Twitter-scale New Event Detection via K-term Hashing

Dominik Wurzer
School of Informatics
University of Edinburgh
d.s.wurzer
@sms.ed.ac.uk

Victor Lavrenko
School of Informatics
University of Edinburgh
vlavrenk
@inf.ed.ac.uk

Miles Osborne
Bloomberg
London
mosborne29
@bloomberg.net

Abstract

First Story Detection is hard because the most accurate systems become progressively slower with each document processed. We present a novel approach to FSD, which operates in constant time/space and scales to very high volume streams. We show that when computing novelty over a large dataset of tweets, our method performs 192 times faster than a state-of-the-art baseline without sacrificing accuracy. Our method is capable of performing FSD on the full Twitter stream on a single core of modest hardware.

1 Introduction

First Story Detection (FSD), also called New Event Detection, is the task of identifying the very first document in a stream to mention a new event[1]. FSD was introduced as port of the TDT[2] initiative and has direct applications in finance, news and government security. The most accurate approaches to FSD involve a runtime of $O(n^2)$ and cannot scale to unbounded high volume streams such as Twitter. We present a novel approach to FSD that operates in $O(1)$ per tweet. Our method is able to process the load of the average Twitter Firehose[3] stream on a single core of modest hardware while retaining effectiveness on par with one of the most accurate FSD systems. During the TDT program, FSD was applied to news wire documents and solely focused on effectiveness, neglecting efficiency and scalability. The traditional approach to FSD (Petrovic et al., 2010) computes the distance of each incoming document to all previously seen documents and the minimum distance determines the novelty score. Documents, whose minimum distance falls above a certain threshold are considered to talk about a new event and declared as first stories. Consequently, the computational effort increases with each document processed.

1.1 Related Work

Researchers have proposed a range of approaches to scale FSD to large data streams. Sankaranarayanan et al. (2009) were one of the first to apply FSD to Twitter. They reduced the volume by classifying documents into news/non-news and only compared to tweets within a 3-day window. They did not perform a quantitative evaluation of their approach. Sakaki et al. (2010) and Li et al. (2012) applied keyword filtering in conjunction with classification algorithms, which allowed them to efficiently detect certain events with high precision. These two approaches, although efficient and effective, require a user to explicitly define a set of keywords or to provide a set of examples that he wants to track. The approach cannot detect previously unknown events.

Phuvipadawat and Murata (2010), Ozdikis et al. (2012) and Cordeiro (2012), scale their systems by only considering tweets containing hashtags. Although efficient, this method don't consider 90% of the tweets (Petrovic, 2013), which limits their scope.

Cataldi et al. (2010), Weng et al.(2011) and Cordeiro (2012) use the degree of burstiness of terms during a time interval to detect new events. This approach is not suitable for FSD as events are detected with a time lag, once they grow in popularity.

Petrovic et al. (2010) were the first to demonstrate FSD on Twitter in constant time and space, while maintaining effectiveness comparable to those of pair-wise comparison systems. The key was to

[1] e.g. a natural disaster or a scandal

[2] TDT by NIST - 1998-2004. http://www.itl.nist.gov/iad/mig/tests/tdt/resources.html (Last Update: 2008)

[3] 5,700 tweets per second https://about.twitter.com/company (last updated: March 31, 2015)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2584–2589,
Lisbon, Portugal, 17-21 September 2015.

reduce the search space using Locality Sensitive Hashing (LSH). Each tweet was hashed, placing it into buckets that contain other similar tweets, which are subsequently compared. Operation in constant space was ensured by keeping the number of tweets per bucket constant. Because LSH alone performed ineffectively, Petrovic et al. (2010) additionally compared each incoming tweet with the k most recent tweets.

Allan et al. (2003) analysed scoring functions for novelty detection while focusing on their effectiveness. They presented a language-model (LM) based novelty measure using the KL divergence between the LM of a document and a single LM built on all previously scored documents, which they referred to as an aggregate measure language model. The idea of maintaining a single representation covering all previously seen documents, instead of performing pairwise comparisons with every document is closely related to the approach presented in this paper.

