Estimating Topical Volume in Social Media Streams

Praveen Bommannavar[1], Jimmy Lin[2], and Anand Rajaraman[3]

[1] Twitter [2] University of Maryland [3] Stanford University

praveen@twitter.com, jimmylin@umd.edu, anand@anandr.com

ABSTRACT

This paper tackles the problem of estimating the volume of social media posts (e.g., tweets) that pertain to a topic. This task differs from related filtering and event detection applications in that filtered content isn't meant for human consumption, but rather we are primarily interested in estimating the cardinality of relevant posts. We present a simple yet effective technique for generating and curating keywords to create what we call "overlap filters", which can be applied to a stream of social media posts. Our approach leverages human labeling and thus a crucial element of the work involves minimizing the cost of human computation. On top of a "day zero" cold start algorithm, we describe a number of optimizations that take advantage of history to further reduce labeling costs. Experimental results show that our overlap filters produce accurate volume estimates at low costs, and our method is simple enough to deploy in practice.

1. INTRODUCTION

The problem of estimating the volume of social media posts (e.g., tweets) pertaining to a topic is fundamental to several applications: tracking the popularity of politicians and brands, the box-office appeal of movies, the audience for a sporting event, the potential sales of a product, and so on. The number of tweets pertaining to unfolding events is an often-quoted statistic in media coverage. These applications demand high recall with a "reasonable" level of precision, which stands in contrast to related formulations of filtering tasks, which require high precision and are often not particularly concerned with recall. In our formulation, filtered content is not meant for direct user consumption: we are primarily interested in the cardinality of the relevant tweets (or equivalently, prevalence), although the content can serve as training data for a machine-learned topic classifier.

Despite the development of sophisticated techniques in the academic literature on related tasks, the state-of-the-art method for finding social media posts about a topic in production settings is to use a manually curated set of keywords to filter the content stream. This method, however, suffers from two drawbacks:

- There is no true measure of the recall of the filter.
- Topics evolve over time, often in unpredictable ways, leading to changes in language, thus degrading the effectiveness of static keyword filters over time.

The first problem is common in many filtering applications because we don't know what the filter is missing. It is particularly hard to solve using sampling-based approaches because almost all real topics of interest are sparse (that is, they occur in a very small fraction of all posts), making unbiased sampling too expensive to be a practical solution. The second problem is known as topic drift, which with curated keywords requires humans to constantly monitor the current solution and add/remove keywords as appropriate.

We describe a family of topic filters called "overlap filters" that solve both of the above problems in a simple yet effective manner. An overlap filter is a set of keywords from which a disjunctive (OR) query can be created to filter a stream of social media posts to retain only those about a particular topic. We present an iterative approach for generating and curating these keywords sets, which allow us to estimate the volume (or prevalence) of posts about a topic in a given time interval. Our technique allows trading off comprehensiveness and cost, and additionally the filters evolve with the topic and thus can handle topic drift. Our approach leverages human labeling and thus a crucial element of the work involves minimizing the cost of human computation. On top of a "day zero" cold start algorithm, we describe a number of optimizations that take advantage of history to further reduce labeling costs. Experimental results show that our overlap filters produce accurate volume estimates at low costs, and our method is simple enough to deploy in practice.

2. RELATED WORK

Topic detection and tracking (TDT) [10, 1] represents one of the earliest formulations of a filtering task over a stream of documents. In particular, the goal of the "tracking" task is to find all news stories about a particular event. There are two main differences between

TDT and our problem: First, TDT is focused primarily on news stories, which are much longer than social media posts. Systems that operate over news stories do not need to handle challenges associated with informal language prevalent in social media. Second, the output of a TDT system is meant for consumption by an analyst, which means that although effectiveness is evaluated in terms of detection error tradeoff (DET) curves, systems usually value precision over recall. This stands in contrast to our problem, where the primary goal is to estimate the volume of posts on a particular topic, and hence we need to achieve high recall. More recently, there has been work on tracking topics on Twitter, e.g., [4, 6, 8], although these techniques are also not focused on high recall.

Our work is related, but distinct from, a large body of work on Twitter event detection (e.g., [9, 7, 5, 2, 11, 3]). The precise formulation of the problem varies, but the systems have the same basic goals as Twitter's "trends" product: to identify interesting, surprising, or unexpected events that are occurring on Twitter. These systems are primarily concerned with precision and latency—the goal is to identify events as accurately and as close to the actual event occurrence as possible. Once again, the setup is different from the problem we tackle in this paper.

