Context Modeling for Ranking and Tagging Bursty Features in Text Streams

Wayne Xin Zhao[†], Jing Jiang[‡], Jing He[†], Dongdong Shan[†], Hongfei Yan[†],Xiaoming Li[†]
[†]School of Electronics Engineering and Computer Science, Peking University, China
[‡]School of Information Systems, Singapore Management University, Singapore
{zhaoxin,hj,sdd,yhf}@net.pku.edu.cn, jingjiang@smu.edu.cn, lxm@pku.edu.cn

ABSTRACT

Bursty features in text streams are very useful in many text mining applications. Most existing studies detect bursty features based purely on term frequency changes without taking into account the semantic contexts of terms, and as a result the detected bursty features may not always be interesting or easy to interpret. In this paper we propose to model the contexts of bursty features using a language modeling approach. We then propose a novel topic diversity-based metric using the context models to find newsworthy bursty features. We also propose to use the context models to automatically assign meaningful tags to bursty features. Using a large corpus of a stream of news articles, we quantitatively show that the proposed context language models for bursty features can effectively help rank bursty features based on their newsworthiness and to assign meaningful tags to annotate bursty features.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Text Mining

General Terms

Algorithms, Experimentation

Keywords

bursty features, bursty features ranking, bursty feature tagging, context modeling

1. INTRODUCTION

Pioneered by Kleinberg's seminal work on bursty structures in streams [5], a particularly important problem in temporal text mining is to find unusual surges of activities related to an event from text streams such as one's incoming emails [5] or a search engine's query logs [8]. These bursts are usually identified by tracing the time series of the frequencies of single terms or phrases [5, 8, 3, 4]. Following [3], we use *bursty feature* to refer to a sudden surge of the frequency of a single term or phrase in a text stream, and represent a bursty feature by the term or phrase itself together with the time interval during which the burst takes place. For example, (`Olympic`, `Aug-08-2008`, `Aug-24-2008`) can be regarded as a bursty feature. We also call the term or phrase in a bursty feature its *bursty term*.

Many effective techniques have been proposed for bursty feature detection, such as the moving average-based method in [8], the two-state automaton model in [5], and the parameter-free methods in [3] and [6]. What these techniques share in common is that they identify bursty features purely based on the frequency changes in the time series of the term or phrase under consideration. However, they do not try to capture the semantic meanings of these bursty features. In particular, they do not take into account the semantic context in which a burst of a term or phrase happens.

As a result, there are several limitations with the kind of bursty features identified by these methods: **1)** A bursty feature identified purely through frequency changes may not correspond to an interesting or newsworthy event. For example, there may be a peak of the word `Wednesday` on every Wednesday, but it is clearly not an interesting burst. Including such bursty features may hurt the performance of downstream text mining applications such as event detection. **2)** Current representation of bursty features is too simple. Although currently bursty features are usually used as input to downstream applications such as event detection and document search, in some situations we may also want to directly display bursty features to the end-users. For example, at microblogging sites such as Twitter, to capture the most recent trends in the online social space, we may want to constantly display and update a list of buzz words, which can be thought of as the the most recent bursty terms from the microblogging streams. However, existing work simply represents a bursty feature as a single term or phrase and its bursty interval. There is no additional semantic annotation attached to it, which makes it hard for an end-user to interpret and understand a bursty feature. **3)** Downstream applications may not be able to make full use of the input bursty features if no context information is provided. An example is event detection through clustering of bursty features [3, 4]. In these methods, correlated bursty features are grouped together to represent an event. However, using only bursty terms to describe an event may not be sufficient to capture the background and context of the event. For example, the bursty terms `bird` and `flu` may be clustered to represent an event [3], but other closely related terms such as `China` and `chicken` are also important but unlikely to be

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

included in the bursty feature cluster simply because they may not be detected as bursty terms.

To summarize, there is a lack of consideration of context in current bursty feature detection methods. We argue that it is important to take account of the context in order to address the limitations above. We therefore propose a language modeling approach to context modeling for bursty features, and show how the context models can be used for two novel tasks, namely, *bursty feature ranking* and *bursty feature tagging*.

