Query-Guided Event Detection from News and Blog Streams[*]

Aixin Sun and Meishan Hu

Nanyang Technological University, Nanyang Avenue, Singapore 639798
axsun@ntu.edu.sg

Abstract

With the advent of Web 2.0, searching and publishing become two major forms of online activities for web users. When an event happens, web users would search for the latest information about the event as well as publish blog posts to discuss the event. Both the queries from users and the blog posts published give strong indications of the real-world events of users' concern. In this paper, we propose to study *query-guided event detection* from two parallel document streams (i.e., news and blog). Our goal is to group user queries, news articles, and blog posts into events to which they are related. The evolution of an event is reflected by the changes in both the query keywords and news/blogs content during the event happening period. We propose a two-stage real-time event detection framework consisting of *event fragment detection* and *event detection*. The proposed framework integrates queries, news articles, and blog posts through the notion of *query profile*. In our experiments, we evaluate the proposed framework using real-world data collected from Technorati and Google News.

1 Introduction

Publishing offered by various Web 2.0 applications has transformed web users from passive information consumers to active information providers. The user-generated content including queries, blog posts, and comments, has become the ideal source for understanding web users' information needs and their interests.

1.1 Motivation

The recently released Google Flu Trends[1] showcases an important application on estimating flu epidemics based on queries received from massive web users [8]. It is observed that more flu-related searches are received by search engines during flu season. This observation is partially consistent with the claim that when a bursty event happens, the number of event-related queries increases dramatically [11, 16, 20]. Here, a bursty event refers to an event that emerged into a burst from the background of non-event, for example, `the launch of iPhone` and the `Assassination of Benazir Bhutto`[2]. Figure 1 shows the popular queries[3] published by Technorati on 28 Dec 07, one day after the `Assassination of Benazir Bhutto`. The top 4 most popular queries are all related to that event. A sharp increment in the number of blog posts mentioning *Benazir Bhutto* right after the event's happening was also observed, as shown in Figure 1. This reflects that, when an event happens, many users would try to locate the most up-to-date information about the event through search engines. It also leads to the following:

- A large volume of event-related queries are issued to news/blog search engines, making them *popular queries* during the event period.
- A large number of news articles and blog posts are published by journalists and bloggers containing updated facts, commentary or discussions about the event.
- From the updated information, web users may formulate new queries (e.g., another person involved in the event) which may subsequently become popular queries. The changes in the queries at different time points become good indications of event evolution.

The above also summarizes the interactions between *query streams*, *news streams*, and *blog streams*. The availability of popular queries[4] and blog posts therefore offers us an opportunity to better understand what web users *want to know about* and what they *talk about*. By taking input from queries, news, and blogs, we believe events can be detected in a user-oriented manner.

In this paper, we study *query-guided event detection*. As defined in [1], an event means something unique happens at some point of time, which can be expected like New Year's Eve or unexpected like the recent Chile Earthquake. Most existing approaches in event detection are to identify events from a single stream of documents (mostly news articles), and to group the documents according to the events they are related to. However, we argue that not all events reported in news articles are of interest to common users. Based on the above discussion, we propose to detect events based on users' queries and their matching news articles and blog

[*]This work was supported by A*STAR Public Sector R&D, Singapore. Project Number 062 101 0031.

[1]`http://www.google.org/flutrends/`.

[2]`http://en.wikipedia.org/wiki/Assassination_of_Benazir_Bhutto`

[3]`http://www.technorati.com/pop/`

[4]Popular/hot search keywords are accessible from websites listed in Yahoo! directory `Popular_Search_Words`, including Google Trends and Technorati.

