Burst Detection in Social Media Streams for Tracking Interest Profiles in Real Time

Cody Buntain
Dept. of Computer Science
University of Maryland
College Park, Maryland 20742
cbuntain@cs.umd.edu

Jimmy Lin
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario
jimmylin@uwaterloo.ca

ABSTRACT

This work described RTTBurst, an end-to-end system for ingesting a user's interest profiles that describe some topic of interest and identifying new tweets that might be of interest to that user using a simple model for bursts in token usage. We laid out RTTBurst's architecture, our participation in and performance at the TREC 2015 Microblog Track, and a post hoc analysis for increasing RTTBurst's performance. While not as relatively performant in the Microblog Track's real-time notification task, RTTBurst did perform well (ranking 4th overall and second in the automatic category of Scenario B) in providing daily summaries for various interest profiles. Following the official TREC evaluation period, we were also able to increase RTTBurst's performance but not by enough to significantly increase its overall ranking.

CCS Concepts

•**Information systems** → **Summarization;** *Social tagging systems;* •**Human-centered computing** → *Social networking sites;*

Keywords

ACM proceedings; LaTeX; text tagging

1. INTRODUCTION

A significant power of social media is the rapidity with which new information is posted and shared. If a user is interested in a particular item, event, or topic, she can often provide a few relevant keywords to a social network's search function and track new developments by reading recent postings. For instance, one can track tweets mentioning "goal" on Twitter during the 2014 World Cup to follow when goals are scored [5]. If a user wants to track some interesting event on current social media platforms, however, she must remain at her computer and manually filter through potentially many duplicate posts to track the event. At the 2015 National Institute of Standards and Technology (NIST) Text Retrieval Conference (TREC), several teams came together to address this problem by automating this tracking and summarization of interesting events, or "interest profiles," in TREC's Microblogs Track [13]. This paper describes RTTBurst[1], one of the systems participating in TREC's 2015 microblog track, and its use of linear regression to detect bursts in Twitter's social media stream in real time.

At a high level, our approach with RTTBurst is to generate vectors of frequency data from a sliding window over the past few minutes for each token in Twitter's 1% stream filtered according to the user's interest profile, fit this frequency data to an exponential curve, and tag tokens with steep exponential increases as "bursty." We make the assumption that bursts in the Twitter stream surrounding an user's interest profile denotes a significant or noteworthy moment relevant to that profile. We then extract Twitter messages, or tweets, containing the most bursty tokens to summarize this moment surrounding the burst. In this work, we detail RTTBurst's mechanics, how it determines messages relevant to a user's interest profile, and the steps needed to construct a system capable of processing Twitter's 1% stream in real time. We then describe our method for tuning RTTBurst given the data from both scenarios of TREC's 2015 Microblog track and the relative performance between RTTBurst and other systems participating in this track.

This work makes the following contributions:

- Presents a real-time streaming algorithm and feature set for the discovery and description of important moments relevant to a user's interests from Twitter's public sample stream, and
- Details RTTBurst's performance relative to similar systems.

2. RELATED WORK

Though RTTBurst focuses on the slightly different problem of discovering interesting moments in social media streams, our work shares foundations with classical event detection research. Identifying key events from the ever-growing body of digital media has fascinated researchers for over twenty years, starting from digital newsprint to blogs and now social media [1]. Early event detection research followed that of Fung et al. in 2005, who built on the burst detection scheme presented by Kleinberg by identifying bursty keywords from digital newspapers and clustering these keywords into groups to identify bursty events [8, 7]. This work succeeded in identifying trending events and showed such detection tasks are feasible. Recognizing that newsprint differs substantially from social media both in content and velocity, the research community began experimenting with new social media sources like blogs, but real gains came when microblogging platforms began their rise in popularity. These microblogging platforms include Twitter and Sina Weibo and are characterized by constrained post sizes (e.g., Twitter constrains user posts to 140 characters) and broadcasting publicly consumable information.

