#hashtags* and *@usermentions* as tokens, the results were dominated by teen idols, which are massively "spammed" by their fans. But as these tweets include few other words (except stopwords such as *ilysim*) they become essentially empty when skipping these tags. Without *#hashtags* and *@usermentions*, the analysis become more focused on the textual content, and thus the results became much more interesting (as seen in Table 4).

4.1.3 Stack Overflow

Stack Overflow is a Q&A website, which publishes its data under a creative commons license.[6] We analyzed the contents of the main website, years 2010 to 2013, totaling to 5.9 million questions. There were few significant trends in this data set, corresponding to important software releases (OS X Mavericks, OS X Mountain Lion, iOS6, iOS7, TypeScript), the `facebook.stackoverflow.com` launch and holidays (new years, christmas), so we omit details for brevity (included in the online demonstration).

4.2 Manual Analysis of Real World Trends

Table 3 examines the top 50 most significant trends in the news data set for the year 2013. For brevity, we omitted economic and sport news outside the top 10. Finance and sports dominate the results due to their extensive coverage in these data sources. In particular, "game days" of sports leagues trend every week due to some new word pair combinations. This may require future work to automatically classify the detected trends into categories such as sports. Nevertheless, a number of events also made it into the top 40, such as the Boston Marathon bombing discussed before, the algeria gas plant attack and Syrias use of chemical weapons.

In Table 4 we also discuss the top 40 trends in Twitter detected. All of these are easily verifiable with Internet search. Detected trends consist of single words, up to almost complete sentences by Twitter standards. Despite not using retweets, many topics such as the Oscar selfie (the most retweeted tweet so far) still surfaced, because a substantial amount of users add a reply text to their tweet.

4.3 Scalability

Due to the use of hash tables, we were able to run the experiments on a single i7-3770 CPU core without the need to use distributed computation or multiple threads. For the news data set, the preprocessed data volume was 1.1 GB (2.6 MB per day on average). Analyzing a year of data took 18 minutes, i.e. 2.5-3 seconds on average per day. The refinement index only stores the headlines, for which we need only 745 KB per day, and 305 MB total for a whole year.

The raw 1% Twitter input data was 228 GB compressed, after filtering retweets and non-English tweets, and preprocessing, 6.7 GB remained. Processing Twitter took 1.5 hours; 46 seconds on average per day. The actual analysis

[4]Open-Source, `http://xapian.org/`

[5]Open-Source, `https://lucene.apache.org/`

[6]`https://archive.org/details/stackexchange`

Table 1: Data set statistics (after stopword removal)

Data set	Documents	Paragraphs	Unique Words	Total Words	Unique Pairs	Total Pairs
News	424,704	5,867,457	300,141	56,661,782	71,289,359	660,430,059
Twitter	94,127,149	94,127,149	25,581,022	245,140,695	179,105,233	473,871,456
StackOverflow	5,932,320	30,423,831	2,040,932	138,205,636	91,460,397	545,570,530

Table 2: Vocabulary statistics (after stopword removal)

	Word Frequency			Pair Frequency		
Data set	Median	90% Quantile	99% Quantile	Median	90% Quantile	99% Quantile
News	3	81	3476	2	18	124
Twitter	1	at 92.9: 2	14	1	at 91.3: 4	29
StackOverflow	1	at 90.1: 7	185	1	at 90.9: 5	73

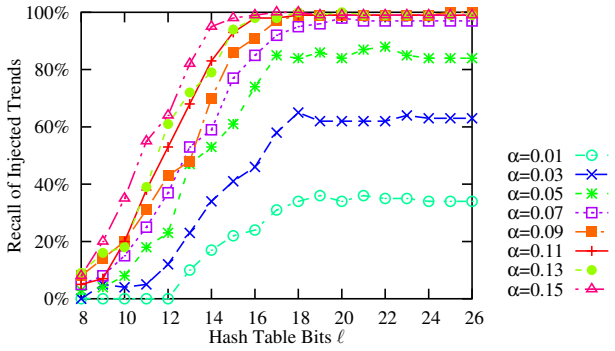


Figure 2: Performance with varying hash table size ℓ, $k = 4$

ran in 48 minutes; 25 seconds per day. The inverted index for Twitter occupied 18 GB of disk space, storing about 160 MB per day of Twitter data. Therefore, this approach should be scalable to the full Twitter "firehose" stream. In particular, there are a number of easy ways for distributing the load: for example, building the backing index, maintaining the hash tables, and refining the detected events can easily be distributed on separate systems. As the backing indexes are sharded already, the shards can also be distributed onto different machines trivially. In fact, we were able to process the news data sets on a Raspberry Pi (700 MHz ARM CPU, 512 MB RAM) at a processing speed of 104.1 seconds on average per day.