2 Approach

First Story Detection is a challenging task (Allan et al., 2000). The highest FSD accuracy is achieved by nearest-neighbour methods, where each incoming document (tweet) is compared to all documents that came before it, and the novelty score is determined by the most-similar documents in the past. This approach requires us to make $n-1$ comparisons[4] to determine the novelty of document d_n. The approach becomes progressively slower with each processed document, and cannot scale up to unbounded streams like Twitter. Prior attempts to speed up FSD involve organising previously seen documents $d_1 \ldots d_{n-1}$ into clusters (Allan et al., 1989) or LSH buckets (Petrovic et al., 2010). The document d_n is then compared only to past documents in the nearest cluster or LSH bucket, resulting in substantially fewer than n comparisons. While this approach is reasonably effective, it does lead to decreased accuracy, as potentially relevant past documents may exist in other clusters/buckets and would not be compared against.

2.1 First Story Detection in constant time

Our method computes the novelty of document d_n in a time that is constant with respect to n. The main difference from previous approaches is that we do not compare d_n to individual documents that came before it. Instead, we store the content of past documents $d_1 \ldots d_{n-1}$ in a single lookup table H_{n-1}. When d_n arrives, we count what fraction of its content is novel by looking it up in H_{n-1}. The number of lookups is polynomial in $|d_n|$ (the length of the document), and is independent of n.

Formally, let d_n denote the set of distinct words occurring in the n'th document in the stream. Let a k-term $t{=}\{w_1, w_2, \ldots\}$ denote a non-empty set of up to k distinct words. We define the content c_n to be the set of all k-terms that can be formed from the words in the document $d_n : c_n = \{\, t : t \subset d_n,\ |t| \leq k\}$. We estimate the novelty of document d_n as the proportion of novel k-terms, i.e. k-terms that do not appear in the history H_{n-1}:

$$\mathrm{N}(d_n) = \sum_{t \in c_n} \alpha_{|t|} \binom{|d_n|}{|t|}^{-1} \begin{Bmatrix} 1 : t \notin H_{n-1} \\ 0 : t \in H_{n-1} \end{Bmatrix} \quad (1)$$

Here $\alpha_{|t|}$ is the weight assigned to k-terms of size $|t|$, and $\binom{|d_n|}{|t|}$ is the total number of such k-terms formed from d_n. After the novelty is computed, we update the history H to include all k-terms formed from d_n:

$$H_n \leftarrow H_{n-1} \cup c_n \quad (2)$$

The computational cost of equations (1) and (2) is determined by the number of k-terms formed from the document d_n, and can be bounded at $O(|d_n|^k)$ operations. The complexity is manageable, as tweets are short and we keep k small.

2.2 Operating in constant time and space

We use a Bloom filter (Bloom, 1970) to maintain the history H_{n-1} of previously seen k-terms. For each k-term t we compute a 32-bit Murmur[5] hashcode, and use it as an index into a fixed-length bit-array. This ensures that both membership testing ($t{\in}H$) and history update can be performed in constant time. Constraining H to be a fixed-length array also means that our method operates in constant space, irrespective of the size of the stream and its vocabulary growth. In contrast to our method, previous approaches to FSD required more and more memory to maintain the history of the stream (see Figure 3).

[4] Each comparison requires $|d_n|$ scalar multiplications; $|d|$ denotes the number of distinct words in document d.

[5] https://en.wikipedia.org/wiki/MurmurHash

A potential downside of using a Bloom filter is that it introduces a small probability of false matches: a novel k-term t_i may collide with a previously observed k-term t_j and would be reported as non-novel. The probability of collision is directly proportional to the load factor of the Bloom filter, i.e. the fraction of non-zero bits in the array. By Heaps law (Egghe, 2007) the number of distinct words (and k-terms) will continue to grow and will eventually saturate the bit-array. To mitigate this problem, we introduce a deletion strategy: whenever the load factor exceeds a pre-determined threshold ρ, we zero out a random bit in H. This allows us to keep low the probability of false matches, at the cost of forgetting some previously-seen k-terms.

2.3 Parameter settings

We make the following parameter choices based on initial experiments on our training dataset. We set the maximum size of k-terms to be $k = 3$ and keep the Bloom filter load factor under $\rho = 0.6$. We tokenize the tweets on punctuation, treat all hashtags and mentions as words, stem them using the stemmer by Krovetz (1993), but do not remove stopwords. We optimise the weights $\alpha_1 \ldots \alpha_k$ using grid search on the same training data set.