[1]Code for this system is available at https://github.com/cbuntain/UMD_HCIL_TREC2015

One of the most well-known works in detecting events from microblog streams is Sakaki, Okazaki, and Matsuo's 2010 paper on detecting earthquakes in Japan using Twitter [18]. Sakaki et al. show that not only can one detect earthquakes on Twitter but also that it can be done simply by tracking frequencies of earthquake-related tokens. Surprisingly, this approach can outperform geological earthquake detection tools since digital data propagates faster than tremor waves in the Earth's crust. Though this research is limited in that it requires pre-specified tokens and is highly domain- and location-specific (Japan has a high density of Twitter users, so earthquake detection may perform less well in areas with fewer Twitter users), it demonstrates a significant use case and the potential of such applications.

Along with Sakaki et al., 2010 saw two other relevant papers: Lin et al.'s construction of a probabilistic popular event tracker [11] and Petrović, Osborne, and Lavrenko's application of locality-sensitive hashing (LSH) for detecting first-story tweets from Twitter streams [15]. Lin's work demonstrated that the integration of non-textual social and structural features into event detection could produce real performance gains. Like many contemporary systems, however Lin's models require seeding with pre-specified tokens to guide its event detection and concentrates on retrospective per-day topics and events. In contrast, Petrović et al.'s clustering research in Twitter avoids the need for seed keywords and retrospective analysis by instead focusing on the practical considerations of clustering large streams of data quickly. While typical clustering algorithms require distance calculations for all pairwise messages, LSH facilitates rapid clustering at the scale necessary to support event detection in Twitter streams by restricting the number of tweets compared to only those within some threshold of similarity. Once these clusters are generated, Petrović was able to track their growth over time to determine impact for a given event. This research was unique in that it was one of the early methods that did not require seed tokens for detecting events and has been very influential, resulting in a number of additional publications to demonstrate its utility in breaking news and for high-impact crisis events [14, 16, 17]. Petrović's work and related semantic clustering approaches rely on textual similarity between tweets, which limits its ability to operate in mixed-language environments and differentiates LABurst and its language agnosticism.

Similar to Petrović, Weng and Lee's 2011 paper on EDCoW, short for Event Detection with Clustering of Wavelet-based Signals, is also able to identify events from Twitter without seed keywords [21]. After stringent filtering (removing stop words, common words, and non-English tokens), EDCoW uses wavelet analysis to isolate and identify bursts in token usage as a sliding window advances along the social media stream. Besides the heavy filtering of the input data, this approach exhibits notable similarities with the language-agnostic method we describe herein with its reliance on bursts to detect event-related tokens. These methods, however, operate retrospectively, focusing on daily news rather than breaking event detection on which our research focuses. Becker, Naaman, and Gravano's 2011 paper on identifying events in Twitter also fall under retrospective analysis, but their findings also demonstrate reasonable performance in identifying events in Twitter by leveraging classification tasks to separate tweets into those on "real-world events" versus non-event messages [2]. Similarly, Diao et al. also employ a retrospective technique to separate tweets into global, event-related topics and personal topics [6].

Many researchers have explored motivations for using platforms like Twitter and have shown interesting dynamics in our behavior around events with broad impact. For instance, Lehmann et al.'s 2012 work on collective attention on Twitter explores hashtags and the different classes of activity around their use [10]. Their work includes a class for activity surrounding unexpected, exogenous events, characterized by a peak in hashtag usage with little activity leading up to the event, which lends credence to our use of burst detection for identifying such events. Additionally, this interest in burst detection has led to several domain-specific research efforts that also target sporting events specifically[20, 23, 9]. Lanagan and Smeaton's work is of particular interest because it relies almost solely on detecting bursts in Twitter's per-second message volume, which we use as inspiration for one of our baseline methods discussed below. Though naive, this frequency approach is able to detect large bursts on Twitter in high-impact events without complex linguist analysis and performs well in streaming contexts as little information must be kept in memory. Detecting such bursts provide evidence of an event, but it is difficult to gain insight into that event without additional processing. LABurst addresses this need by identifying both the overall burst and keywords related to that burst.

More recently, Xie et al.'s 2013 paper on TopicSketch seeks to perform real-time event detection from Twitter streams "without pre-defined topical keywords" by maintaining acceleration features across three levels of granularity: individual token, bigram, and total stream [22]. As with Petrović's use of LSH, Xie et al. leverage "sketches" and dimensionality reduction to facilitate event detection and also relies on language-specific similarities. Furthermore, Xie et al. focus only on tweets from Singapore rather than the worldwide stream. In contrast, our approach is differentiated primarily in its language-agnosticism and its use of the unfiltered stream from Twitter's global network.