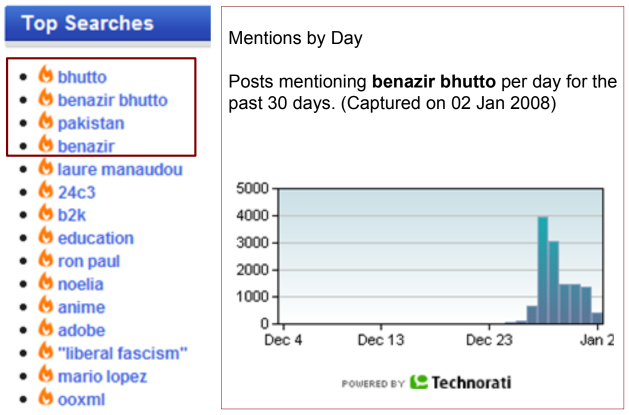


Figure 1: Top-15 popular searches from Technorati.com captured on 28 Dec 2007 and statistics on blog posts captured on 02 Jan 2008.

posts. As user queries represent users' information needs at the time of issuing (hence representing user interests), query-guided event detection could detect the events that are of interest to common web users. Moreover, the evolution of a detected event is well represented by the change in query keywords related to the event [17].

1.2 Research Challenges and Contributions

Query-guided event detection is a challenging task. First, not all popular queries issued by masses of web users are event-related. The observation that event-related queries increase dramatically when an event happens does not necessarily imply that all popular queries are event-related. Many extremely popular queries are likely to be website names, such as *Google*, *MySpace* and *YouTube*, and they are often not event-related [15]. Second, multiple query keywords may be related to the same event [14]. For instance, all the top 4 query keywords shown in Figure 1 refer to the same event. Nevertheless, the same query keyword *Pakistan* issued at different time points may refer to different events happened in that country. Third, given the large number of news articles and blog posts accessible online, *minimizing the computational cost* is always a demanding requirement.

In addressing these challenges, we have made the following contributions.

1. We propose the notion of *query profile* and define two measures, namely, *clarity* and *recency*, to characterize query profiles in Section 3. Put simply, the profile of a query q at time point t is the set of most recently published documents from news (or blog) stream that match q. The clarity measure quantifies the difference on the word distribution in a query profile against the expected word distribution. Recency refers to the time difference between the documents in the query profile and the time of issuing the query. A query is likely related to an event if its clarity is large and recency is small.

2. We propose a two-stage framework for real-time event detection in Section 4. In the first stage, query profiles from the same stream (either news or blog) that are related to the same event are grouped into an *event fragment*. In the second stage, event fragments from both streams are clustered into events.

3. We conducted experiments using real-world data collected from Technorati and Google News spanning over 21 months. The experimental results and observations are reported in Section 5.

2 Related Work

Event detection is often modeled as a real-time incremental clustering task [1, 2, 18, 19]. For each received document, its similarity to the prior events is computed. The document is assigned to either the most similar prior event or a new event. A comprehensive study on the efficiency of event detection using real-time clustering approach is reported in [13]. Another approach for event detection is to identify bursty features from a document stream [9, 12]. Features sharing similar bursty patterns in the similar time period are grouped together as an event. The detected event is therefore described by a set of features, not a cluster of documents.

Event detection based on prior user queries is reported in [7, 10]. Fung *et al.* [7] proposed to first identify the bursty features related to the user query and then organize the documents related to those bursty features into an event hierarchy. In [10], a user specifies an event (or a topic) of interest using several keywords as a query. The response to the query is a combination of streams (e.g., news feeds, emails) that are sufficiently correlated and collectively contain all query keywords within a time period. In our problem setting, not all queries are event-related. Further, queries are not predefined from a single user but accumulatively received from the whole population of web users. Our work is also related to event detection using click-through data [20]. Although both aim to detect events guided by users' queries, our work differs in two major aspects: (i) the proposed method in [20] is for retrospective event detection, while ours is for real-time event detection; and (ii) we utilize documents returned by search engines but not click-through data as in [20].