3 Experiments

In a streaming setting, documents arrive one at a time on a continual basis. FSD requires computing a novelty score for each document in a single-pass over the data. High novelty scores indicate new topics. We use the standard TDT evaluation procedure (Allan, 2002) and the official TDT3 evaluation scripts with standard settings for evaluating FSD accuracy. The Detection Error Trade-off (DET) curve shows the trade-off between miss and false alarm probability for the full range of novelty scores. The normalized Topic Weighted Minimum Cost (C_{min}) is a linear combination of miss and false alarm probabilities, which allows comparing different methods based on a single value metric. Efficiency is measured by the throughput of tweets per second and the memory footprint. To ensure a fair comparison, all reported numbers are averaged over 5 runs on an idle machine using a single core (Intel-Xeon CPU with 2.27GHz).

3.1 Data set

We use the data set developed by Petrovic (2013), Petrovic et al. (2013b) as a test set, which consists of 27 topics and 116,000 tweets from the period of April till September 2011. Parameters were tuned using a sample of the data set annotated by Wurzer et al. (2015) as a training set.

3.2 Baselines

We compare our system (**k-term**) against 3 baselines.

UMass is a state-of-the-art FSD system, developed by Allan et al. (2000). It is known for its high effectiveness in the TDT2 and TDT3 competitions (Fiscus, 2001) and widely used as a benchmark for FSD systems (Petrovic et al., 2010; Kasiviswanathan et al., 2011; Petrovic 2013;). UMass makes use of an inverted index and k-nearest-neighbour clustering, which optimize the system for speed by ensuring a minimal number of comparisons. To maximise efficiency, we set-up UMass to operate in-memory by turning off its default memory mapping to disk. This ensures fair comparisons, as all algorithms operate in memory.

LSH-FSD is a highly-scalable system by Petrovic et al. (2010). It is based on Locality Sensitive Hashing (LSH) and claims to operate in constant time and space while performing on a comparable level of accuracy as UMass. We configure their system using the default parameters (Petrovic et al., 2010).

KL-FSD We also compare our approach with the aggregate measure language model (Allan et al., 2003) because it builds upon a similar principle.

3.3 Effectiveness and Efficiency

In Table 1, the UMass system shows state-of-the-art accuracy ($C_{min} = 0.79$, lower is better), but can only process 30 tweets per second. LSH-FSD operates 17 times faster, at the cost of a 13% decrease in accuracy ($C_{min} = 0.90$). Our system (k-term) operates on par with UMass in terms of accuracy, while being 197 times faster. KL-FSD, which is based on uni-grams, reveals the highest throughput at a considerable cost of efficiency ($C_{min} = 0.96$).

To further investigate accuracy we also compare the systems over the full range of the novelty thresholds illustrated by the DET plot in Figure 1.

Algorithm	Cmin	%-diff	tweets/sec	speed-up
UMass	0.7981	-	30	-
LSH-FSD	0.9061	-13.5%	500	17x
KL-FSD	0.9648	-21%	**6,600**	220x
k-term	**0.7966**	+0.2%	5,900	197x

Table 1: Comparing the effectiveness and efficiency of our system (k-term) with the 3 baselines

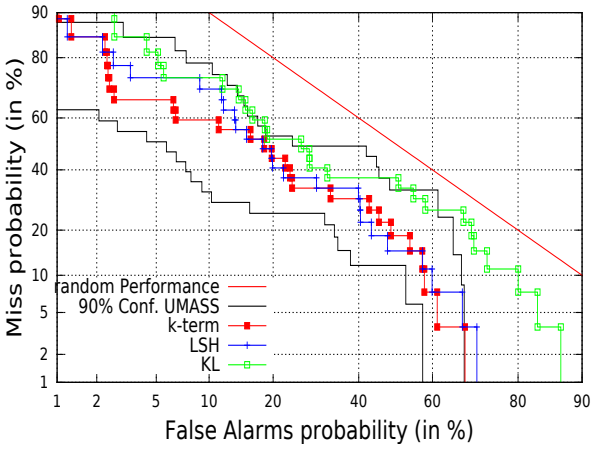

Figure 1: DET plot of UMass, Kl-FSD, LSH-FSD and k-term showing that LSH and k-term are statistically indistinguishable from UMass in terms of effectiveness;

Additionally we show the 90% confidence interval of UMass in two solid lines. We observer that both, FSD-LSH and our system (k-term) are statistically indistinguishable form UMass at any Miss-False Alarm trade-off point: their DET curves fall entirely within the 90% confidence interval of UMass. Note that DET curve of UMass is formed by the middle of it's 90% confidence interval curves. KL-FSD in contrast results in significantly worse accuracy than UMass in the mid-range and in particular the high recall area of the DET plot. We conclude that uni-grams are insufficient for determining the novelty of tweets.