Despite this extensive body of research, it is worth asking how event detection on Twitter streams differs from Twitter's own offerings on "Trending Topics," which they make available to all their users. When a user visit's Twitter's website, she is immediately greeted with her personal feed as well as a listing of trending topics for her city, country, worldwide, or nearly any location she chooses. These topics offer insight into the current popular topics on Twitter, but the main differentiating factor is that these popular topics are not necessarily connected to specific events. Rather, popular memetic content like "#MyLovelifeInMoveTitles" often appear on the list of trending topics. Additionally, Twitter monetizes these trending topics as a form of advertising [19]. These trending topics also can be more high-level than the interesting moments we seek to identify: for instance, during the World Cup, particular matches or the tournament in general were identified as trending topics by Twitter, but individual events like goals or penalty cards in those matches were not. It should be clear then that Twitter's trending topics serves a different purpose than the streaming event detection described herein.

3. METHODS

This section describes the stages that comprise the RTTBurst pipeline, followed by an overview of the parameter optimization process we used to determine "good" values for RTTBurst's hyper parameters.

3.1 RTTBurst Pipeline

RTTBurst's high-level pipeline is composed of several stages:

1. Identify tokens relevant to a user's interests,
2. Filter Twitter's 1% sample stream for relevant tweets,
3. Identify tokens currently experiencing a burst in usage,

4. Summarize the bursty moment, and

5. Return relevant tweets to the user.

Each stage is described below.

3.1.1 Parsing Interest Profiles

TREC 2015's Microblog track was built around a real-time filtering task targeting the Twitter social network platform and its 1% public sample stream (an unfiltered stream containing a random subset of all tweets being posted to Twitter) [13]. As mentioned in the track's 2015 overview paper by Lin et al, this filtering task's goal is to identify new tweets relevant to a set of given interest profiles, each of which is comprised of a unique identifier, title (a few keywords describing the information need), a brief one-sentence description, and a paragraph-length narrative describing the topic of interest. One can then extract keywords from these interest profiles to discard tweets from Twitter's 1% sample stream that do are not relevant to the interest profiles.

For our work, we processed each profile's title and disregarded the profile's description and narrative. While one could leverage query expansion with the description and narrative fields, our experience with the sample topics suggested the keywords in the title field had the most signal and were the least ambiguous. For example, interest profile MB227 had the title, "Pradaxa side effects," with the description, "Find information on the negative side effects associated with the blood thinning drug Pradaxa," but keywords like "information" and "associated" are low-signal tokens.

To extract keywords from an interest profile's title, we leveraged two libraries: Benjamin Piwowarski's datasets[2] package to parse the the TREC topic files, and Python's Natural Language Toolkit (NLTK) [3]. With the NLTK package, we were able to remove stop words and lemmatize keywords and get a cleaner keyword set for filtering. As an example, RTTBurst would ingest the title, "arson fires in inner cities," and produce the following tokens for filtering: "arson," "fire," "inner," and "city."

3.1.2 Filtering the Twitter Sample Stream

After collecting keywords relevant to the given interest profiles, RTTBurst leveraged Apache Spark's[3] built-in Twitter receiver to collect all tweets from the public sample stream. Each tweet was tokenized using CMU's ARK TweetNLP tokenizer[4]. We then applied a series of quality metrics to remove non-English tweets and low-quality tweets with more than three hashtags, more than five web links (URLs), fewer than two tokens, and any tweet containing the string "follow" (motivated by the large amount of "follow-me" spam where Twitter users ask others to follow them).

3.1.3 Identifying Bursty Tokens

Our first step in identifying bursty tokens was to capture how a token's usage changes over time. The Spark Streaming library provides such a mechanism with its sliding window functionality. We maintained a sliding window over all tweets generated by the Twitter streaming API within the past 2 minutes and incremented the window by 60-second time slices. Therefore, each window had an overlap with the previous 60 seconds to smooth the input.