Event ranking with user attention is reported in [16] where the events are firstly detected from news streams. User attention is then derived from the number of pageviews (collected through web browser toolbars) for all the news articles in the same event. In our approach, we detect events from the two parallel streams (i.e., news and blogs) simultaneously and we show that some events emerge from blog stream earlier than news. Moreover our event detection process is guided by user queries while no query was used in [16].

3 Query Profile

Let $\mathcal{S}$ be a document stream where each document $d \in \mathcal{S}$ is associated with its timestamp $d.t$. Let q be a query with query keyword $q.w$ received at time $q.t$. Note that, a query is defined by *query keyword* and *timestamp*. The same keyword issued at multiple time points hence leads to multiple queries.

The **query profile** of query q against document stream $\mathcal{S}$, denoted by $C_q^{\mathcal{S}}$, is a set of N most recent documents from $\mathcal{S}$ matching q that are created before $q.t$. N is a predefined constant defining the maximum size of the query profile, i.e., $|C_q^{\mathcal{S}}| \leq N$, and $N = 50$ throughout this paper. The timestamp of a query profile is the same as its query's timestamp, or $C_q.t = q.t$. In our following discussions, we may simply use C_q to denote the query profile of q for brevity when $\mathcal{S}$ is clear in the context.

In this work, we assume that the news and blog streams contain a sufficient number of documents received from various sources. For instance, we may consider that the news stream contains all news articles accessible through Google News or Yahoo! News. With the availability of search engines dedicated to news/blogs, the construction of a query profile for a given query can be easily reduced to the task of retrieving the top-N matching documents from the corresponding search engines as recency is always a key ranking factor in news/blog search engines. Nevertheless, the query profiles constructed will then depend on the proprietary ranking functions of the search engines. In the following, we describe the recency and clarity measures in detail.

3.1 Recency

Query profile *recency* is the averaged time difference between documents in the query profile to the query's issuing time (see Equation 1).

$$recency(C_q) = \frac{1}{|C_q|} \sum_{d \in C_q} (q.t - d.t) \tag{1}$$

Given the large number of news sources/blogs, even if each source reports the event in one article, the number of articles matching q could be very large. Moreover, all the matching articles are published within a short period of the event's happening given the nature of news/blogs. Consequently, a user query q at time $q.t$ is likely to be event-related if $recency(C_q)$ is small.

3.2 Clarity

Assume an event E is described by a set of words $E = \{w_1, w_2, \cdots, w_n\}$, e.g., person and location names involved in the event. If a query is event-related, its word features describing the event can often distinguish it from the background. In other words, the word distribution of the query profile and that of a general document collection is expected to be diverge, measured through across distance metrics such as Kullback-Leibler (KL) divergence. Based on [4], we define *clarity* of a query profile by the KL-divergence between the query profile language model and the background language model (i.e., all documents in the stream) (see Equation 2).

$$clarity(C_q) = KL(\hat{C}_q \| \hat{\mathcal{S}}) = \sum_{w \in C_q} P(w|C_q) \log_2 \frac{P(w|C_q)}{P(w|\mathcal{S})} \tag{2}$$

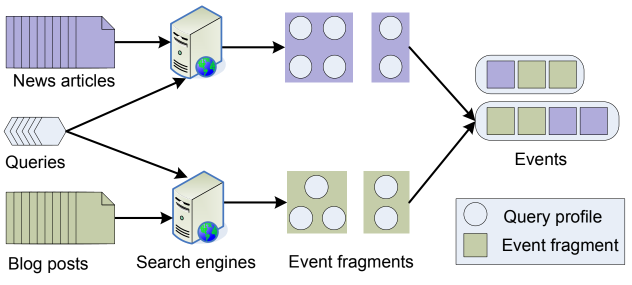


Figure 2: Overview of the proposed framework.