4.4 Hash Table Size

For the hash table sizes, we expect a saturation effect to happen. Too small hash tables will lead to masking and swamping [9, 19]; but once the hash table has become big enough to not have collisions, the *EWMA* estimates are expected to be good. To verify this, we used semi-synthetically data sets derived from the news data sets. Into these, we injected artificial words following a narrow Poisson distribution with $\lambda = 2 \ldots 9$ weakened by a constant factor of α. After then random onset of the artificial trend, we modified each document with a probability of

$$\alpha \cdot pmf_{Poisson}(\lambda, k) = \alpha \frac{\lambda^k}{k!} e^{-\lambda}$$

Due to the randomization of λ, some trends will be easier to spot, others will be harder. Furthermore, both the existing natural trends in the data and injected trends may mask the injected trends. The maximum value of the probability mass function is 0.27, so for $\alpha = 0.15$ about 4% of documents were modified to include the artificial keyword. For $\alpha = 0.01$, the chance of trend injection drops to 0.27%. Figure 2 summarizes the results on the injected trends data using $k = 4$ hash functions. As you can see, at $\alpha > 0.05$ the artificial trends are detected reliably, and the hash tables show a typical saturation effect, where the performance no longer increases once we have reached a reasonable size. A hash table of 20 bits requires just 2^{25} bytes of memory (32 Megabytes).

4.5 Online Demonstration

The results of our trend detection system are available at `http://signi-trend.appspot.com/` for exploration.

5. CONCLUSIONS

In this article, we first discussed the use of exponentially weighted floating averages and variance to score a trending topic with respect to its recent occurrences in the data stream. The proposed statistic needs litte memory – two floating point values only – and can be efficiently updated using an incremental equation. The numerical properties of this equation are well understood, and it was shown to be more stable than the naïve equation of the variance involving a difference of squares. This improved scoring function is able to capture the background noise as well as general trends in the data, and by measuring variability it can produce a much more meaningful significance score than e.g. nearest-neighbor distances that were used in state-of-the-art prior work to measure emerging topics.

Secondly, we showed how to scale our statistical approach to monitoring every word and word pair of a data stream with limited memory, by using hashing techniques and exploiting that the majority of words and word pairs occurrs only rarely in the stream. By resolving hash collissions in favor of the more common word or word pair, we will usually have an exact statistic for these words and only occur information loss on rarely seen words – which then by definition are not trending yet. Scalability of this approach was demonstrated by analyzing state-of-the-art data sets faster than real time on a single CPU.

Third, we suggested and demonstrated the use of clustering techniques to aggregate the observed trends – which usually only affect a small subset of the vocabulary, for which the results can be refined from an inverted index – into larger trends. This has become more important than in previous work, as we monitor a much larger set of candidates; and in particular cooccurrences tend to overlap and form clusters.

To the best of our knowledge, this is the first system that can monitor all word pair cooccurrences on a large data stream without the need for parallelization, whereas previous work can only monitor a filtered set of terms or word pairs, based on a seed set. Such a restriction is no longer necessary with the memory reduction obtained via hashing.

Table 3: Excerpt of top 50 trends on news data set 2013 (dominated by economy, sports and politics)