For each 2-minute window, we calculated the number of users tweeting with each token and stored this frequency over the previous N windows. We normalize these frequencies by the number of unique tokens in the past N windows and use add-one additive smoothing to correct for tokens with zero occurrences in a single window. Following the features set forth in the paper by Buntain et al. in 2016, we then used the Apache Commons linear regression library to fit a line to the natural log of this frequency data [4]. By transforming this frequency data to the logarithmic space, exponential curves will appear linear, simplifying the linear regression step, and the steeper the slope of the best-fit line, the steeper the exponential growth of the token's usage. Based on this fit, we then scored each token by the product of the slope of the best-fit line and its Pearson's R^2 coefficient. Since Pearson's R^2 coefficient is in the range $[-1, +1]$, this product reduced scores for highly deviant frequency curves. In this manner, tokens experiencing large bursts in usage, which we would expect to exhibit exponential growth, were scored highly. We then discarded all tokens with scores below a burst threshold γ and any token whose length is less than four characters.

3.1.4 Moment Summarization

Every sixty seconds, RTTBurst identified a new (possibly empty) set of bursty tokens, which correspond to moments of note in the relevant interest profile. For TREC's Microblog track, however, returning these bursty tokens was not adequate for summarizing the moment. Rather, the Microblog track required systems to return tweets to summarize these moments similar to the ReDites system proposed by Osborne and colleagues [14].

To this end, every sixty seconds, RTTBurst parsed all tweets in the previous N windows to create a subset of tweets containing these bursty tokens. We then calculated a Jaccard similarity score for each tweet in this subset by comparing the tweet to tweets returned to the user in previous windows. Any new tweet whose Jaccard similarity was above our threshold $J_t = 0.7$ was discarded, and the remaining tweets were sorted by their similarity scores in decreasing order. Finally, the top M least similar tweets containing bursty tokens from the past N windows were assigned to the relevant interest profiles and stored.

3.1.5 Relevant Tweet Selection

RTTBurst's final step before pushing a tweet to a user included one last pass through the tweets to select those that were most relevant to the given interest profile. For each candidate tweet stored up to this point, RTTBurst then selected only those tweets that contained at least X tokens from the relevant interest profile. All other tweets were then discarded.

In summary, for Scenario A of TREC's Microblog track, the top 10 most dissimilar tweets containing bursty tokens and at least two tokens from the relevant interest profile were returned to the user per day. Scenario B followed the same pipeline with the additional relaxation of returning the top 100 most dissimilar tweets.

3.2 Parameter Optimization

In the previous few sections, we described the RTTBurst system and mentioned a set of important parameters that control the system. The most important parameters were the threshold for bursty tokens γ, the number of windows N over which to track frequency, and the number of tweets M to save per minute. To determine appropriate settings for these main parameters, we ran RTTBurst through a randomized parameter optimization following the official run for the 2015 TREC Microblog tasks.

We bound the number of windows $N \in [5, 45]$, the number of tweets per minute $M \in [10, 50]$, and the burst score threshold γ within the top 10% of the burst scores generated for that setting of N. For the number of tokens X required for relevance selection, we used both $X = 1, 2$ during the TREC experiment and fixed

[2]Available at https://github.com/bpiwowar/datasets

[3]https://spark.apache.org

[4]http://www.cs.cmu.edu/~ark/TweetNLP/

it at $X = 2$ for our post hoc analysis. Other parameters like the maximum number of hashtags or the maximum Jaccard similarity were considered fixed.

Owing to time constraints and the amount of time necessary to run RTTBurst against the full 10-day period covered by TREC's Microblog track, we only ran this parameter optimization with ten random values of N and ten values for M and γ. For each of these parameter sets, we ran RTTBurst on the entire 10-day time period, against all 51 topics in the TREC evaluation set, and scored each run using the scenario A and scenario B evaluation scripts provided by the TREC organizers. Since these runs were against Twitter's entire 1% sample stream and the TREC organizers could not possibly score each tweet in the stream for relevance to the interest profiles, some tweets returned by RTTBurst do not have associated relevance scores for these tasks, so we omit those tweets and report the number of such tweets in the results section below.

Table 1: Official TREC 2015 Scores, Scenario A

Run	ELG	nDG
Type A	0.2471	0.2471
Type B	0.2020	0.2020

Table 2: Official TREC 2015 Scores, Scenario B

Run	nDCG@10
Type A	0.2471
Type B	0.2020

4. RESULTS

We divide our results into two sets: The first covers official results from the real-time Microblog task as scored by NIST, and the second covers results from our post hoc runs using the relevance and scoring data provided by NIST after the official run.

