A Novel Burst-based Text Representation Model for Scalable Event Detection

Wayne Xin Zhao[†], **Rishan Chen**[†], **Kai Fan**[†], **Hongfei Yan**[†*] **and Xiaoming Li**[†‡]
[†]School of Electronics Engineering and Computer Science, Peking University, China
[‡]State Key Laboratory of Software, Beihang University, China
{batmanfly,tsunamicrs,fankaicn,yhf1029}@gmail.com, lxm@pku.edu.cn

Abstract

Mining retrospective events from text streams has been an important research topic. Classic text representation model (i.e., vector space model) cannot model temporal aspects of documents. To address it, we proposed a novel burst-based text representation model, denoted as BurstVSM. BurstVSM corresponds dimensions to bursty features instead of terms, which can capture semantic and temporal information. Meanwhile, it significantly reduces the number of non-zero entries in the representation. We test it via scalable event detection, and experiments in a 10-year news archive show that our methods are both effective and efficient.

1 Introduction

Mining retrospective events (Yang et al., 1998; Fung et al., 2007; Allan et al., 2000) has been quite an important research topic in text mining. One standard way for that is to cluster news articles as events by following a two-step approach (Yang et al., 1998): 1) represent document as vectors and calculate similarities between documents; 2) run the clustering algorithm to obtain document clusters as events.[1] Underlying text representation often plays a critical role in this approach, especially for long text streams. In this paper, our focus is to study how to represent temporal documents effectively for event detection.

Classical text representation methods, i.e., Vector Space Model (VSM), have a few shortcomings when dealing with temporal documents. The major one is that it maps one dimension to one term, which completely ignores temporal information, and therefore VSM can never capture the evolving trends in text streams. See the example in Figure 1, D_1 and D_2 may have a high similarity based on VSM due to the presence of some general terms (e.g., "election") related to U.S. presidential election, although general terms correspond to events in different periods (i.e., November 2004 and November 2008). Temporal information has to be taken into consideration for event detection. Another important issue is scalability, with the increasing of the number in the text stream, the size of the vocabulary, i.e., the number of dimensions in VSM, can be very large, which requires a considerable amount of space for storage and time for downstream processing.

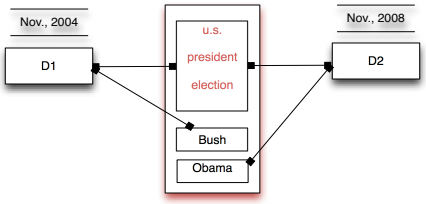

Figure 1: A motivating example. D_1 and D_2 are news articles about U.S. presidential election respectively in years 2004 and 2008.

To address these difficulties, in this paper, we propose a burst based text representation method for scalable event detection. The major novelty is to naturally incorporate temporal information into dimensions themselves instead of using external time decaying functions (Yang et al., 1998). We instantiate this idea by using bursty features as basic representation units of documents. In this paper, *bursty feature* refers to a sudden surge of the frequency of a single term in a text stream, and it is represented as the term itself together with the time interval during which the burst takes place. For example, `(Olympic, Aug-08-2008, Aug-24-2008)`[2] can be regarded as a bursty feature. We also call the term in a bursty

[*]Corresponding author.

[1]Post-processing may be also needed on the preliminary document clusters to refine the results.

[2]Beijing 2008 Olympic Games

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 43–47,
Jeju, Republic of Korea, 8-14 July 2012.

feature its bursty term. In our model, each dimension corresponds to a bursty feature, which contains both temporal and semantic information. Bursty features capture and reflect the evolving topic trends, which can be learnt by searching surge patterns in stream data (Kleinberg, 2003). Built on bursty features, our representation model can well adapt to text streams with complex trends, and therefore provides a more reasonable temporal document representation. We further propose a *split-cluster-merge* algorithm to generate clusters as events. This algorithm can run a mutli-thread mode to speed up processing.

Our contribution can be summarized as two aspects: 1) we propose a novel burst-based text representation model, to our best knowledge, it is the first work which explicitly incorporates temporal information into dimensions themselves; 2) we test this representation model via scalable event detection task on a very large news corpus, and extensive experiments show the proposed methods are both effective and efficient.