3.4 FSD in constant time and space

High-volume streams require operation in constant time and space. Figure 2 compares the change in throughput of LSH-FSD, UMass and k-term as we process more and more tweets in the stream. Additionally, the plot also shows the average rate of tweets in the Twitter Firehose[6] at 5,787 tweets per second. Note that our system processes the equivalent of the full Twitter stream on a single core of modest hardware. This surpasses the throughput of LSH-FSD, a system known for high

[6]https://about.twitter.com/company (last updated: March 31, 2015)

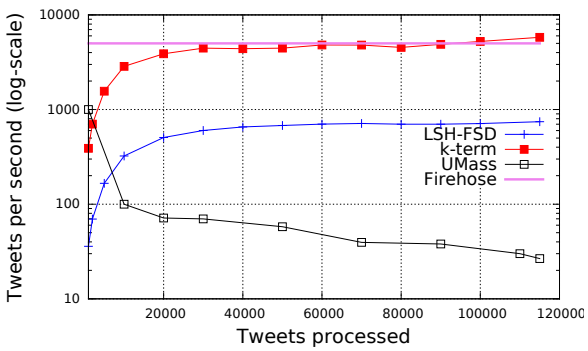

Figure 2: Throughput of UMass, LSH-FSD and k-term in comparison to the full Twitter stream (Firehose)

efficiency, by more than an order of magnitude. The throughput of LSH-FSD and k-term increases up until 20k documents because both approaches require initialisation of their data structures, which makes them slow when the number of documents is low. UMass has no initialization and performs the fastest when the number of documents is kept low. The pair-wise comparison of UMass causes it's throughput to decrease drastically with every new document. In Figure 2 we compare the memory requirements of k-term and LSH-FSD at different points in the stream. Although Petrovic et al. (2010) designed their system (LSH-FSD) to operate in constant space, we found that the memory requirement gradually increases with the number of documents processed, as seen in Figure 3. We hypothesise that this increase results from new terms added to the vocabulary. Our system has a strictly constant memory footprint.

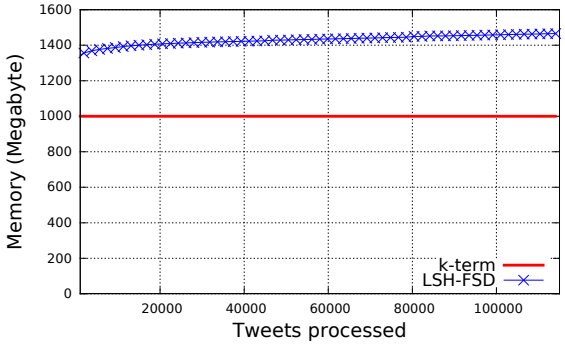

Figure 3: Space requirement for LSH-FSD and k-term; showing constant space for k-term

4 Conclusion

We presented an approach to FSD in a high volume streaming setting in constant time and space. Our approach computes novelty based on a single

lookup table that represents past documents. Shifting from direct comparisons with previous documents to comparisons with a single model that combines them, accounts for a great increase in efficiency. For the first time, we showed that it is possible to perform FSD on the full Twitter stream on a single core of modest hardware. This greatly outperforms state-of-the-art systems by an order of magnitude without sacrificing accuracy.

References

J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In SIGIR 03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pages 314 - 321. ACM Press, 2003

James Allan. 2002. Topic Detection and Tracking: Event-Based Information Organization. Kluwer Academic Publishers, Norwell, MA, USA.

James Allan, Victor Lavrenko and Hubert Jin. 2000. First story detection in TDT is hard. In Proceedings of the ninth international conference on Information and knowledge management. ACM.

James Allan, Ron Papka and Victor Lavrenko. 1998. On-line new event detection and tracking. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. ACM.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7), 422-426.