$$P(w|C_q) = \frac{df(w, C_q)}{\sum_{w' \in V} df(w', C_q)} \tag{3}$$

However, in our problem setting, the event is unknown hence the set of words describing it. Assuming q is about a particular event, then the set of words that are frequently observed among documents in C_q are likely to be describing the event. For instance, if we expect event-related words to appear in at least 10% of the documents describing the event, then those words with $df(w, C_q)/|C_q| \geq 0.1$ are likely to be event-related and should be included in computing $clarity(C_q)$, where $df(w, C_q)$ is the document frequency of word w in document set C_q. As described in Equation 3, we estimate $P(w|C_q)$ using the relative document frequency of w in C_q, where V denotes the vocabulary. Analogously, $P(w|\mathcal{S})$ is computed using Equation 3 by replacing C_q with $\mathcal{S}$.

A query profile C_q will have a high clarity score if and only if (i) there is a large number of words having high document frequencies in C_q, and (ii) the probability distribution of these words are very different from that in the whole document collection $\mathcal{S}$. Without holding both conditions, a query profile will have a low clarity score.

Based on the recency and clarity measures, a query is predicted to be event-related if and only if $recency(C_q) \geq \theta_r$ and $clarity(C_q) \geq \theta_c$.

4 Event detection

4.1 Framework Overview

The proposed query-guided event detection framework is shown in Figure 2. The two stages are *event fragment detection* and *event detection*. We illustrate the two stages in the sequel.

For each query received from the query stream, two query profiles are constructed from the news and blog streams respectively, with each utilizing its dedicated search engine. The event-related query profiles are further passed to event fragment detection. The non-event-related query profiles are dropped. An event fragment is a set of query profiles that is about the same event received from the same document stream within a predefined time window T. As documents from different streams may demonstrate different properties (e.g., blog posts are noisier than news articles in general), event fragment detection may require different parameter settings for different document streams. More importantly, some events may emerge earlier in one stream than another.

In event detection, event fragments from both document streams are grouped into events. Hence, an event is a sequence of event fragments from both news and blog streams. Recall that an event fragment contains query profiles that each contains documents matching a query. A detected event contains queries and news articles/blog posts matching them.

One advantage of the proposed approach is that not all documents in the two document streams are processed. Only those documents matching the event-related queries are further processed for event fragment detection and event detection. This significantly reduces the computational cost. Another advantage is that the event detection process is guided by user queries. Hence the events detected are likely to be of interest to common web users with the price of missing some events reported in news/blogs but not heavily searched by users.

4.2 Event Fragment Detection

Given the query profiles received from a particular document stream within the last time window T, the event fragment detection task is to group the query profiles related to the same event into one event fragment. To perform the grouping, (i) an appropriate distance metric between any pair of query profiles, and (ii) an appropriate clustering algorithm, are required.

There are many ways to compute the distance (or similarity) between two sets of documents. One is to compute the cosine similarity between document features (e.g., words weighted by $tf \times idf$ scheme). Another commonly used method is to compute the divergence between the language models of the two (sets of) documents [3]. We used the square root of Jensen-Shannon divergence $\sqrt{JSD}$ as the metric since the language model of each query profile is available from earlier processing (see Equation 3). JSD between $\hat{C_q}$ and $\hat{C_{q'}}$ is defined in Equation 4 where $M = (\hat{C_q} + \hat{C_{q'}})/2$.

$$JSD(C_q, C_{q'}) = \frac{KL(\hat{C_q}||M) + KL(\hat{C_{q'}}||M)}{2} \qquad (4)$$

In event fragment detection, we need a clustering algorithm satisfying the following requirements: (i) the algorithm requires no prior knowledge on the number of event fragments to be detected from a set of query profiles, as it is unreasonable to guess how many events would happen in a given time window, (ii) the algorithm should be able to filter away noise, as predicting whether a query is event-related can never be perfect, and (iii) the algorithm should be able to handle large dataset with reasonable space and time complexity. DBScan [6] is a density-based clustering algorithm that satisfies the above requirements.