2 Burst-based Text Representation

In this section, we describe the proposed burst-based text representation model, denoted as *BurstVSM*. In BurstVSM, each document is represented as one vector as in VSM, while the major novelty is that one dimension is mapped to one bursty feature instead of one term. In this paper, we define a bursty feature f as a triplet (w^f, t_s^f, t_e^f), where w is the bursty term and t_s and t_e are the start and end timestamps of the bursty interval (period). Before introducting BurstVSM, we first discuss how to identify bursty features from text streams.

2.1 Burst Detection Algorithm

We follow the batch mode two-state automaton method from (Kleinberg, 2003) for bursty feature detection.[3] In this model, a stream of documents containing a term w are assumed to be generated from a two-state automaton with a low frequency state q_0 and a high frequency state q_1. Each state has its own emission rate (p_0 and p_1 respectively), and there is a probability for changing state. If an interval of high states appears in the optimal state sequence of some term, this term together with this interval is detected as a bursty feature. To obtain all bursty features in text streams, we can perform burst detection on each term in the vocabulary. Instead of using a fixed p_0 and p_1 in (Kleinberg, 2003), by following the moving average method (Vlachos et al., 2004) ,we parameterize p_0 and p_1 with the time index for each batch, formally, we have $p_0(t)$ and $p_1(t)$ for the tth batch. Given a term w, we use a sliding window of length L to estimate $p_0(t)$ and $p_1(t)$ for the tth batch as follows: $p_0(t) = \frac{\sum_{j \in W_t} N_{j,w}}{\sum_{j \in W_t} N_j}$ and $p_1(t) = p_0(t) \times s$, where $N_{j,w}$ and N_j are w 's document frequency and the total number of documents in jth batch respectively. s is a scaling factor lager than 1.0, indicating state q_1 has a faster rate, and it is empirically set as 1.5. W_t is a time interval $[\max(t - L/2, 0),\ \min(t + L/2, N)]$, and the length of moving window L is set as 180 days. All the other parts remain the same as in (Kleinberg, 2003). Our detection method is denoted as *TVBurst*.

[3]The news articles in one day is treated as a batch.

2.2 Burst based text representation models

We apply TVBurst to all the terms in our vocabulary to identify a set of bursty features, denoted as $\mathcal{B}$. Given $\mathcal{B}$, a document $\mathbf{d}_i(t)$ with timestamp t is represented as a vector of weights in ***bursty feature dimensions***:

$$\mathbf{d}_i(t) = (d_{i,1}(t), d_{i,2}(t), ..., d_{i,|\mathcal{B}|}(t)).$$

We define the jth weight of $\mathbf{d}_i$ as follows

$$d_{i,j} = \begin{cases} \text{tf-idf}_{i,w^{\mathcal{B}_j}}, & \text{if } t \in [t_s^{\mathcal{B}_j}, t_e^{\mathcal{B}_j}], \\ 0, & \text{otherwise.} \end{cases}$$

When the timestamp of $\mathbf{d}_i$ is in the bursty interval of $\mathcal{B}_j$ and contains bursty term $w^{\mathcal{B}_j}$, we set up the weight using common used *tf-idf* method. In BurstVSM, each dimension is mapped to one bursty feature, and it considers both semantic and temporal information. One dimension is active only when the document falls in the corresponding bursty interval. Usually, a document vector in BurstVSM has only a few non-zero entries, which makes computation of document similarities more efficient in large datasets compared with traditional VSM.

The most related work to ours is the boostVSM introduced by (He et al., 2007b), it proposes to weight different term dimensions with corresponding bursty scores. However, it is still based on term dimensions and fails to deal with terms with multiple bursts. Suppose that we are dealing with a text collection related with U.S. presidential elections, Fig. 2 show sample dimensions for these three methods. In BurstVSM, one term with multiple bursts will be naturally mapped to different dimensions. For example, two bursty features (presidential, Nov., 2004) and (presidential, Nov., 2008) correspond to different dimensions in BurstVSM, while

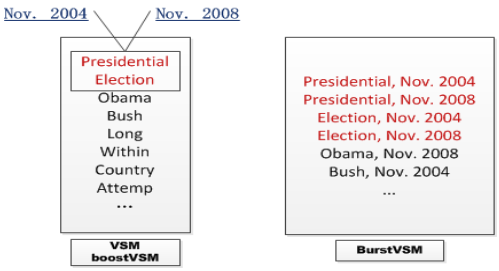

Figure 2: One example for comparisons of different representation methods. Terms in red box correspond to multiple bursty periods.