Cataldi, M., Caro, L. D., and Schifanella, C. (2010). Emerging topic detection on Twitter based on temporal and social terms evaluation. In Proceedings of the 10th International Workshop on Multimedia Data Mining, pages 4:1 - 4:10. ACM.

Cordeiro, M. (2012). Twitter event detection: Combining wavelet analysis and topic inference summarization. In Doctoral Symposium in Informatics Engineering, pages 123 - 138.

Leo Egghe. 2007. Untangling Herdan's law and Heaps' law: Mathematical and informetric arguments. Journal of the American Society for Information Science and Technology 58.5: 702-709.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbours: towards removing the curse of dimensionality. In Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC '98). ACM, New York, NY, USA.

Shiva Prasad Kasiviswanathan, Prem Melville, Arindam Banerjee, and Vikas Sindhwani. Emerging topic detection using dictionary learning. In Proceedings of the Twentieth ACM international conference on Information and knowledge management, 2011.

Robert Krovetz. 1993. Viewing morphology as an inference process. Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. ACM.

Li, R., Lei, K. H., Khadiwala, R., and Chang, K. C.-C. (2012). TEDAS: A Twitter-based event detection and analysis system. In Proceedings of 28th International Conference on Data Engineering, pages 1273 - 1276. IEEE Computer Society.

Li, C., Sun, A., and Datta, A. (2012b). Twevent: Segment-based event detection from tweets. In Proceedings of ACM Conference on Information and Knowledge Management. ACM.

Jimmy Lin, Rion Snow, and William Morgan. 2011. Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11). ACM, New York, NY, USA, 422-429.

S. Muthukrishnan. 2005. Data streams: Algorithms and applications. Now Publishers Inc.

Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing sporting events using twitter. In Proceedings of the 2012 ACM international conference on Intelligent User Interfaces (IUI '12). ACM, New York, NY, USA.

Ozdikis, O., Senkul, P., and Oguztuzun, H. (2012). Semantic expansion of hashtags for enhanced event detection in Twitter. In Proceedings of the 1st International Workshop on Online Social Systems.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. 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). Association for Computational Linguistics, Stroudsburg, PA, USA.

Sasa Petrovic. 2013. Real-time event detection in massive streams. Ph.D. thesis, School of Informatics, University of Edinburgh.

Sasa Petrovic, Miles Osborne, Richard McCreadie, Craig Macdonald, Iadh Ounis, and Luke Shrimpton. Can Twitter replace Newswire for breaking news? In Proc.of ICWSM, 2013b.

Raymond K. Pon, Alfonso F. Cardenas, David Buttler, and Terence Critchlow. 2007. Tracking multiple topics for finding interesting articles. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07). ACM, New York, NY, USA.

Phuvipadawat, S. and Murata, T. (2010). Breaking news detection and tracking in Twitter. In Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pages 120 - 123. IEEE Computer Society.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. In Proceedings of ACL.

Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., and Sperling, J. (2009). Twitterstand: news in tweets. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pages 42 - 51. ACM.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes Twitter users: real-time event detection by social sensors. In Proceedings of the 19th International Conference on World Wide Web, pages 851 - 860. ACM.

I. Soboroff, I. Ounis, and J. Lin. 2012. Overview of the trec-2012 microblog track. In Proceedings of TREC.

Jintao Tang, Ting Wang, Qin Lu, Ji Wang, and Wenjie Li. 2011. A Wikipedia based semantic graph model for topic tracking in blogosphere. In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Three (IJCAI'11).

TDT by NIST - 1998-2004. http://www.itl.nist.gov/iad/mig/tests/tdt/resources.html (Last Update: 2008)

Jianshu Weng, Erwin Leonardi, Francis Lee. Event Detection in Twitter. 2011. In Proceeding of ICWSM. AAAI Press.

Weng, J., Yao, Y., Leonardi, E., and Lee, F. (2011). Event detection in Twitter. In Proceedings of the 5th International Conference on Weblogs and Social Media, pages 401 - 408. The AAAI Press.

Dominik Wurzer, Victor Lavrenko, Miles Osborne. 2015. Tracking unbounded Topic Streams. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL) and the 7th International Joint Conference on Natural Language Processing, pages 1765 - 1773.

Xintian Yang, Amol Ghoting, Yiye Ruan, and Srinivasan Parthasarathy. 2012. A framework for summarizing and analysing twitter feeds. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '12). ACM, New York, NY, USA.