

Identification of Bursts in a Document Stream

Toshiaki FUJIKI¹, Tomoyuki NANNO¹,
Yasuhiro SUZUKI¹ and Manabu OKUMURA²

¹ Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology

² Precision and Intelligence Laboratory, Tokyo Institute of Technology

Abstract. We propose a method for extracting ‘burst of a word’ relating to a popular topic in a document stream in which we do not assume that document arrives at a uniform rate. We regard blogs and BBSs as document streams to apply the method originally proposed by Kleinberg. However, since Kleinberg’s algorithm cannot be applied to document streams where the distribution of documents is not uniform, we extended the method to be able to apply to blogs and BBSs. We also describe experiments for blog and BBS with our method and discuss the results.

1 Introduction

BBSs (Bulletin Board Systems) and blogs (Weblogs) are considered important information sources since people express their personal opinions and experiences actively. However, it is difficult to identify useful information from these information sources, and a technique for automatically extracting useful information is needed. In previous research, Wakefield attempted to predict stock prices from reputations on BBSs[1], and Matsuo tried to summarize replies on BBSs to obtain useful information using the reply-replied structure of BBS[2].

The burst detection algorithm proposed by Kleinberg[3] can be used to extract a topic word in a certain period of time from these information sources. This algorithm regards these information sources as document streams (i.e., a set of documents with time information), and models the phenomenon that the frequency of a specific word increases rapidly when a topic attracts attention. Though Kleinberg originally applied the algorithm to identifying events from sets of e-mails, the algorithm can be used on different kinds of document streams. Kumar applied the method to hyperlinks between blog communities and extracted bursty evolution of communities[4], while Mane applied it to scientific publications to generate maps of major research topics[5].

Originally, we tried to apply the algorithm to blogs that had been collected by our blog collecting system[6], however, it did not work properly. This may have been because Kleinberg’s algorithm is based on the assumption that documents appear uniformly, while the distribution of blog entries collected by our system is not necessarily uniform. Therefore, we present our method that is an extension of Kleinberg’s method to deal with blogs and BBSs which do not exhibit uniform distribution. We also describe the experiments for blogs and BBSs with our proposed method and discuss the results.

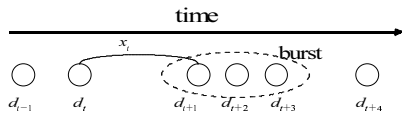


Fig. 1. Document stream

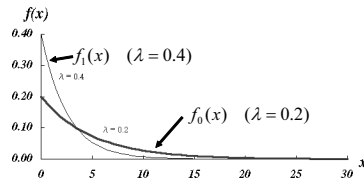


Fig. 2. Exponential distribution

2 Characteristics of Document Streams and Their Relationship to Burst Identification

A document stream is defined as a sequence of documents that arrive in a temporal order as shown in Figure 1. In the case of e-mail, this model suggests that an e-mail d_t is followed by an e-mail d_{t+1} with an inter-arrival gap x_t . Similarly, BBS and blog entries can be considered as document streams.

Kleinberg’s algorithm which identifies bursts in a document stream, assumes x_t changes depending on the number of documents relevant to the appearance of events in document streams. The number of documents relevant to a news event increases just after the event occurs and/or a new related event occurs, while it decreases in the period between these events. A period when the number of documents increases suggests the occurrence of an event, and it is called a burst state (q_1) as opposed to a normal state (q_0).

The state transition from a normal state to a burst state or vice versa can be detected by observing the changes in the arrival interval x_t . If we want to identify the burst of an event in a document stream, we search for shorter inter-arrival gaps in the document streams relating to the event.