Score	Date	Stemmed Keywords (excerpt, edited for readability)	Explanation
58	11-18	thomson text summary 3000xtra alert outperform eikon	Reuters artifact – 233 research alerts
13	10-09	janet yellen ben bernank vice barack obama nomin	Obama nominates Yellen as U.S. Fed Chief
10	02-14	heinz buffett gdp hj merger shrank berkshir hathaway warren	Berkshire Hathaway, 3G Capital buy Heinz
9.7	07-03	turmoil armi unrest egyptian lisbon egypt mursi portug bailout	Yen rises because of turmoil in Egypt, Portugal
9.4	04-19	boston search bomb suspect marathon manhunt	Boston Marathon suspect manhunt
9.3	04-15	boston explos marathon	Boston Marathon bombing
9.3	01-28	durabl caterpillar pend	Four-month high oil, strong durable goods
8.8	09-19	inning era	Baseball end of season reports
8.8	02-26	ben bernank testimoni defend deadlock stalem	Bernanke defends bond-buying, Italian stalemate
8.6	07-11	ben bernank minut accommod forese dovish	Bernanke reassures euro bonds markets
		Economical, financial and sports news largely omitted outside the top 10 for brevity	
8.6	04-16	finish sharp metal boston marathon rebound terror bomb explos injur	Boston Marathon attack details
8.6	03-25	rescu cyprus bailout eurogroup guarante relief dutch jeroen dijsselbloem	EU cyprus bailout
8.5	01-17	hostag desert milit algeria algerian	Algerian gas plant attack
8.5	09-03	finnish wireless handset nokia smartphon microsoft	Microsoft buys Nokia's handset business
8.3	01-23	davo forum switzerland cameron referendum	Cameron promises referendum to leave EU
7.8	11-24	atom reactor iranian geneva enrich breakthrough lift uranium	Breakthrough on Iran nuclear activity
7.8	08-26	chemic weapon secretari holiday durabl	Kerry comments on Syria over chemical weapons
7.7	05-21	oklahoma moor tornado	Moore, Oklahoma hit by deadly tornados
7.6	09-23	merkel angela coalit victori william dudley shutdown	German elections succes for Angela Merkel
7.3	10-01	deadlock shutdown midnight began trillion unpaid barack obama	U.S. government partial shutdown
7.2	08-27	syrian chemic weapon strike assad tension kerri secretari	Escalations in Syria
6.9	04-20	dzhokhar tamerlan tsarnaev brother suspect captur dead watertown injur	Boston Marathon suspects captured
6.8	07-24	appl iphon smartphon markit flash beat faster	Surge in iPhone sales
6.8	12-06	nelson mandela die apartheid african africa	Nelson Mandela died
6.7	02-15	moscow communiqu	G20 finance ministers in Moscow
6.7	04-08	thatcher iron margaret	Margaret Thatcher died
6.6	07-12	airport dreamlin ethiopian heathrow boe	Boeing Dreamliner catches fire

6. REFERENCES

[1] E. Achtert, H.-P. Kriegel, E. Schubert, and A. Zimek. Interactive data mining with 3D-Parallel-Coordinate-Trees. In *Proc. SIGMOD*, pages 1009–1012, 2013.

[2] C. C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proc. SIGMOD*, pages 575–586, 2003.

[3] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proc. VLDB*, pages 81–92, 2003.

[4] J. Allan, V. Lavrenko, D. Malin, and R. Swan. Detections, bounds, and timelines: UMass and TDT-3. In *TDT–3*, pages 167–174, 2000.

[5] F. Alvanaki, S. Michel, K. Ramamritham, and G. Weikum. See what's enBlogue: real-time emergent topic identification in social media. In *Proc. EDBT*, pages 336–347, 2012.

[6] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proc. PODS*, pages 234–243, 2003.

[7] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proc. SIGMOD*, pages 28–39, 2003.

[8] N. Bansal and N. Koudas. Blogscope: a system for online analysis of high volume text streams. In *Proc. VLDB*, pages 1410–1413, 2007.

[9] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley&Sons, 3rd edition, 1994.

[10] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[11] A. Z. Broder. On the resemblance and containment of documents. In *Proc. SEQUENCES*, pages 21–29, 1997.

[12] M. Cataldi, L. D. Caro, and C. Schifanella. Emerging topic detection on Twitter based on temporal and social terms evaluation. In *Proc. MDM/KDD*, page 4, 2010.

[13] J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei. Reading tea leaves: How humans interpret topic models. In *Proc. NIPS*, volume 22, pages 288–296, 2009.

[14] C. Chen. CiteSpace II: Detecting and visualizing emerging trends and transient patterns in scientific literature. *ASIS&T*, 57(3):359–377, 2006.

[15] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SICOMP*, 31(6):1794–1813, 2002.

[16] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *ASIS*, 41(6):391–407, 1990.

[17] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *Proc. SIGMOD*, pages 527–538, 2004.

[18] T. Finch. Incremental calculation of weighted mean and variance. Technical report, University of Cambridge, 2009.

[19] A. S. Hadi, A. H. M. Rahmatullah Imon, and M. Werner. Detection of outliers. *WIREs Comp. Stat.*, 1(1):57–70, 2009.

[20] A. Hausman. The sad state of sentiment analysis. Online: `http://www.hausmanmarketingletter.com/sad-state-sentiment-analysis/`, 2013.

[21] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. STOC*, pages 604–613, 1998.