Before starting, DBScan requires two pre-specified parameters: a distance threshold Eps, and the minimum number of points $MinPts$, in an Eps-neighborhood. Starting from an arbitrary point p, if p's Eps-neighborhood has at least $MinPts$ number of points, a new cluster containing p and its Eps-neighborhood is formed; If a point q belongs to this cluster, then q's Eps-neighborhood are added to this cluster. The process continues until no more points can be added to this cluster. Then an unvisited point is picked and the process repeats. Finally, the points that cannot be added to any cluster are noise.

4.3 Event Detection

The event fragments detected from the two streams are further grouped into events as an event may last for several days. The incremental DBScan [5] was adopted in this procedure, which has been proven to yield the same results as DBScan. That is, the clustering results is independent of the order of event fragments received from the two streams. We then need to define a distance measure between two event fragments. The following three components are considered in measuring the distance.

- *Semantic distance*. As event fragment is a set of query profiles and each is a set of documents, a language model similar to that of query profile is estimated for an event fragment from all its documents. Similarly, the semantic distance between two event fragments is measured by $\sqrt{JSD}$ between their language models.
- *Query distance*. We observed that semantic distance worked well for event fragments received from the same stream but not across streams, probably due to the differences in vocabulary and writing style between news and blogs. We therefore make an assumption that event fragments received within a short time period are likely related to the same event if they share common query keywords. Let Q_g be the set of unique query keywords obtained from the query profiles in event fragment G. The query distance between two event fragments G and G' is defined in Equation 5 where $\alpha \in [0, 1]$ is a parameter adjusting the contribution of query distance in measuring the distance between two event fragments. For a given α, the query distance is in the range of $[1 - \alpha, 1]$.

$$Dist_q(G, G') = 1 - \alpha \frac{|Q_g \cap Q'_g|}{|Q_g \cup Q'_g|} \qquad (5)$$

- *Temporal distance*. As an event may last for a long time period and evolve at a fast pace, event fragments of the same event but are temporally far apart may not be similar to each other. We therefore only compute the distance between a newly detected event fragment to those recently detected within 5 days. The timestamp of the event fragment is derived from its query profiles.

To summarize, the distance between two event fragments G and G' is defined in Equation 6 if their temporal difference is within 5 days. Otherwise $Dist(G, G') = 1$.

$$Dist(G, G') = \sqrt{JSD(G, G')} \times Dist_q(G, G') \qquad (6)$$

If an event does not receive any new event fragment in previous 5 days, the event is archived. This significantly reduces the number of events to be kept during computation.

5 Experiments

5.1 Data Collection

We collected the 15 most popular queries published by Technorati every 3 hours from 2006-11-08 to 2008-07-31 (21 months). Each popular query was submitted to Technorati (TR for short) and Google News (GN for short) separately right after it was obtained throughout the data collection. The first 50 results returned by each search engine were recorded for constructing query profiles[5]. The title and snippet of each search result is extracted as the document content. In total 73,241 queries were collected with 2,954 unique query keywords after punctuation removal and case standardization. Many query keywords appeared once or twice through the data collection period while a small number of query keywords appeared over one thousand times. In particular, the most popular query keyword *youtube* appeared 3,836 times.

Two undergraduate students were hired to label the queries based the blog posts and news articles retrieved for each query from TR and GN respectively. A query was first labeled as event-related or non-event-related based on the 50 blog posts (resp. news articles) returned for that query. The event-related queries were then further grouped into their related events. The students were asked to write a short description for each observed event. One example event description is "Jason Calacanis Resigns from AOL" reported between 2006-11-17 to 2006-11-29 with query keyword *calacanis*. In total, 514 events were obtained from the collected data where each event lasted for at least one day. As an event maybe only reported in the news stream but not in the blog steam, one query maybe labeled as event-related according to its retrieved news articles, but non-event-related according to its blog posts. Finally, 16,652 queries from the blog stream and 17,160 queries from the news stream were labeled as event-related.