2. CONTEXT MODELING FOR BURSTY FEATURES

2.1 Preliminaries

In this paper, we define a *bursty feature* f as a triplet (w, t_s, t_e), where w is the bursty term and t_s and t_e are the start and end timestamps of the bursty interval. Conceptually, a *context* is defined as "the situation within which something exists or happens, and that can help explain it."[1] For a bursty feature, intuitively its context should help explain the background and details of the event this bursty feature is related to. Formally, we define the *context language model* θ_f of a bursty feature f to be a multinomial distribution over the vocabulary $\mathcal{V}$. In other words, for $v \in \mathcal{V}$, $\theta_{f,v}$ is the probability of generating word v from this context language model, and $\sum_v \theta_{f,v} = 1$.

2.2 Bursty Feature Detection

We use the batch mode two-state automaton method from [5] for bursty feature detection. In this model, a stream of documents containing a term v 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, 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, and the *weight* of a bursty feature $f = (w, t_s, t_e)$ is defined as

$$\text{weight}(f) = \sum_{t=t_s}^{t_e} \Big[\sigma(0, r_t, s_t) - \sigma(1, r_t, s_t)\Big], \tag{1}$$

where $\sigma(\cdot, r_t, s_t)$ is the cost function of the t'th batch[2] and 0 and 1 represent the low and the high states respectively. This weight can be seen as an indicator of the strength of burstiness for this bursty feature. In Section 5.2, we will use this weight to rank bursty features as a baseline.

2.3 Generative Models of Context

Given a bursty feature f, we assume that we have some text that can provide the context for f. We represent this text as a bag of words $\mathcal{C}_f = \{w_1^f, w_2^f, \ldots, w_M^f\}$. We assume that $\mathcal{C}_f$ is generated from θ_f.

Simple Context Generation

The simplest way to generate $\mathcal{C}_f$ is to assume that words from $\mathcal{C}_f$ are generated only from θ_f. Based on maximum likelihood estimation, we have

$$\theta_{f,v} = p(v|\theta_f) = \frac{c(v, \mathcal{C}_f)}{|\mathcal{C}_f|}. \tag{2}$$

One problem with this generative model is that it assumes all words are generated by a single underlying context language model, even though stop words and some other words may not be representative of the context for the bursty feature.

Two-Mixture Context Generation

Intuitively, the text that provides contextual information for a bursty feature can be decomposed into two main parts: contextual words closely related to the bursty feature and general background words. So we argue that a more reasonable generative model is to assume that a word in $\mathcal{C}_f$ is generated either from θ_f or from a general background model θ_B. This kind of two-mixture models have been shown to be effective in pseudo relevance feedback for information retrieval [9]. Formally, with the two-mixture model, we have

$$\log p(\mathcal{C}_f|\theta_f, \theta_B) = \sum_{v \in \mathcal{V}} c(v, \mathcal{C}_f) \log\Big((1-\lambda)p(v|\theta_f) + \lambda p(v|\theta_B)\Big).$$

where λ is the probability that a word in $\mathcal{C}_f$ is generated from the general background model and $c(v, \mathcal{C}_f)$ be the count of word v in $\mathcal{C}_f$.

We propose to use different background models for different time periods in order to adapt to the dynamic nature of text streams. Here we use the documents during the bursty interval of f to estimate the background model θ_B^f for f. Let $\mathcal{B}_f$ denote the bag of words from *all* documents within the bursty interval of f (including those that do not contain the bursty term of f). Let $c(v, \mathcal{B}_f)$ denote the count of word v in $\mathcal{B}_f$. The estimation formula for θ_B^f is as follows:

$$p(v|\theta_B^f) \quad = \quad \frac{c(v, \mathcal{B}_f)}{|\mathcal{B}_f|}. \tag{3}$$