[22] P. Jaccard. Distribution de la florine alpine dans la Bassin de Dranses et dans quelques regiones voisines. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

Table 4: Top 40 trends on Twitter data (dominated by celebrities and teen idols)

Score	Date	Keywords	Explanation
174	03-06	boosi releas jail	Rapper Lil Boosie released from jail early
154	05-28	rip author poet inspir angelou maya peac dr	Civil rights activist Dr. Maya Angelou died
127	05-12	elev jayz attack jay solang beyonc	Solange, Jay Z and Beyonce elevator incident
98	03-03	ellen degener host selfi pizza	Ellen's Oscar Selfie and Pizza
76	05-22	ewok	Band 5SOS changed its name on Twitter to "Ewok Village"
76	03-21	bracket mercer duke	Mercer surprise win over Duke in March Madness
73	05-24	ronaldo bale gareth	Champions league final
63	04-07	geldof dead rip peach	Peaches Geldof died of heroin
61	04-15	moon eclips lunar blood	Blood moon (lunar eclipse)
60	05-05	shovel	Snow in May + "shovel girl fight" viral video
51	05-24	ramo sergio	Champions league final
51	04-14	zac efron	Zac Efron shirtless at the MTV movie awards
50	03-17	punch pinch green	St. Patricks Day
50	05-06	mimi	Mimi Faust and Nikko announce private video
49	03-17	lizzi	Lizzie in the series The Walking Dead
48	02-19	angela	Talking Angela hoax
47	04-29	ban silver adam clipper donald sterl nba owner	Donald Sterling banned for life from NBA
46	05-15	sm exo kris	Rumor of Kris (of Exo-M) parting
43	05-10	austria	Austria wins the Eurovision song contest
43	03-03	arm bradley longer cooper	Oscar selfie: "If only Bradley's arm was longer. Best photo ever"
43	02-24	rip harold ghostbust rami	Actor Harold Ramis died
42	04-10	sibl nation	National siblings day
42	03-09	arsenal wigan	Arsenal vs. Wigan Athletic in FA cup
41	04-13	million matt	Matthew Espinosa follow spree (#mattTo1Mil)
41	03-17	earthquak	4.4 earthquake strongly felt in Los Angeles
41	05-09	billionair dre dr beat billion appl	Apple buys Dr. Dre
40	05-10	eurovis	Eurovision Song Contest
38	05-06	bambi	Love & Hip Hop Atlanta Premiere with Bambi
38	03-19	ezra dead	Ezra gets shot in series "Pretty little liars"
38	04-16	bale gareth goal	Winning goal in Copa del Rey final
38	03-03	actor oscar mcconaughey matthew leo leonardo dicaprio	No Oscar for Leonardo di Caprio, again
37	05-06	scrappi	Love & Hip Hop Atlanta Premiere with Scrappy
36	05-09	cleveland johnni manziel dalla footbal draft brown	Johnny Manziel draft in NFL
35	04-30	rip bob actor hoskin	Actor Bob Hoskins died
35	04-26	racist donald clipper sterl owner	Donald Sterlin accused of racism (see above!)
35	03-18	allison	Allison in TV series "teen wolf"
34	04-02	warn tsunami chile magnitud earthquak quak	Chile 8.2 magnitude earthquake
34	05-21	cav	Cleveland Cavaliers win NBA draft lottery again
34	05-06	joselin	Love & Hip Hop Atlanta Premiere with Joseline Hernandez
34	03-14	pie pi	π day (3.14)

[23] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Min. Knowl. Disc.*, 7(4):373–397, 2003.

[24] D. Laniado and P. Mika. Making sense of Twitter. In *ISWC*, pages 470–485. Springer, 2010.

[25] T. Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras. On the spatiotemporal burstiness of terms. *PVLDB*, 5(9):836–847, 2012.

[26] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proc. KDD*, pages 497–506, 2009.

[27] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the Twitter stream. In *Proc. SIGMOD*, pages 1155–1158, 2010.

[28] M. Naaman, H. Becker, and L. Gravano. Hip and trendy: Characterizing emerging trends on Twitter. *ASIS&T*, 62(5):902–918, 2011.

[29] S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In *Human Language Technologies*, pages 181–189, 2010.

[30] M. Platakis, D. Kotsakos, and D. Gunopulos. Searching for events in the blogosphere. In *Proc. WWW*, pages 1225–1226, 2009.

[31] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. SIGMOD*, pages 427–438, 2000.

[32] E. Schubert, A. Zimek, and H.-P. Kriegel. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Min. Knowl. Disc.*, 28(1):190–237, 2014.

[33] Y. Takahashi, T. Utsuro, M. Yoshioka, N. Kando, T. Fukuhara, H. Nakagawa, and Y. Kiyota. Applying a burst model to detect bursty topics in a topic model. In *Adv. NLP*, pages 239–249, 2012.

[34] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[35] J. Weng and B.-S. Lee. Event detection in Twitter. In *ICWSM*, 2011.

[36] D. H. D. West. Updating mean and variance estimates: an improved method. *Commun. ACM*, 22(9):532–535, 1979.

[37] Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *Proc. SIGIR*, pages 28–36, 1998.

[38] A. Zimek, E. Schubert, and H.-P. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Stat. Anal. Data Min.*, 5(5):363–387, 2012.