3. OVERLAP FILTERS

In our problem formulation, we are given a topic of interest and the goal is to estimate the volume (or prevalence) of posts in a social media stream that pertain to the topic. We describe an iterative procedure for generating, selecting, and curating a set of keywords, which are used in a disjunctive (OR) query and applied to each post, retaining only those that match as being relevant. By obtaining several sets of filter keywords at varying precisions, we can estimate the overall volume of posts about the topic based on the cardinalities of the filtered sets.

Note that in this work, we assume the volume estimation is performed daily over the course of a longer period of time, but of course the granularity can be adjusted as needed. We begin with what we refer to as the "day zero" algorithm, which assumes a cold start and does not leverage results from previous days. Later, we describe optimizations that exploit history.

3.1 Alpha-Neighborhoods

Our starting point for candidate keyword generation is a simple construction called an *α-neighborhood*. Given a set of seed keywords, its α-neighborhood gives an expanded set of candidate keywords with respect to the topic of interest.

Let X denote a set of keywords and all tweets containing any keyword from X as $\mathbf{X}$ (i.e., we treat X as a disjunctive query applied to the tweet stream). Similarly, let y be a candidate keyword to be added to a given keyword set, and let the set of tweets containing keyword y be $\mathbf{y}$. The α-neighborhood of X is defined as follows:

$$\{y : [X|y] \geq \alpha\}$$

where $[X|y]$ is the *overlap* of candidate y with X:

$$[X|y] = \frac{|X \cap y|}{|y|}.$$

Intuitively, the α-neighborhood of X is the set of all keywords co-occurring frequently enough with X, where this is defined by the overlap score exceeding the threshold α. A straightforward algorithm is to iterate through all keywords and compute the overlap for each one, determining whether or not to include it in the expansion.

As a running example, let us consider the topic Apple (the technology company). A seed set of terms with which to begin this process might be:

$$X = [\text{apple, iphone, ipod, mac}]$$

These might be keywords that the topic developer thinks of prior to actually looking at data. The α-neighborhood of this seed keyword set X for $\alpha = 0.1$ might be:

$$X = [\text{pear, orange, macbook, ipad, itunes, jobs}]$$

Note that the terms 'pear' and 'orange' appear because they co-occur with 'apple', but do not pertain to our topic of interest. For this reason, a second phase to remove such spurious keywords is required.

3.2 (Alpha, Beta)-Neighborhoods

After obtaining an α-neighborhood, we prune the returned keywords to retain only those with high enough precision (a parameter we denote by β). In our notation, the (α, β)-*neighborhood* of X is:

$$\{y : [X|y] \geq \alpha\} \cap \{y : \text{Precision}(y) \geq \beta\}.$$

where Precision(y) refers to precision with respect to filtering based solely on keyword y. To determine this, human labeling must be employed. In later sections we focus on minimizing the cost of this labeling effort. However, for the present description we assume that topicality of a given tweet can be ascertained via a query to an external service such as Amazon Mechanical Turk.

A procedure for accepting or rejecting a keyword y into the (α, β)-neighborhood is as follows: First, we shuffle the day's tweets to remove all time dependency. Next, we iterate through the shuffled tweets. If a tweet matches any keyword in the α-neighborhood of X, submit this tweet for evaluation. This is the setup for running what is known as a Sequential Probability Ratio Test (SPRT). We have hypotheses $H_0 : p = p_0$ and $H_1 : p = p_1$. In our case, $p_0 = \beta - 0.1$ and $p_1 = \beta$.

These are sufficiently separated so that we do not have to distinguish between very close values, and as a result overspend on labeling.

In the SPRT procedure, after each tweet is labeled we update the following score:

$$S_i = S_{i-1} + \begin{cases} \log \frac{p_1}{p_0}, & \text{if tweet is on topic.} \\ \log \frac{1-p_1}{1-p_0}, & \text{otherwise} \end{cases}$$

with $S_0 = 0$. If this score crosses above b then we remove y from the candidates list and accept it into X. If it crosses below threshold a, we remove y and reject it altogether. If the score is in between, we conclude that there is not enough evidence to make a judgment, and continue submitting tweets matching the keyword for evaluation. Parameters a and b are selected to achieve a particular false positive rate (FPR) and false negative rate (FNR). In our applications, we opt for FPR and FNR of $\frac{1}{11}$, which results in parameters $a \approx \log \frac{FNR}{1-FPR}$ and $b \approx \log \frac{FPR}{1-FNR}$ of $\log(0.1)$ and $\log(10)$ respectively. We note that substantial savings on labeling is achieved using this scheme, as opposed to the alternative common practice of labeling tweets in batches, because we only expend as much effort as is required to obtain a particular confidence interval.