After deriving θ_B^f for f and fixing λ, we can use the expectation maximization (EM) algorithm [2] to estimate the context language model for f. The updating formulas of the E-step and the M-step are shown below:

$$\text{E-step:} \qquad \alpha^{(n)}(v) = \frac{(1-\lambda)p^{(n)}(v|\theta_f)}{(1-\lambda)p^{(n)}(v|\theta_f) + \lambda p(v|\theta_B^f)}.$$

$$\text{M-step:} \qquad p^{(n+1)}(v|\theta_f) = \frac{c(v, \mathcal{C}_f)\alpha^{(n)}(v)}{\sum_{v' \in \mathcal{V}} c(v', \mathcal{C}_f)\alpha^{(n)}(v')}.$$

Selecting Context Units

To select the context, we consider two options: **1) Document-level context**, to use the words from all the documents within the bursty interval $[t_s, t_e]$ that also contain the bursty term w as $\mathcal{C}_f$. **2) Sentence-level context**, to use the words from all the sentences within the bursty interval that also contain the bursty term as $\mathcal{C}_f$. To estimate the context language model for a bursty feature, we have the following four variations: ***SM-sen***: Using the simple context generation model with sentence-level context; ***SM-doc***: Using the simple context generation model with document-level context; ***TM-sen***: Using the two-mixture context generation model with sentence-level context; ***TM-doc***: Using the two-mixture context generation model with document-level context.

[1] http://dictionary.cambridge.org/dictionary/british/context_1

[2] The t'th batch contain r_t relevant documents out of a total of s_t documents.

3. RANKING BURSTY FEATURES WITH CONTEXT MODELS

In this section, we make use of the context language models of bursty features proposed in the previous section together with topic analysis for bursty feature ranking. Intuitively, a bursty feature is interesting to a general audience if it is related to a specific event that does not happen very often, e.g. `tsunami`. On the other hand, bursty terms such as `Wednesday` and `October` are not interesting because their semantic contexts are likely to contain various topics and events that can generally happen on other days of a week and in other months of a year. In the context of topic analysis, an uninteresting bursty feature such as `Wednesday` is therefore likely to have a diverse distribution of topics. To capture this intuition, we can define a *topic diversity* measure for bursty features and then rank bursty features in increasing order of their topic diversity. The top-ranked ones are then likely to be interesting or newsworthy bursty features.

Formally, following a standard topic modeling approach, let $\mathcal{E}$ denote a set of topics, where each topic e is represented as a multinomial word distribution over the vocabulary, denoted as θ_e. Such topics can be obtained by applying standard LDA [1]. We first define a normalized similarity measure between a bursty feature f and a topic e as follows:

$$\mathrm{sim}(f,e) = \frac{\exp\big(-\mathrm{div}(\theta_f||\theta_e)\big)}{\sum_{e'\in\mathcal{E}}\exp\big(-\mathrm{div}(\theta_f||\theta_{e'})\big)}.$$

Here $\mathrm{div}(\theta_f||\theta_e)$ denotes the KL-divergence between two word distributions θ_f and θ_e, and θ_f is the context language model for the bursty feature f.

Since this similarity measure is normalized over all topics, we can also think of it as the probability of a topic given a bursty feature f, that is, we can define

$$p(e|f) \quad = \quad \mathrm{sim}(f,e).$$

Finally, we define the topic diversity (TopicDiv) of a bursty feature f to be the entropy of this topic distribution:

$$\mathrm{TopicDiv}(f) \quad = \quad -\sum_{e\in\mathcal{E}} p(e|f)\log p(e|f). \tag{4}$$

Our hypothesis is that a bursty feature with a small TopicDiv measure is likely to be related to a newsworthy event and therefore of high quality. As we will show in Section 5, this topic diversity measure can indeed help boost more interesting bursty features to the top of the list.

4. AUTOMATIC TAGGING OF BURSTY FEATURES

In this section we consider another task based on context language models: bursty feature tagging. In this tagging task, we consider the whole vocabulary $\mathcal{V}$ as our candidate tag set. Given a bursty feature f together with its estimated context language model θ_f, we formulate the automatic tagging problem as a term ranking problem where we want to rank all terms in $\mathcal{V}$ and select the top-K terms as tags for f. Note that this tagging problem is different from the part-of-speech tagging problem in NLP. It is similar to social tagging in Web 2.0 except that it is done automatically.