5.2 Experiment Settings

In our experiments, queries with query profiles containing fewer than 50 documents were ignored as they were unlikely to be event-related due to the small number of hits from search engines. As a result, 54,836 query profiles from GN and 68,057 from TR respectively were left for further processing. Recall in Section 3, we propose recency and clarity measures to predict whether a query profile is event-related. In our experiments, the recency threshold θ_r was set to 24 hours and the setting of clarity threshold θ_c was based on [4] where 80% of randomly generated queries were assumed to have low clarity scores. In our dataset, among the top-5 most searched query keywords, three are website names (i.e., *youtube*, *myspace*, and *google*) which are often not related to events. Let μ_c and σ_c be the mean and standard deviation of the clarity scores of the 7,775 query profiles involving these three query keywords. We set $\theta_c = \mu_c + \sigma_c$ and θ_c=3.87 for TR, and 5.88 for GN. With the above parameter settings, 41,261 query profiles from TR and 22,764 query profiles from GN were predicted as event-related for event detection.

For event fragment detection and event detection, we set the following parameters for DBScan. $MinPts$ was 3 for fragment detection, and 4 for event detection; Eps for fragment detection from TR was 0.65, Eps for fragment detection from GN was 0.75, and Eps for event detection was set to 0.55 and $\alpha = 0.35$. With the above settings, we evaluated five methods, namely QP, EFD-3, EFD-6, EFD-12, and EFD-24. These five methods are to evaluate the impact of performing event fragment detection (EFD) in the proposed event detection framework (see Figure 2). With QP method, each query profile is considered one event fragment. With EFD-T method, event fragments are detected from the query profiles received in every time window T (see Section 4.2), and T was set to 3, 6, 12, and 24 hours. We chose to evaluate the event fragment detection for two reasons. First, the proposed event fragment detection distinguishes our approach from other works in event detection. Second, the time window T is a parameter affecting the responsiveness of the proposed event detection system in real-time setting. For instance, if T=24 hours, then the query profiles have to be collected for 24 hours before event fragments can be detected from them. Then the detected events will be presented to the end users of the system with 24 hours of delay in the worst case.

5.3 Performance Evaluation

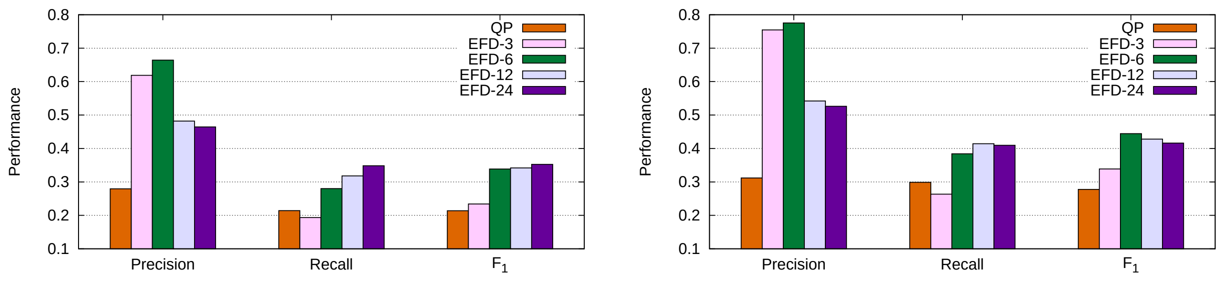
We propose to use a modified version of precision/recall metric. For each detected event E_d, we locate for its reference true event E_t from the manually labeled events, which is the one with the largest number of matching query profiles with E_d. Note that, a true event may match more than one detected event.

Query-based metric. For each pair of detected event and its reference true event $\langle E_d, E_t \rangle$, the precision (resp. recall) of the detection is defined to be the ratio of the number of matching query profiles to the number of query profiles in E_d (resp. E_t). Precision is 0 for a detected event E_d which cannot match any reference true event. Similarly, recall is 0 for a reference true event cannot match any detected event.