Table 1: Summary of different representation models. Here dimension reduction refers to the reduction of non-zero entries in representation vector.

	semantic information	temporal information	dimension reduction	trend modeling
VSM	✓	×	×	bad
boostVSM	✓	partially	×	moderate
BurstVSM	✓	✓	✓	good

VSM and boostVSM cannot capture such temporal differences. Some methods try to design time decaying functions (Yang et al., 1998), which decay the similarity with the increasing of time gap between two documents. However, it requires efforts for function selection and parameters tuning. We summarize these discussions in Table 1.

3 *split-cluster-merge* algorithm for event detection

In this section, we discuss how to cluster documents as events. Since each document can be represented as a burst-based vector, we use cosine function to compute document similarities. Due to the large size of our news corpus, it is infeasible to cluster all the documents straightforward. We develop a heuristic clustering algorithm for event detection, denoted as *split-cluster-merge*, which includes three main steps, namely split, cluster and merge. The idea is that we first split the dataset into small parts, then cluster the documents of each part independently and finally merge similar clusters from two consecutive parts. In our dataset, we find that most events last no more than one month, so we split the dataset into parts by months. After splitting, clustering can run in parallel for different parts (we use *CLUTO*[4] as the clustering tool), which significantly reduces total time cost. For *merge*, we merge clusters in consecutive months with an empirical threshold of 0.5. The final clusters are returned as identified events.

4 Evaluation

4.1 Experiment Setup

We used a subset of 68 millon deduplicated timestamped web pages generated from this archive (Huang et al., 2008). Since our major focus is to detect events from news articles, we only keep the web pages with keyword "news" in *URL* field. The final collection contains $11,218,581$ articles with total $1,730,984,304$ tokens ranging from 2000 to 2009. We run all the experiments on a 64-bit linux server with four Quad-Core AMD Opteron(tm) Processors and 64GB of RAM. For *split-cluster-merge* algorithm, we implement the *cluster* step in a multi-thread mode, so that different parts can be processed in parallel.

4.2 Construction of test collection

We manually construct the test collection for event detection. To examine the effectiveness of event detection methods in different grains, we consider two type of events in terms of the number of relevant documents, namely significant events and moderate events. A significant event is required to have at least 300 relevant docs, and a moderate event is required to have $10 \sim 100$ relevant docs. 14 graduate students are invited to generate the test collection, starting with a list of 100 candidate seed events by referring to Xinhua News.[5] For one target event, the judges first construct queries with temporal constraints to retrieve candidate documents and then judge wether they are relevant or not. Each document is assigned to three students, and we adopt the majority-win strategy for the final judgment. Finally, by removing all candidate seed events which neither belong to significant events nor moderate events, we derive a test collection consisting of 24 significant events and 40 moderate events.[6]

4.3 Evaluation metrics and baselines

Similar to the evaluation in information retrieval , given a target event, we evaluate the quality of the returned "relevant" documents by systems. We use average precision, average recall and mean average precision(MAP) as evaluation metrics. A difference is that we do not have queries, and the output of a system is a set of document clusters. So for a system, given an event in golden standard, we first select the cluster (the system generates) which has the

[4] www.cs.umn.edu/~karypis/cluto

[5] http://news.xinhuanet.com/english

[6] For access to the code and test collection, contact Xin Zhao via batmanfly@gmail.com.

Table 2: Results of event detection. Our proposed method is better than all the other baselines at confidence level 0.9.

	Signifcant Events				**Moderate Events**			
	P	*R*	*F*	*MAP*	*P*	*R*	*F*	*MAP*
timemines-χ^2(nouns)	0.52	0.2	0.29	0.11	0.22	0.27	0.24	0.09
timemines-χ^2(NE)	0.61	0.18	0.28	0.08	0.27	0.25	0.26	0.13
TVBurst+boostVSM	0.67	0.44	0.53	0.31	0.22	0.39	0.28	0.13
swan+BurstVSM	0.74	0.56	0.64	0.48	0.39	0.54	0.45	0.38
kleiberg+BurstVSM	0.68	0.63	0.65	0.52	0.35	0.53	0.42	0.36
TVBurst+BurstVSM	**0.78**	**0.69**	**0.73**	**0.63**	**0.4**	**0.61**	**0.48**	**0.39**

Table 3: Comparisons of average intra-class and inter-class similarity.