A Naïve Method

A naïvely method is to rank candidate tags (all terms in $\mathcal{V}$) based on their probabilities in the context language model of f, namely, to rank tags in decreasing order of the following scoring function for a candidate tag v given f:

$$\mathrm{score}_{\mathrm{naive}}(v,f) = p(v|\theta_f). \tag{5}$$

A PMI-Based Method

If we treat the context language model of a bursty feature as a topic, then the bursty feature tagging problem is similar to the problem of labeling topics in topic modeling. We can therefore apply an existing method in [7] that has been shown to be effective in labeling topics using pointwise mutual information. Formally, we can rank a candidate tag v given f according to the expectation of the pointwise mutual information between v and a term v' under the context language model θ_f of f, as defined below:

$$\mathrm{score}_{\mathrm{PMI}}(v,f) \stackrel{\mathrm{rank}}{=} \sum_{v'\in\mathcal{V}} p(v'|\theta_f)\cdot\mathrm{PMI}(v,v'|\mathcal{B}_f), \tag{6}$$

where $\mathcal{B}_f$ is background for f within its bursty interval as defined in Section 2.

5. EMPIRICAL EVALUATION

5.1 Data Collection and Preprocessing

We crawled all the articles from the news archives at `http://english.peopledaily.com.cn` in a time span of two and a half years from January 6, 2005 to July 1, 2007. In total we obtained 179,672 articles. After pre-processing (stemming and stopword removal), we obtained a vocabulary of 30,718 terms. We applied the two-state automaton method described in [5] to obtain a list of bursty features. Articles from the same day were grouped into one batch. On average, there are about 200 articles per day. Parameter setting follows [5]. We obtained 12,115 bursty features of which there are 1,024 unique bursty terms in total.

5.2 Evaluation of Bursty Feature Ranking

As a first step of our ranking method, we need to get a set of topics $\mathcal{E}$ from the whole text collection. We used the popular open source topic modeling package GibbsLDA++[3]. We selected top-200 bursty features discovered by the method in [5] without duplicate bursty terms. We asked two human judges to separately judge the interestingness or newsworthiness of the 200 bursty features. A third human judge would give the final decision when there was disagreement. In the end we obtained 156 newsworthy bursty features and 44 noisy bursty features. For comparison, we consider a baseline method that ranks bursty features by their burstiness weights as defined in Equation (1).

To measure the ranking performance, we use two metrics: `P@k`, defined as the percentage of bursty features that are labeled as newsworthy among the top-k bursty features ranked by a method; `#IP`, defined as the number of bursty feature pairs that are inversely ordered by this method, i.e. when a noisy bursty feature has been ranked higher than a newsworthy one.

[3]http://gibbslda.sourceforge.net

Table 1: Comparison of the methods using #IP.

Methods	BL	SM-sen	TM-sen	SM-doc	TM-doc
#IP	4803	297	239	268	**216**

Table 2: Comparison of the performances using P@k.

Methods	P@5	P@10	P@25	P@50	P@100
Baseline	1	0.8	0.44	0.5	0.68
TM-doc	1	1	1	1	0.99

In Table 1 we show the #IP of the different methods and in Table 2 we show the P@k of the baseline and our *TM-doc* method for a range of k. In these results, we set the number of topics of LDA to 150, and for the two-mixture models we fixed $\lambda = 0.3$. We can see from Table 1 that there is clearly a big gap between the baseline method and our methods in terms of #IP. Our methods have much fewer inverse pairs, which shows that our ranking can push the interesting or newsworthy bursty features to the top of the ranked list. It also shows that the two-mixture model generally performs better than the simple model, and document-level context performs better than sentence-level context. Overall, *TM-doc* is the best among the four variations of our method. In Table 2, we can see that in terms of P@k, *TM-doc* also significantly outperforms the baseline. In particular, precision remains close to 1 up to at least P@100 for *TM-doc*, but P@25 is below 0.5 for the baseline.