Returning to our running example for the topic Apple, the (α, β)-neighborhood of the seed set above with $\beta = 0.5$ might look like the following set:

$$X = [\text{apple, iphone, ipod, mac,} \\ \text{macbook, ipad, itunes}]$$

where only terms of sufficiently high precision survive.

3.3 (Alpha, Beta)-Closures

Computing an (α, β)-neighborhood expands the set of filter keywords, and we can continue this expansion by iteratively running the procedure until convergence. We say the closure has been computed once the iterations:

$$X \leftarrow X + (\alpha, \beta)\text{-neighborhood}(X)$$

no longer add keywords to X.

After computing $\overline{X} = \text{closure}(X)$, we can determine the precision and the correctly matched volume of $\overline{X}$. Although the value of β corresponds to individual keywords having precision β or higher, this is not a guarantee that the disjunction of these terms applied to the tweet stream will have a similar precision bound. While counterexamples exist, in practice we find that the precision of the set of keywords is often higher than β (because β is just a threshold).

To determine the precision of $\overline{X}$, we once again iterate through the shuffled tweets, but rather than maintaining statistics for each candidate keyword, we can keep track of a precision p for the entire closure, and terminate when a minimum number of tweets have been seen (say $n \geq 200$), and the 95% confidence interval for p is narrow enough compared to p itself:

$$\left(\frac{1.96}{0.04}\right)^2 \left(\frac{1-p}{p}\right) < n. \quad (1)$$

which is easily derived from the general expression for the length of a 95% confidence interval: $1.96\frac{\sigma}{\sqrt{n}}$ where we choose $0.04p$ as the margin of error. The margin of error depends on p because we care about the relative error of the overall volume, which is what this precision estimate is to be used for.

Specifically, knowing this precision allows us to obtain the volume of tweets found by the closure by simply multiplying the measured precision with the number of tweets matched by the filter.

3.4 Parameter Estimation

How do we choose the value of α and β? The choices for β will be covered in the next section. For α, while there is evidence that many topics follow similar patterns and a principled approach to selecting α can be derived, we found a much simpler approach that is effective in practice: we shrink α by a factor of 0.75 and repeatedly compute the closure until the increase in coverage is below some tolerance. For a fixed β and initial keyword set, there is some value of α below which the filter no longer includes additional candidates in order for the number of tweets matching the filter set to stabilize. This repeated computation does not significantly increase the cost of labeling, since previously labeled results can be reused, using the same randomization of the corpus.

The closure of the seed set for the topic Apple in our running example might finally look like the following:

$$\overline{X} = [\text{apple, ipad, ipads, 4s, mac, itunes} \\ \text{macbook, imessage, icloud, ipads,} \\ \text{ipod, itouch, imac, 32gb, teamiphone}]$$

For example, after we compute the $(0.1, 0.5)$-closure of X, we can reduce α by a factor of 0.75 to see if number of tweets returned by this filter increases substantially. If not, we terminate the computation and estimate precision of the closure (see next section); otherwise we continue with the $(0.075, 0.5)$ closure, reusing as many labeled examples as possible, since we can maintain the shuffled order of tweets streamed through our algorithm. In this way, for a given β, we tune α downwards until the closure no longer grows.

Say we find that the precision of the closure in our example is 0.78 (despite having computed the closure with $\beta = 0.5$). We can simply count all the tweets matching the filter on the entire corpus and multiply by the measured precision to obtain our first estimate of the prevalence of the topic. Let us suppose the value