Methods	**Significant Events**		**Moderate Events**	
	Intra	Inter	Intra	Inter
TVBurst+boostVSM	0.234	0.132	0.295	0.007
TVBurst+BurstVSM	0.328	0.014	0.480	0.004

Table 4: Comparisons of observed runtime and storage.

	boostVSM	**BurstVSM**
Aver. # of non-zero entries per doc	149	14
File size for storing vectors (gigabytes)	3.74	0.571
Total # of *merge*	10,265,335	9,801,962
Aver. *cluster* cost per month (sec.)	355	55
Total *merge* cost (sec.)	2,441	875
Total time cost (sec.)	192,051	4,851

most relevant documents, then sort the documents in the descending order of similarities with the cluster centroid and finally compute P, R ,F and MAP in this cluster. We perform Wilcoxon signed-rank test for significance testing.

We used the event detection method in (Swan and Allan, 2000) as baseline, denoted as timemines-χ^2. As (Swan and Allan, 2000) suggested, we tried two versions: 1) using all nouns and 2) using all named entities. Recall that BurstVSM relies on bursty features as dimensions, we tested different burst detection algorithms in our proposed BurstVSM model, including *swan* (Swan and Allan, 2000), *kleinberg* (Kleinberg, 2003) and our proposed *TVBurst* algorithm.

4.4 Experiment results

Preliminary results. In Table 2, we can see that 1) *BurstVSM* with any of these three burst detection algorithms is significantly better than timemines-χ^2, suggesting our event detection method is very effective; 2) *TVBurst* with *BurstVSM* gives the best performance, which suggests using moving average base probability will improve the performance of burst detection. We use *TVBurst* as the default burst detection algorithm in later experiments.

Then we compare the performance of different text representation models for event detection, namely *BurstVSM* and *boostVSM* (He et al., 2007b; He et al., 2007a).[7] For different representation models, we use *split-cluster-merge* as clustering algorithm. Table 2 shows that BurstVSM is much effecitve than boostVSM for event detection. In fact, we empirically find boostVSM is appropriate for clustering documents in a coarse grain (e.g., in topic level) but not for event detection.

Intra-class and inter-class similarities. In our methods, event detection is treated as document clustering. It is very important to study how similarities affect the performance of clustering. To see why our proposed representation methods are better than *boostVSM*, we present the average intra-class similarity and inter-class similarity for different events in Table 3.[8] We can see BurstVSM results in a larger intra-class similarity and a smaller inter-class similarity than boostVSM.

Analysis of the space/time complexity. We further analyze the space/time complexity of different representation models. In Table 4. We can see that BurstVSM has much smaller space/time cost compared with boostVSM, and meanwhile it has a better performance for event detection (See Table 2). In burst-based representation, one document has fewer non-zero entries.

Acknowledgement. The core idea of this work is initialized and developped by Kai Fan. This work is partially supported by HGJ 2010 Grant 2011ZX01042-001-001, NSFC Grant 61073082 and 60933004. Xin Zhao is supported by Google PhD Fellowship (China). We thank the insightful comments from Junjie Yao, Jing Liu and the anonymous reviewers. We have developped an online Chinese large-scale event search engine based on this work, visit http://sewm.pku.edu.cn/eventsearch for more details.

[7] We use the same parameter settings in the original paper.

[8] For each event in our golden standard, we have two clusters: relevant documents and non-relevant documents(within the event period).

References

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*.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Huan Liu, and Philip S. Yu. 2007. Time-dependent event hierarchy construction. In *SIGKDD*.

Q. He, K. Chang, and E. P. Lim. 2007a. Using burstiness to improve clustering of topics in news streams. In *ICDM*.

Qi He, Kuiyu Chang, Ee-Peng Lim, and Jun Zhang. 2007b. Bursty feature representation for clustering text streams. In *SDM*.

L. Huang, L. Wang, and X. Li. 2008. Achieving both high precision and high recall in near-duplicate detection. In *CIKM*.

J. Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*.

Russell Swan and James Allan. 2000. Automatic generation of overview timelines. In *SIGIR*.

Michail Vlachos, Christopher Meek, Zografoula Vagena, and Dimitrios Gunopulos. 2004. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*.

Yiming Yang, Tom Pierce, and Jaime Carbonell. 1998. A study of retrospective and on-line event detection. In *SIGIR*.