Kleinberg’s algorithm was originally applied to e-mail messages. E-mail messages are usually distributed uniformly, however, this is not always the case for other web content such as BBSs and blogs. Figure 3 shows a graph of the number of postings on ‘Ni-channel’, the largest BBS in Japan. This graph shows the elapsed time plotted at 20-minute intervals along the x-axis and the number of postings over five days plotted along the y-axis. As shown in the graph, the average number of postings varies depending on the time of day. Our blog collecting system gathered blogs from 1995 to 2004, including blogs written using blog tools and those written as normal web pages. Their distribution is also non-uniform, as shown in Figure 4. This could be due to the growth of the WWW and blog communities.

In Section 2.1, we illustrate Kleinberg’s algorithm and show how it entails that document arrivals are expected to follow a uniform distribution. Then we describe our proposed algorithm in Section 2.2.

2.1 Kleinberg’s Algorithm

Kleinberg’s algorithm assumes that the sequence of document arrival times follows an exponential distribution. This assumption can be considered as equiv-

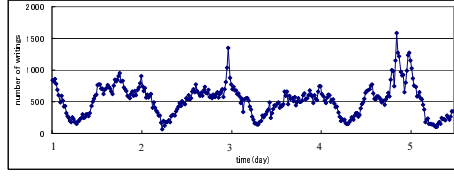


Fig. 3. Distribution of postings on Ni-channel

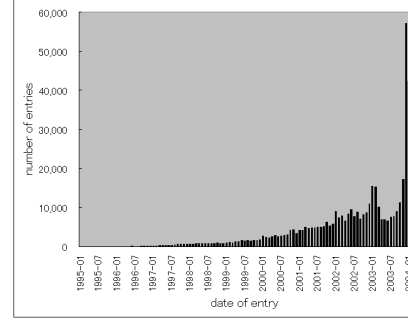


Fig. 4. Distribution of collected blog entries

alent to the assumption that documents arrive randomly. The equivalence can be explained as follows. Suppose that there are K documents within a period T and the arrival of the documents follows a uniform distribution. Then, the probability that a document arrives on time τ within the period $(y, y+x]$ is :

$$P(\tau \in (y, y+x] \subset (0, T]) = x/T \quad (1)$$

for $0 < y < y+x < T$. Below, we denote $A((y, y+x])$ as the number of documents that arrive in the period $(y, y+x]$. Hence, the probability that k documents arrive in the period $(y, y+x]$ is

$$P(A((y, y+x]) = k) = \frac{K!}{k!(K-k)!} \left(\frac{x}{T}\right)^k \left(1 - \frac{x}{T}\right)^{K-k} \quad (2)$$

Here, we let the average number of document arrivals be $\lambda = K/T$. Then, Eqn. (2) can be written as

$$P(A((y, y+x]) = k) = e^{-\lambda x} \frac{(\lambda x)^k}{k!} \quad (3)$$

which shows that the number of documents arriving within the period $(y, y+x]$ follows a Poisson distribution.

Furthermore, we let $Z(t_0)$ be the phenomenon that a document arrives at the time t_0 and X be the time interval to the next document. Then, Eqn. (3) can be written as

$$\begin{aligned} P(X > t \mid Z(t_0)) &= P(A((t_0, t_0+t]) = 0 \mid Z(t_0)) \\ &= P(A((t_0, t_0+t]) = 0) = e^{-\lambda t} \end{aligned} \quad (4)$$

Eqn. (4) means that the cumulative distribution function and the probability density function of document arrival intervals follows the exponential distribution of

$$P(X < t \mid Z(t_0)) = 1 - e^{-\lambda t} \quad (5)$$

$$P(t) = \lambda e^{-\lambda t} \quad (6)$$

The above argument entails that a sequence of document arrival times follows an exponential distribution, which is equivalent to saying that document arrivals follow a uniform distribution.