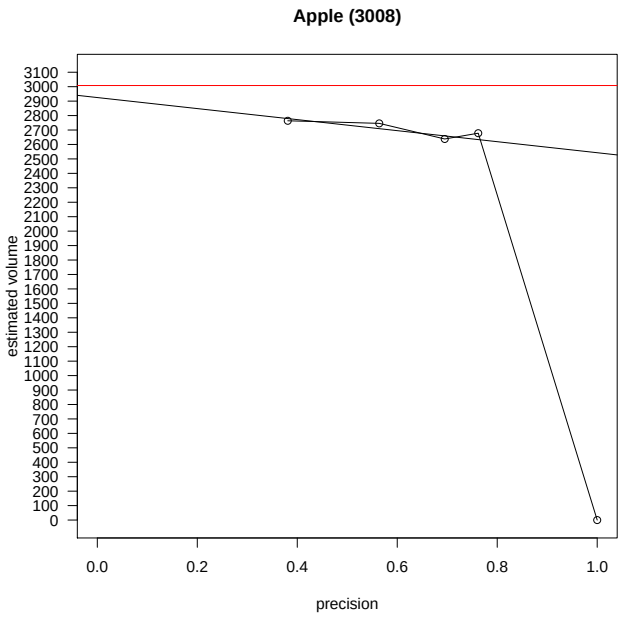

Figure 1: Estimated volume vs. precision curve. The true volume of the topic (indicated by the horizontal red line) can be inferred with high accuracy based on our approach.

we get is 2750 tweets; this serves as input to the volume estimation step, discussed next.

3.5 Volume Estimation

In the running example from the previous section, we found that with a given seed set of terms, we were able to produce an expanded keyword set with estimated precision of 0.78 and estimated volume of 2750 tweets (with parameters $\alpha = 0.1$ and $\beta = 0.5$). This setting can be represented on a plot of precision vs. estimated volume. By computing the (α, β)-closure for several decreasing values of β, we can plot a curve, from which the volume of the topic can be inferred.

Figure 1 illustrates the typical shape of such a curve. The horizontal red line represents the actual volume of the topic based on exhaustive labeling (more details in Section 5). One can observe that the points are populated from right to left, and the further left we go, the more expensive the points are to obtain—because SPRT runs for longer and more judgments are required. We empirically observe in our datasets that the intersection point with the y-axis is an estimate of the true volume of the topic in the dataset.

In general, this curve monotonically decreases with increasing precision (modulo precision measurement errors). Another important property we observe is that for smaller values β, the curve tends towards a linear trajectory. From this fact, we can infer the y-intercept using a few data points computed by the appropriate (α, β)-closures. Specifically, after the change in slope of the curve has stabilized in the lower-precision region, we can apply linear regression to obtain an estimate for volume of the topic. Additionally, the construction of these closures produces keyword filters of known precisions, which can be used as needed for other applications (such as a input to machine learning applications).

4. OPTIMIZATIONS

We refer to the algorithm described in the previous section as the "day zero" algorithm because it works in a "cold start" scenario, without any prior knowledge or dependencies (i.e., the topic developer can simply start with a set of keywords without looking at data). While we can in principle run this algorithm daily, it is not particularly cost effective since it does not take advantage of history—the fact that many keywords will carry over from day to day (although some won't, precisely due to topic drift). We can take advantage of history to reduce labeling costs in what we refer to as the "day one" algorithm; further optimizations are presented as part of the "optimized day one" algorithm.

4.1 Day One Algorithm

Recall that the day zero algorithm begins with a seed set of terms known to be topically relevant (as input to the iterative curation process). The following day, it is possible to reuse much of the information captured by the day zero algorithm. To this end, we propose two changes which allow for high fidelity estimates, but significantly reduces human labeling costs—we refer to this as the day one algorithm. To cope with topic drift, however, we can re-run the day zero algorithm periodically (e.g., weekly) to eliminate any idiosyncratic historical dependencies in the results.

Seed keywords with the closure from the previous day. After computing several closures on the previous day's seed set, one can simply seed the current day's (α, β)-closure with the expanded set corresponding to the previous day's closure with the smallest $\hat{\beta}$ not exceeding β. By doing so, we can run the current day's closures with larger values of α and thus consider fewer candidates in the SPRT process.

Note that due to topic drift, it is possible that some keywords are no longer relevant to the topic. To alleviate this issue we can run SPRT on tweets matching the expanded set and determine which of the previous day's terms to keep. The SPRT is run on a shuffling of the current day's tweets; evaluated tweets can be re-used for later closure computations.

Reuse tweet judgments for stable keywords. Another optimization we can employ is to recognize that many terms do not change meaning (and hence topicality) from day to day. For such terms, we can sample already

judged tweets from the previous day's closure computations instead of paying for new judgments. One metric to guide whether or not a term has sufficiently changed is the volume of tweets matching that term. If the volume changes significantly from day to day, it is an indication that we cannot rely on the previous day's data, whereas if this number is stable, it is likely "safe" to reuse the previous day's data.