4.1 Official NIST Results

The version of RTTBurst deployed in July of 2015 was simpler than the version described in the previous section. Primarily, the original RTTBurst implementation did not employ several of the tweet quality metrics (i.e., it did not filter out tweets with many hashtags, many links, or few tokens). The original version did discard non-English tweets and tweets including the string "follow" however. In addition, the first version of RTTBurst did not include mechanisms for preventing duplicate tweet content from being reported to the user. Prior to using these quality metrics, it became clear that a significant amount of spam was being caught by our system. For example, while the original RTTBurst implementation did prevent the same tweet ID from being reported twice, two different tweets with the same content might still be reported, and we saw many Twitter bots spamming the same tweet content with only slight differences (one token at the end of the tweet might differ from one spam tweet to the next).

For both scenarios in the 2015 Microblog track, RTTBurst used the following parameter settings: $\gamma = 0.07$, $N = 30$, and $M = 10$. Furthermore, we submitted two runs for both scenarios: type "a," in which we required each tweet reported to the user to include two tokens from the relevant interest profile, and type "b," where we required only one token to be present in both the tweet and interest profile.

For the track's mobile notification scenario (Scenario A), our scores after deploying RTTBurst and submitting our results to NIST are shown in Table 1. The scoring metrics used for Scenario A were Expected Latency Gain (ELG) and normalized Cumulative Gain (nCG) (see the Microblog Track overview paper for more details on these metrics [13]). For the email digest scenario (Scenario B), our scores are shown in Table 2. The only scoring metric in Scenario B was normalized Discounted Cumulative Gain (nDCG) for the first ten (10) tweets per day (nDCG@10) (again, see the Microblog Track overview paper for more details [13]).

4.2 Post Hoc Results

Following the official run in July and the following TREC conference, we implemented the additional tweet similarity and quality metrics described above. Given the poor performance of the type "b" version (requiring only a single token overlap between returned tweets and interest profiles), our post hoc analysis used only our type "a" version, thereby requiring two tokens in common between reported tweets and interest profiles. Parameter optimization covered γ, N, M, and two sets of tweet quality metrics. A list of the highest-scoring parameters is shown in Table 3, and score graphs for each parameter are shown in Figure 1. From the graphs, it seems the window count parameter N has the strongest positive correlation, and scores with the number of tweets returned per minute M having the least correlation.

Following these post hoc runs, our highest scores would not significantly increase RTTBurst's rankings in the official results; for Scenario A, RTTBurst would move up two spots, from 14th to 12th, and Scenario B would see no change.

5. DISCUSSION

In reviewing the results above and those from the Microblog overview paper by Lin et al., it appears higher scores are associated with fewer reported tweets. This result is confirmed by the score for a system that returns no tweets at all: an ELG, nCG, and nDCG@10 of 0.2471. Therefore, a system that returns nothing would rank 14th out of 37 in Scenario A and 4th out of 42 for Scenario B. While we were able to increase our performance beyond this silent baseline in our post hoc analysis, an interesting correlation presented itself between the number of tweets returned by RTTBurst and the scores achieved. As shown in Figure 2, a strongly negative, nearly linear correlation ($R^2 = 0.8172$) exists between the more tweets RTTBurst returns to the user, the lower the score produced by the TREC evaluations.