Figure 2 shows two graphs of exponential distribution functions $f_0(x) = \lambda_0 e^{-\lambda_0 x}$, $f_1(x) = \lambda_1 e^{-\lambda_1 x}$ ($\lambda_0 < \lambda_1$), showing that $f_1(x) > f_0(x)$ for small x and $f_1(x) < f_0(x)$ for large x , where x is the inter-arrival gap. Recall that λ , the parameter for exponential distribution, is $\lambda = K/T$, which represents the average number of document arrivals. We define λ_0 and λ_1 as parameters that correspond to normal and burst states, respectively. When the parameters are thus defined, a document in a time interval x_t can be considered to be in a burst state if $f_0(x_t) < f_1(x_t)$. Conversely, a document in a time interval x_t can be considered to be in a normal state if $f_0(x_t) > f_1(x_t)$.

This algorithm can be used to determine whether a document is in a burst state or a normal state. However, if we determine the state simply using this algorithm, numerous small clusters of bursts will be identified. This is undesirable when we need to treat small clusters as one large cluster. Therefore, the algorithm calculates the costs for all pairs of documents and states using a two-state automaton that expresses the sequence of states. When the costs are calculated, the algorithm selects the sequence of states that has the smallest cost. At this point, a state transition cost is introduced for calculating the cost not to change the states easily.

The sequence of states with the minimum cost can be calculated as follows. Let t be the index number of a document, $\tau(l, j)$ be a state transition cost from state l to state j , and $C_j(t)$ be a cost for document d_t and its state j .

1. For the initial state $t = 0$, define $C_0(t) = 0$, $C_1(t) = \infty$
2. $t = t + 1$
3. Calculate the cost $C_j(t)$ ($j = 0, 1$)
 $C_j(t)$ is defined as

$$C_j(t) = -\ln f_j(x_t) + \min_l (C_l(t-1) + \tau(l, j)) \quad (7)$$

Here, the state of document d_t is described as j , and the state of d_{t-1} is described as l . In addition, the transition cost $\tau(l, j)$ is defined as $\tau(l, j) = \gamma \log n$ for $l < j$, and $\tau(l, j) = 0$ for $l \geq j$.

4. Repeat steps 2 and 3 for all documents
5. Select the sequence of states that has the minimum cost

The minimum cost sequence of states from calculated document-state pairs is determined using Viterbi algorithm. That is, starting with the state j of the document d_{last} with a minimum of $C_{j=0}(last)$ and $C_{j=1}(last)$, it recursively determines the state of the previous document by traversing the state transition from which state the current state comes.

Having determined the state of each document, continuous sequences of burst state documents are treated as bursts. Therefore, the length of a burst is defined as being from the arrival of the first burst document to the arrival of the last burst document.

The weight of burst for each document can be defined by the difference between the costs calculated for documents in burst and normal states. The weight of a burst can be defined using this weight. In this paper, the weight of burst is defined as the sum of the weights of the documents included in the burst.

2.2 Extending Kleinberg’s Method

The burst detection algorithm described in Section 2.1 is a method of identifying a change in the number of documents relating to an event. However, this algorithm assumes that the number of documents in a document stream is stable in the normal state. This assumption holds true for some document streams, such as e-mail messages, but not for all document streams. As we saw in Figures 3 and 4, it does not hold for BBSs or blogs.

As shown in Section 2.1, λ_0 can be interpreted as the average number of documents in each document stream. In other words, λ_0 is equal to the average number of document arrivals in a normal state. This implies that λ_0 is a parameter that can be tuned to reflect the average number of document arrivals in a document stream of any distribution.

In a BBS, changes in the number of postings are cyclic, i.e., they are sparse in the daytime and dense at night time. We modify λ_0 to account for the fact that the number of postings to a BBS differs between daytime and night time. We can determine the value of λ_0 taking into account that there is a cycle and that each period has an average number of documents.

Consider a BBS that has n postings per day. The value of $\lambda_{0,i}$, which means λ_0 for period i , can be defined as follows. Let K_i be the expected number of postings in the period i and T_i be the length of the period.

$$\lambda_{0,i} = \frac{n \times (K_i / \sum K_i)}{T_i} \quad (8)$$

As original $\lambda_0 = K/T$, the numerator of $\lambda_{0,i}$ suggests the expected value of the postings.