5.3 Evaluation of Bursty Feature Tagging

In this section, we turn to the second task of automatically tagging bursty features and report our quantitative evaluation results. Similar to the ranking task, we also need to create a test set for the tagging task. We randomly selected 50 newsworthy bursty features from Section 5.2. We adopted the pooling strategy commonly used in information retrieval. For each bursty feature, we applied the different variations of our method and took the union of the top-50 tags assigned by the different variations. The human judges were asked to assign a relevance score to each tag given the same information of the bursty feature as they had in Section 5.2. We used three levels of relevance and their scores, namely, 2 for *relevant*, 1 for *marginally relevant* and 0 for *non-relevant*. We thus obtained a test set of 50 bursty features with their gold standard tags and corresponding scores.

The tagging task aims to generate interpretable, comprehensive and discriminative tags. For each bursty feature, a tagging method returns a ranked list of tags, and the quality of the top few tags is the most important. It is similar to Web search where the relevance of the top few search results is the most important. We therefore use a modified version of the popular nDCG measure as our evaluation metric. In particular, given a bursty feature f and its ranked list of tags given by method $\mathcal{M}$, we define our nDCG@k as follows:

$$\texttt{nDCG@k}(f, \mathcal{M}) = \frac{\sum_{i=1}^{k} \frac{1}{\log_2(i+1)} \cdot \text{score}(\mathcal{M}_{f,i})}{\texttt{iDCG@k}(f)},$$

where $\mathcal{M}_{f,i}$ is the i'th tag for bursty feature f given by method $\mathcal{M}$, score$(\cdot)$ returns the relevance score of a tag as defined above, and iDCG@k(f) is the maximum DCG score at k for f assuming an ideal ranking.

Table 3: Performance of bursty feature tagging.

Method	Naïve		PMI	
	nDCG@5	nDCG@10	nDCG@5	nDCG@10
SM-sen	0.649	0.619	0.671	0.669
SM-doc	0.685	0.632	0.728	0.692
TM-sen	0.636	0.638	0.667	0.672
TM-doc	**0.746**	**0.709**	**0.741**	**0.717**

Given a set of bursty features $\mathcal{F}$, we define nDCG@k$(\mathcal{M})$ to be the average of nDCG@k$(f, \mathcal{M})$ over all $f \in \mathcal{F}$ and use this as the overall performance measure for method $\mathcal{M}$. We consider $k = 5$ and $k = 10$.

We show the basic results in Table 3. We set $\lambda = 0.9$ in the two-mixture models in this case because tagging task needs a more discriminative context language model to generates discriminative tags. We can see from the table that overall we achieved good nDCG@5 and nDCG@10 values. Among the four variations, again *TM-doc* performed the best among the four variations, confirming that document-level context coupled with two-mixture model gives the best context language model for a bursty feature for the purpose of tagging. We can also see that while the PMI-based tagging method consistently improved the performance for *SM-sen*, *TM-sen* and *SM-doc*, for *TM-doc*, it did not show any advantage. It suggests that the context language model estimated by *TM-doc* can already provide a good keyword representation of the bursty feature.

ACKNOWLEDGMENT

The authors Xin Zhao, Hongfei Yan and Xiaoming Li are partially supported by NSFC under the grant No. 70903008 and 60933004, CNGI grant No. 2008-122, 863 Program No. 2009AA01Z143, and the Open Fund of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2010KF-03, Beihang University.

6. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 2003.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society,1977.*

[3] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *VLDB*, 2005.

[4] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *SIGIR*, 2007.

[5] J. Kleinberg. Bursty and hierarchical structure in streams. *DMKD*, 2003.

[6] T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos. On burstiness-aware search for document sequences. In *SIGKDD*, 2009.

[7] Q. Mei, X. Shen, and C. Zhai. Automatic labeling of multinomial topic models. In *SIGKDD*, 2007.

[8] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*, 2004.

[9] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM*, 2001.