4.2 Optimized Day One Algorithm

Additional optimizations can further reduce cost, in what we call the "optimized day one" algorithm:

Keyword grouping for SPRT. The main source of cost in our algorithm comes from running SPRT for each candidate. Since the cost of judgment is the same for each keyword, it is somewhat wasteful to run SPRT on low frequency terms which do not contribute much to the topic. Therefore, we can group several infrequent terms together so that the combined volume of tweets matching any term in the group is at some minimum level. In our experiments, the minimum volume in a group was 100. Then we can run SPRT on the group and either reject or accept them together.

Retweets and #hashtags. We additionally find that exploiting document structure specific to tweets can improve our algorithm. First, we remove retweets from the SPRT. This is because co-occurrences between unrelated terms can appear artificially high due to heavily retweeted text. Next, we can augment the seed set with the hashtag version of terms already in the seed set. This simple high precision heuristic is almost always helpful since the hashtags would be discovered anyway. Without including these hashtags ourselves, however, they would need to be discovered via our set expansion and pruning mechanism which incurs labeling cost.

5. EXPERIMENTS

5.1 Data

Our data set is comprised of approximately 800,000 English language Tweets obtained from the public sample stream on August 6, 2012, where we have normalized the tweets so that all words are in lowercase and non-ASCII characters have been removed. For each tweet, we have utilized Amazon Mechanical Turk to obtain a judgment on whether it pertains to any of four topics: Apple (the technology company), Mars (the planet), Obama, and Olympics. This *exhaustive labeling*, while expensive, allows us to perform unbiased evaluation of our techniques. However, due to the cost we were unable to explore more topics.

The chosen topics are interesting for a number of reasons. On August 6, 2012, two of these topics (Olympics and Mars) were prominent news topics, and hence frequently appeared in Twitter posts. Additionally, Mars and Apple are ambiguous without further specification; Apple can also refer to a fruit, and Mars has other interpretations as well. Amazon Mechanical Turk workers were told to specifically focus on the interpretations that we wish to study.

	Apple	Mars	Obama	Olympics
Actual	3034	2376	1253	23138
Day Zero Algorithm				
Estimated	2925 (3.6%)	2346 (1.3%)	1243 (0.8%)	23746 (2.6%)
Closure	2764 (0.91)	2345 (0.99)	1199 (0.96)	22979 (0.99)
Cost	$204	$169	$156	$2252
Day One Algorithm				
Estimated	2956 (2.6%)	2503 (5.3%)	1214 (3.1%)	-
Closure	2832 (0.93)	2322 (0.98)	1164 (0.93)	-
Cost	$210	$89	$62	-
Optimized Day One Algorithm				
Estimated	2942 (3.0%)	2554 (7.5%)	1167 (6.8%)	-
Closure	2884 (0.95)	2312 (0.97)	1084 (0.87)	-
Cost	$20	$34	$26	-

Table 1: Results (percent error of estimate and largest closure recall in parentheses).

5.2 Results

In these experiments, we assume that the cost of obtaining a human judgment is $0.05 per label. We can consider uniform sampling to be a baseline for determining the volume of a topic. In order to get even 100 tweets about a rare topic (1% of all tweets or fewer), one needs to uniformly sample and label 10,000 tweets from the corpus, which comes to $500. Ideally, more than 100 topical tweets are needed for accurate volume estimates, and in many cases, rare topics comprise far less than 1% of all tweets.

Results from our algorithms are shown in Table 1. The second row shows the true volume of each topic based on exhaustive labeling. Below that, for each of the day zero, day one, and optimized day one algorithms, we give the estimated volume, estimate volume from the leftmost closure point (on the estimated volume vs. precision plot), and labeling cost. For the estimated volume, we show the percent error in parentheses and for the closure, we show the recall in parentheses. In all cases, we see that we are able to accurately estimate the true volume of these topics, and can produce high-recall keyword filters that can be used in other applications. Figure 2 shows the corresponding estimated volume vs. precision curves for the day zero algorithm on three topics (the fourth is shown in Figure 1).