Blogs form different document streams, thus, we have different λ_0 . Since there is no periodicity in the number of blog entries, we cannot calculate the average number of blog entries from the past history. Therefore, we assume the number of a subset of blog entries relating to a topic changes similarly to the number of the total set.

When a set of blog entries consists of n entries, $\lambda_{0,i}$ for a period of time, i can be defined as follows:

$$\lambda_{0,i} = \frac{n \times (K_i / \sum K_i)}{T_i} \quad (9)$$

In this equation, the numerator is equal to the expected value of the documents.

Clearly, the length of the time period T_i can be shortened for the cases like a BBS. However, T_i must have some width for blog entries, which cannot refer to past history. T_i must have a minimum width that is not influenced by an increase or decrease in documents.

Table 1. Identified bursts for “Olympic” using Kleinberg’s algorithm

| Interval of burst | # of docs | Weight |
|---------------------------|-----------|--------|
| Feb. 6 '98 – Feb. 25 '98 | 34 | 47.33 |
| Aug. 22 '00 – Oct. 14 '00 | 171 | 121.4 |
| Feb. 6 '02 – Mar. 3 '02 | 137 | 84.90 |
| Jun. 3 '02 – Jun. 26 '02 | 30 | 36.80 |
| Feb. 7 '03 – Mar. 6 '03 | 30 | 39.61 |
| Nov. 1 '04 – Jan. 21 '04 | 114 | 138.0 |

Table 2. Identified bursts for “Olympic” using our algorithm

| Interval of burst | # of docs | Weight |
|---------------------------|-----------|--------|
| Feb. 6 '98 – Feb. 25 '98 | 34 | 70.67 |
| Aug. 22 '00 – Oct. 14 '00 | 171 | 126.1 |
| Feb. 6 '02 – Mar. 2 '02 | 135 | 67.45 |

3 Experiment

We conducted an experiment to identify which words could be considered topic words in a certain time period, to assess whether our algorithm is effective for BBSs and blogs.

3.1 Experiments on Blog Data

We made an experiment on blog data which was collected by our system, which included blogs published from 1 January 1995 to 22 January 2004; 466,809 entries were collected in total. The distribution of blog entries is shown in Figure 4. We observe some irregularity in the distribution of entries, and apply our extended algorithm to detecting bursts in the blog streams. Note that the length of a time period (T_i) is set at one month.

Our first experiment is to examine the burstiness of a word. That is, for a given word w , we investigate if the word had obtained attention in a blog stream as follows:

1. Collect the blog entries that contain the word w
2. Sort these entries into chronological order
3. Apply the algorithm to blog entries
4. Output the resulting entries that are in a burst state

We can know whether an entry is in a burst state or not. Then, gathering the entries that are in a burst state are gathered into a cluster. The cluster is treated as a burst in which an event relating to the word w is occurred during the period. Furthermore, we can output the burst weight for each burst.

Tables 1 and 2 show the results for the word ‘Olympic’ using Kleinberg’s algorithm and our one, respectively. These tables show that Kleinberg’s algorithm misidentified the last three bursts. This indicates that Kleinberg’s algorithm cannot distinguish increases of word frequency from bursts. As shown in Figure 4, the misidentification was caused by the increase in the number of blog entries in recent years, not directly related to the event.

In contrast, every burst extracted by our method (Table 2) corresponded to the Olympics held at Nagano, Sydney, and Salt Lake City. This indicates that our method can identify bursts correctly.