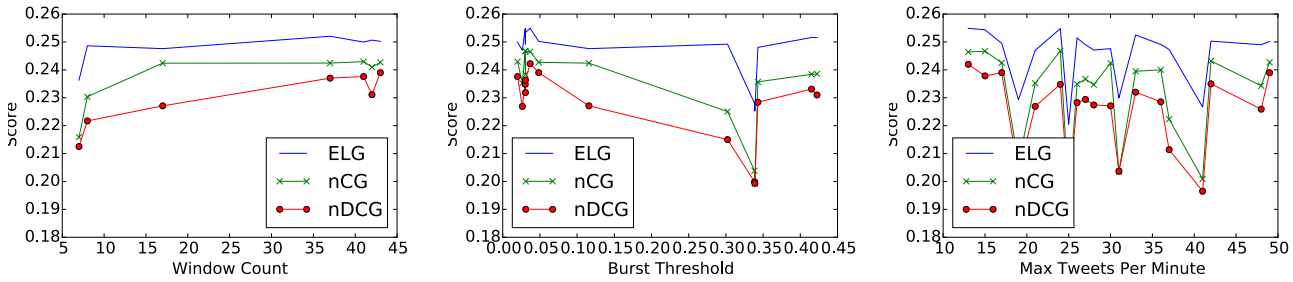
6. LIMITATIONS

A persistent issue with our post hoc analysis is the presence of unevaluated tweets, which makes a true performance comparison between our previous runs and post hoc runs difficult. That is, while the NIST judges provided relevance assessments for approximately 94k tweets, the Twitter sample stream over the TREC evaluation period contains around 40 million tweets, so it is highly likely post hoc runs of RTTBurst may return tweets without relevance assessments. Going forward, we need to explore better methods for scoring these unjudged tweets or comparing judged and unjudged tweets and scores.

Another limitation present in RTTBurst is the absence of query expansion techniques. RTTBurst was originally designed as an open-domain system without tracking capabilities, and the modifications to the system to capture tweets relevant only to specific in-

Table 3: Best-Scoring Parameters After Optimization

Window Count N	Tweets Per Minute M	Threshold γ	SA Scored/Unscored	ELG	nCG	SB Scored/Unscored	nDCG
37	13	0.036854	6/8	**0.2549**	0.2464	6/19	0.2420
18	34	0.138824	3/5	0.2525	**0.2494**	3/11	0.2479
37	48	0.067306	4/1	0.2506	0.2479	4/2	**0.2489**



(a) Scores Over Window Counts (b) Scores Over Burst Thresholds (c) Scores Over Tweets Per Minute

Figure 1: Scores Over Individual Parameters

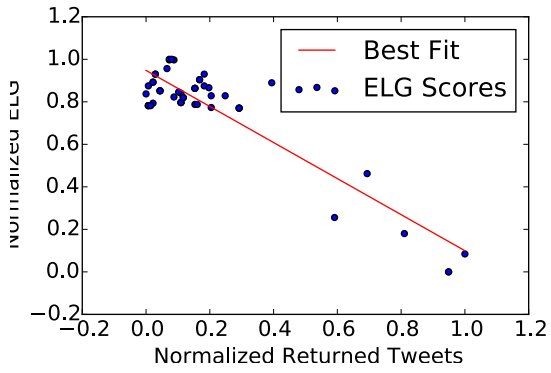


Figure 2: Tweets Versus Scores

terest profiles was not very nuanced (i.e., discarding tweets that do not contain lemmatized tokens from the interest profile). Modern information retrieval systems employ more principled approaches to expanding queries, and an interest profile is essentially a query. Since RTTBurst does not include this kind of expansion, we are potentially discarding many otherwise relevant tweets, something that future versions of this system should address.

Finally, RTTBurst's driving principal is that bursts in the usage of specific tokens indicate an interesting event or moment surrounding that token. Examples supporting this assumption include bursts in usage of the token "goal" during the World Cup when a player scores or "kidschoice" during the Kids' Choice Awards. In theory, the moments that see such bursts are the moments about which someone following the event would want to be notified. In an ideal case, this theory might be adequate, but after participating the 2015 TREC Microblog Track, the influence of spam became a major issue that our RTTBurst does not handle. Following interest profiles related to Taylor Swift and Ariana Grande demonstrated that spammers would try to drive visitors to websites by mentioning either or both of these major celebrities, and many Twitter bots would post large volumes of these spam tweets with small minor differences between the messages (presumably to avoid Twitter's own spam filter). We initially did not expect spammers to be able to influence the Twittersphere to such a degree, so future versions of RTTBurst will require additional filtering mechanisms to sanitize this data.

7. CONCLUSIONS

This work described RTTBurst, an end-to-end system for ingesting a user's interest profiles that describe some topic of interest and identifying new tweets that might be of interest to that user using a simple model for bursts in token usage. We laid out RTTBurst's architecture, our participation in and performance at the TREC 2015 Microblog Track, and a post hoc analysis for increasing RTTBurst's performance. While not as relatively performant in the Microblog Track's real-time notification task, RTTBurst did perform well (ranking 4th overall and second in the automatic category of Scenario B) in providing daily summaries for various interest profiles. Following the official TREC evaluation period, we were also able to increase RTTBurst's performance but not by enough to significantly increase its overall ranking. Further steps could be taken, however, to integrate modern information retrieval techniques like query expansion and spam detection to increase RTTBurst's performance. Given RTTBurst's mathematically simple model and its amenities for processing streams, its approach may be useful in integrating with other systems as well.

8. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under CNS-1405688 [12]. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the sponsors. This work also made use of the Open Science Data Cloud (OSDC), which is an Open Cloud Consortium (OCC) - sponsored project. The OSDC is supported in part by grants from Gordon and Betty Moore Foundation and the National Science Foundation and major contributions from OCC members like the University of Chicago.

9. REFERENCES

[1] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–45. ACM, 1998.

[2] H. Becker, M. Naaman, and L. Gravano. Beyond Trending Topics: Real-World Event Identification on Twitter. *ICWSM*, 11:438–441, 2011.

[3] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. " O'Reilly Media, Inc.", 2009.

[4] C. Buntain, J. Lin, and J. Golbeck. Discovering Key Moments in Social Media Streams. In *Consumer Communications and Networking Conference (CCNC), 2016 13th Annual IEEE*, jan 2016.

[5] L. Cipriani. Goal! Detecting the most important World Cup moments. Technical report, Twitter, 2014.

[6] Q. Diao, J. Jiang, F. Zhu, and E.-P. Lim. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544. Association for Computational Linguistics, 2012.

[7] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 181–192. VLDB Endowment, 2005.

[8] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 91–101, New York, NY, USA, 2002. ACM.

[9] J. Lanagan and A. F. Smeaton. Using twitter to detect and tag important events in live sports. *Artificial Intelligence*, pages 542–545, 2011.

[10] J. Lehmann, B. Gonçalves, J. J. Ramasco, and C. Cattuto. Dynamical Classes of Collective Attention in Twitter. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 251–260, New York, NY, USA, 2012. ACM.

[11] C. X. Lin, B. Zhao, Q. Mei, and J. Han. PET: a statistical model for popular events tracking in social communities. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 929–938, New York, NY, USA, 2010. ACM.

[12] J. Lin. Hadoop NextGen Infrastructure for Heterogeneous Approaches to Data-Intensive Computing. Award Abstract CNS-1405688, National Science Foundation, 2014.

[13] J. Lin, M. Efron, Y. Wang, G. Sherman, and E. Voorhees. Overview of the TREC-2015 Microblog Track. In *Proceedings of the Twenty-Fourth Text REtrieval Conference (TREC 2015)*, Gaithersburg, MD, 2015.

[14] M. Osborne, S. Moran, R. McCreadie, A. Von Lunen, M. Sykora, E. Cano, N. Ireson, C. Macdonald, I. Ounis, Y. He, and Others. Real-Time Detection, Tracking, and Monitoring of Automatically Discovered Events in Social Media. *Association for Computational Linguistics*, 2014.

[15] S. Petrović, M. Osborne, and V. Lavrenko. Streaming First Story Detection with Application to Twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 181–189, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[16] S. Petrovic, M. Osborne, R. McCreadie, C. Macdonald, I. Ounis, and L. Shrimpton. Can Twitter replace Newswire for breaking news? In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*, volume 2011, 2013.

[17] J. Rogstadius, M. Vukovic, C. A. Teixeira, V. Kostakos, E. Karapanos, and J. A. Laredo. CrisisTracker: Crowdsourced social media curation for disaster awareness. *IBM Journal of Research and Development*, 57(5):4:1–4:13, sep 2013.

[18] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 851–860, New York, NY, USA, 2010. ACM.

[19] L. Sydell. How Twitter's Trending Algorithm Picks Its Topics, dec 2011.

[20] V. Vasudevan, J. Wickramasuriya, S. Zhao, and L. Zhong. Is Twitter a good enough social sensor for sports TV? In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 181–186. IEEE, 2013.

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

[22] W. Xie, F. Zhu, J. Jiang, E.-p. Lim, and K. Wang. TopicSketch: Real-time Bursty Topic Detection from Twitter. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 837–846. IEEE, 2013.

[23] S. Zhao, L. Zhong, J. Wickramasuriya, and V. Vasudevan. Human as Real-Time Sensors of Social and Physical Events: A Case Study of Twitter and Sports Games. *CoRR*, abs/1106.4, 2011.