To evaluate the day one algorithm, we ran the day zero algorithm on tweets from the previous day (Au-

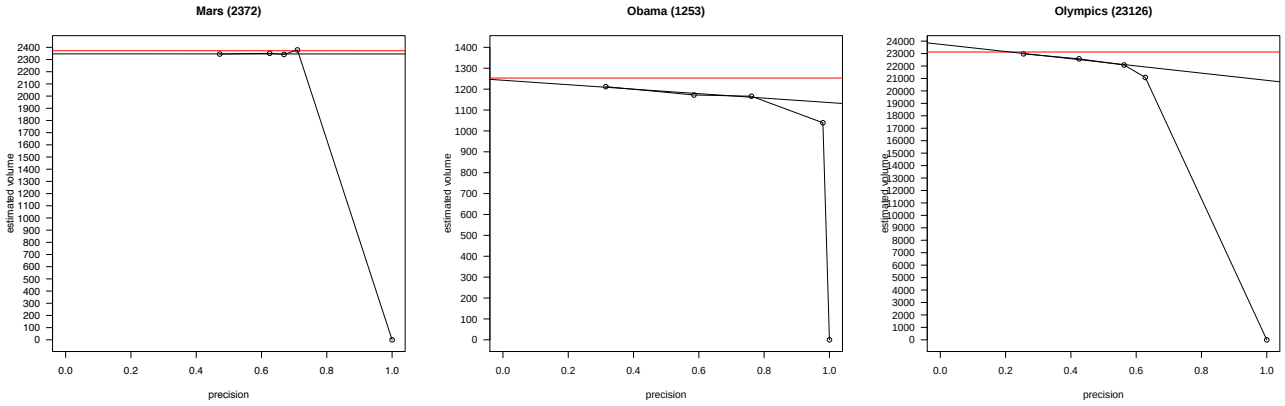

Figure 2: Estimated volume vs. precision curves for the day zero algorithm, with the true volume of the topic indicated by the horizontal red line.

gust 5, 2012) and used the keywords and labeled tweets as input. We did not perform this experiment on the Olympics topic as our budget did not permit further human labeling. Results from the day one algorithm are shown in the middle of Table 1. As we can see, cost was reduced with the day one algorithm with one exception. In the case of Apple, the cost remained roughly the same, we believe due to the generation of too many new candidates.

Utilizing the cost saving techniques of the optimized day one algorithm yields further savings from the day one algorithm (even with the Apple topic). These results are shown in the bottom of Table 1. Note, however, that the lower costs come with increased error on the volume estimation.

6. CONCLUSION

In this work, we tackle a slightly different take on filtering document streams. In our application, we are primarily concerned with the cardinality estimation of topically relevant posts, which puts us in a region of the precision/recall tradeoff space that is under-explored. From a practical standpoint, however, estimating topical volume in social media streams is an important application. We present a solution that is simple, accurate, and cost-effective—important criteria for practical algorithms meant for production deployment.

7. ACKNOWLEDGMENTS

Many thanks to Alek Kolcz for fruitful discussions about the existing literature on this topic and to the Stanford InfoLab for early feedback.

8. REFERENCES

[1] J. Allan. *Topic Detection and Tracking: Event-Based Information Organization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[2] H. Becker, M. Naaman, and L. Gravano. Beyond trending topics: Real-world event identification on twitter. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM 2011)*, pages 438–441, Barcelona, Spain, 2011.

[3] 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 (ACL 2012)*, pages 536–544, Jeju Island, South Korea, 2012.

[4] R. Li, S. Wang, and K. C.-C. Chang. Towards social data platform: Automatic topic-focused monitor for Twitter stream. In *Proceedings of the 39th International Conference on Very Large Data Bases (VLDB 2013)*, pages 1966–1977, Trento, Italy, 2013.

[5] 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 (SIGKDD 2010)*, pages 929–938, Washington, D.C., 2010.

[6] W. Magdy and T. Elsayed. Adaptive method for following dynamic topics on twitter. In *Proceedings of the 8th International AAAI Conference on Weblogs and Social Media (ICWSM 2014)*, pages 335–345, Ann Arbor, Michigan, 2014.

[7] S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to Twitter. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2010)*, pages 181–189, Los Angeles, California, 2010.

[8] E. Ruiz, V. Hristidis, and P. G. Ipeirotis. Efficient filtering on hidden document streams. In *Proceedings of the 8th International AAAI Conference on Weblogs and Social Media (ICWSM 2014)*, pages 446–455, Ann Arbor, Michigan, 2014.

[9] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes Twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International World Wide Web Conference (WWW 2010)*, pages 851–860, Raleigh, North Carolina, 2010.

[10] C. L. Wayne. Multilingual topic detection and tracking: Successful research enabled by corpora and evaluation. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*, 2000.

[11] J. Weng and B.-S. Lee. Event detection in Twitter. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM 2011)*, pages 401–408,

Barcelona, Spain, 2011.