A comparison of the Tables 3 and 4 that are the results for the word ‘Christmas’ shows that there is a dispersion of burst weight in Table 3. This is due

Table 3. Identified bursts for “Christmas” using Kleinberg’s algorithm

| Interval of burst | # of docs | Weight |
|---------------------------|-----------|--------|
| Dec. 23 '98 – Dec. 26 '98 | 34 | 9.724 |
| Dec. 19 '99 – Dec. 27 '99 | 78 | 19.32 |
| Dec. 10 '00 – Dec. 28 '00 | 176 | 39.84 |
| Nov. 25 '01 – Jan. 2 '02 | 471 | 86.94 |
| Nov. 13 '02 – Jan. 8 '03 | 833 | 137.3 |
| Nov. 5 '03 – Jan. 17 '04 | 3193 | 201.3 |

Table 5. Topic word list for Nov. 2003 using Kleinberg’s algorithm

| Word | Interval of burst | # of docs | Weight |
|-----------|-------------------|-----------|---------|
| comment | Nov. 1 – Nov. 30 | 2752 | 12196.6 |
| track | Nov. 1 – Nov. 30 | 1946 | 10947.1 |
| back | Nov. 1 – Nov. 30 | 2098 | 10900.0 |
| posted | Nov. 1 – Nov. 30 | 1347 | 7247.4 |
| vote | Nov. 1 – Nov. 30 | 444 | 3269.8 |
| election | Nov. 1 – Nov. 30 | 324 | 3026.3 |
| reviewer | Nov. 1 – Nov. 30 | 283 | 2997.7 |
| reference | Nov. 1 – Nov. 30 | 602 | 2183.2 |
| review | Nov. 1 – Nov. 30 | 379 | 2029.6 |
| profile | Nov. 2 – Nov. 30 | 216 | 1541.9 |

Table 4. Identified bursts for “Christmas” using our algorithm

| Interval of burst | # of docs | Weight |
|---------------------------|-----------|--------|
| Dec. 18 '97 – Dec. 30 '97 | 24 | 23.73 |
| Dec. 11 '98 – Dec. 27 '98 | 53 | 38.28 |
| Dec. 10 '99 – Dec. 31 '99 | 107 | 47.28 |
| Dec. 10 '00 – Dec. 28 '00 | 176 | 41.27 |
| Dec. 1 '01 – Dec. 29 '01 | 428 | 59.93 |
| Nov. 30 '02 – Dec. 27 '02 | 670 | 58.07 |
| Dec. 9 '03 – Dec. 29 '03 | 2621 | 49.30 |

Table 6. Topic word list for Nov. 2003 using our algorithm

| Word | Interval of burst | # of docs | Weight |
|-----------------------------|-------------------|-----------|--------|
| election | Nov. 7 – Nov. 12 | 187 | 1361.3 |
| reviewer | Nov. 6 – Nov. 30 | 260 | 954.56 |
| vote | Nov. 7 – Nov. 11 | 162 | 794.46 |
| manifesto | Nov. 1 – Nov. 13 | 49 | 423.38 |
| Kokubo (baseball player) | Nov. 3 – Nov. 16 | 39 | 402.98 |
| matrix revolutions | Nov. 3 – Nov. 27 | 51 | 402.92 |
| political party | Nov. 7 – Nov. 14 | 49 | 387.60 |
| election system | Nov. 9 – Nov. 11 | 39 | 364.91 |
| House of Representatives | Nov. 1 – Nov. 10 | 44 | 301.00 |
| lose an election | Nov. 9 – Nov. 14 | 38 | 298.88 |

to the influence of an increasing number of entries. Thus, our algorithm works better than Kleinberg’s algorithm in these examples.

The aim of the second experiment is to list topic words that indicate what events occurred in the document stream. The list shows ranking of words that attracted attention in each month. A collection of single words (uni-gram) is not informative enough to infer an event that the document stream indicates. N-gram phrases, which include the target word and its surrounding nouns are much more informative. Therefore, we made a list using the following steps:

1. Collect blog entries that include word w and calculate whether it is a burst
2. If there is a burst, memorize word w and its period of burst
3. Extract all n-grams including w from entries that relate to the burst
4. Memorize n-grams that are sequences consisting of nouns and the target word w
5. Process all words
6. Sort by weight for each month and output the result

Tables 5 and 6 show the top 10 of the topic word rankings on Nov. 2003, generated by the Kleinberg’s algorithm and our algorithm, respectively. We omitted extracted n-grams for reasons of space.

3.2 Experiments on BBS

We conducted a similar experiment on a BBS. We used ‘Ni-channel’[7], the largest BBS in Japan, and collected 41,4806 postings from Dec.3 2003 to Jan.3 2004. There are usually more postings to this BBS during the day and fewer at night, as shown in Figure 3. Therefore, we monitored the postings to the BBS over Sep. 2003, and calculated them as an average value.

We also generated a list of topic words from the BBS. However, they are not included here since many of them were jargon that cannot be readily translated into English.

4 Evaluation

We conducted evaluation experiments to assess the effectiveness of our algorithm by evaluating a topic word list manually.

Five months, Feb. 1998(Nagano Olympics), Sep. 2000(Sydney Olympics), Jun. 2002(Football World Cup), Nov. 2003(Japanese elections), and Dec. 2003, were selected for evaluation and topic word lists were generated using the following four algorithms.

1. Kleinberg’s algorithm($s = 4, \gamma = 2$)
2. Our algorithm($s = 4, \gamma = 2$)
3. Our algorithm($s = 2, \gamma = 1$)

Kleinberg’s algorithm and our proposed algorithm have two parameters, s and γ , which have the default values $s = 2$ and $\gamma = 1$. Since we defined the weight of a burst as the sum of the weights of the documents, the weight of a burst consisting of a large number of documents tends to be higher. Therefore, we conduct experiments with alternate values for the parameters that produced good results in preliminary experiments.

4. TFIDF

We define the weight of word w as the sum of tfidf values for each entry in each month.

We made word lists for each month by extracting the top 10 words from the topic words lists generated by the four algorithms. Each word list is independently evaluated by two subjects to see whether it is appropriate for the topic word of that month.

The result of this evaluation is shown in Table 7. The numbers shown in the table indicate how many words are judged appropriate for the topic word. The numbers outside the brackets are the number of words judged appropriate by at least one subject, and the numbers in brackets are the number of words that the two subjects judged to be appropriate. Note that Kleinberg’s algorithm extracted only two topic words for Feb. 1998. This table shows that algorithm 2 gives the best results for each month.

Table 8 also shows the number of topic words extracted by each algorithm. Algorithm 4 calculates the weight for all words. Therefore, its maximum number

Table 7. Number of correct topic words within top 10 words: blog

| years | Feb.'98 | Sep.'00 | Jun.'02 | Nov.'03 | Dec.'03 | Average |
|---|---------|---------|---------|---------|---------|----------|
| Kleinberg's algorithm ($s = 4, \gamma = 2$) | 1(0) | 7(6) | 10(10) | 4(2) | 2(0) | 4.8(3.6) |
| Our algorithm ($s = 4, \gamma = 2$) | 3(2) | 7(6) | 10(10) | 9(9) | 7(7) | 7.2(6.8) |
| Our algorithm ($s = 2, \gamma = 1$) | 1(1) | 5(4) | 10(10) | 4(2) | 2(1) | 4.4(3.8) |
| TFIDF | 0(0) | 1(1) | 0(0) | 3(1) | 1(0) | 1.0(0.4) |

Table 8. Number of extracted topic words: blog

| years | Feb.'98 | Sep.'00 | Jun.'02 | Nov.'03 | Dec.'03 | Average |
|---|---------|---------|---------|---------|---------|---------|
| Kleinberg's algorithm ($s = 4, \gamma = 2$) | 2 | 25 | 260 | 7912 | 14881 | 23080 |
| Our algorithm ($s = 4, \gamma = 2$) | 23 | 27 | 80 | 36 | 245 | 411 |
| TFIDF | 756 | 1745 | 3595 | 4823 | 12999 | 23918 |

of topic words is equivalent to the total number of words. As Table 8 shows, Kleinberg's algorithm did not extract enough topic words for 1998 and identified too many for 2003 since the distribution of entries is not uniform. In contrast, the number of topic words extracted by our method is stable.

We conduct a similar evaluation experiment for a BBS. Table 9 shows the results. Here, we used the following three algorithms:

1. Kleinberg's algorithm ($s = 4, \gamma = 2$)
2. Our algorithm ($s = 4, \gamma = 2$)
3. TFIDF

The weight of a word is defined as the sum of the TFIDF values for the words in all postings.

The top 20 topic words are extracted by each algorithm and evaluated by two subjects, as in the blog experiment. However, there is no significant difference between algorithms 1 and 2.

Table 10 shows the abilities of algorithms 1 and 2 to extract the topic words for each period. While algorithm 1 did not extract potential topic words from daytime postings, algorithm 2 did.

5 Conclusion

In this paper, we proposed an extension of the burst identification algorithm originally proposed by Kleinberg and conducted experiments on extracting topic words from BBSs and blogs.

Our proposed algorithm produced better results than Kleinberg's algorithm or TFIDF for document streams with non-uniform distribution. However, our blog collecting system currently collects duplicated blog entries. It can be collected via mirror sites and causes some misidentifications in both algorithms by appearing multiple times in the document stream.

Our algorithm assumes that the length of a time period is fixed, which is equivalent to assuming that the documents are distributed uniformly in each

Table 9. Number of correct topic words within top 20 words: BBS

| | |
|--|--------|
| Kleinberg's algorithm ($s = 4, \gamma = 2$) | 18(17) |
| Our algorithm ($s = 4, \gamma = 2$) | 19(19) |
| TFIDF | 1(1) |

Table 10. Number of extracted topic words: BBS

| Hour | 0 - 6 | 6 - 12 | 12 - 18 | 18 - 24 | Total |
|-----------------------|-------|--------|---------|---------|-------|
| Kleinberg's algorithm | 13 | 4 | 26 | 29 | 72 |
| Our algorithm | 8 | 3 | 94 | 27 | 132 |

period. In our experiments, we tentatively set the length of a period at one month, but this may not be appropriate for every document stream as we discussed. In future, we plan to develop a method for calculating suitable time periods for any given document stream.

Our current proposal extracts surrounding n-grams to help understand what events the topic words corresponds to. Since not all the extracted n-grams correspond to events, we are developing a method to determine which n-grams are most appropriate.

Though current research is focused on topic word extraction, it would be interesting to apply our algorithm to the problem of detecting bursts in a hyperlinked space of blog communities as in [4].

6 Acknowledgments

This work was partly supported by the Exploratory Software Project 2003, Information-technology Promotion Agency, Japan (IPA).

It was also supported by The 21st Century COE Program "Framework for Systematization and Application of Large-scale Knowledge Resources", Japan Society for the Promotion of Science.

We thank Kaoru YAMAMOTO for her help in completing this paper.

References

1. J. Wakefield. Catching a buzz. *Scientific American*, Nov. 2001.
2. Y. Matsuo, Y. Ohsawa, and M. Ishizuka. Mining messages in an electronic message board by repetition of words. The Second International Workshop on Chance Discovery, Pacific Rim International Conference on AI, Aug. 2002.
3. J. Kleinberg. Bursty and hierarchical structure in streams. In *Proc. the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul. 2002.
4. R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *Proc. of the 12th International World Wide Web Conference*, May 2003.
5. K. Mane and K. Borner. Mapping topics and topic bursts in PNAS. In *Proc. National Academy of Sciences*, Feb. 2004.
6. T. Nanno, T. Fujiki, Y. Suzuki, and M. Okumura. Automatically collecting, monitoring, and mining japanese weblogs. 13th International World Wide Web Conference, May 2004.
7. Ni-channel, <http://www.2ch.net/>.