Day-based metric. With this metric, instead of comparing the number of matched query profiles in the detected and reference events. We partition of the query profiles in an event according to the calender day. A day match is defined to be at least one query profile match in that day between the event pairs. The precision and recall is then computed based on the ratio of day matches. This metric reflects the fact that most end-user do not need to read multiple query profiles for the same event within a short time period. For instance, in Google News, a few thousands news articles are often found reporting the same piece of news and an end-user needs to read just one of them to get the information.

Figure 3(a) and 3(b) reports the macro-averaged precision, recall and F_1 measures respectively for the five methods. Observe the significant increase in precision with EFD-T methods compared to QP especially with T=3 and T=6 hours. However, EFD-3 led to worse recall compared

[5]Only top 50 results were recorded due to API constraints.



(a) Query-based metric (b) Day-based metric

Figure 3: Macro-averaged precision, recall and F_1 for the 5 methods.

to QP. Among the five methods, EFD-6 was the best performing method according to day-based F_1 and slightly worse than EFD-12 and EFD-24 on query-based F_1. In general, the experiment results showed that the proposed event detection could achieve very good precision and fairly good recall. The low recall could be attributed to the fact that a long lasting event may be detected as a few short events.

5.4 Case Study

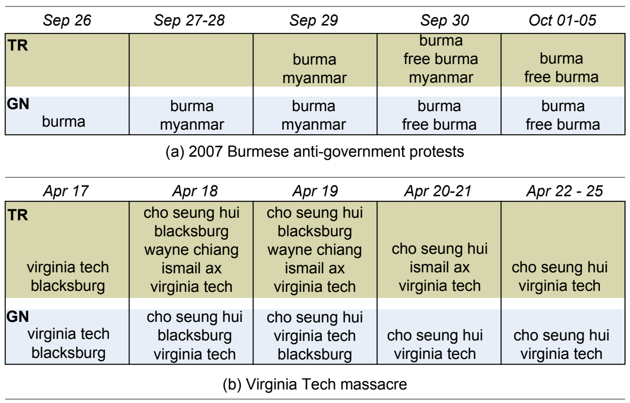
In this section, we report two case studies with different event evolution patterns.

Burmese anti-government protests. This event was first reported in the news for some days before it attracted attention from bloggers. Shown in Figure 4(a), event fragments with keywords *burma* and *myanmar* were detected from 2007.09.26 to 09.28 from the news stream only. There was no heavy discussion from the blog stream until 09.29. Then on 09.30, a new keyword *free burma* was formulated and event fragments were detected from both news and blogs. In fact, all bloggers and website owners worldwide were called upon to support the "free Burma" campaign according to the Wikipedia entry on the event.

Virginia Tech massacre. This was a major event which happened on 2007.04.16. The event fragments from both GN and TR were detected on the second day with keywords *virginia tech* and *blacksburg*, shown in Figure 4(c). This shows that the query profiles could define the semantics of the query keywords using the matched documents at the time of search since both keywords could be related to different events if searched at other time points. From 04.18 to 04.25, we noticed that the fragments detected from GN were mainly *virginia tech* and *cho seung hui*. However, event fragments with keywords *wayne chiang* and *ismail ax* were detected from TR. Compared to news, bloggers have much more freedom in publishing with little or even no verification on the information, making the rumor about Wayne Chiang spread far among bloggers. The words *ismail ax* also attracted much attention among bloggers trying to understand the meaning behind it.

6 Conclusion and Discussion

In this paper, we studied query-guided event detection, taking input from user queries and blogs in addition to news.



	Sep 26	*Sep 27-28*	*Sep 29*	*Sep 30*	*Oct 01-05*
TR			burma myanmar	burma free burma myanmar	burma free burma
GN	burma	burma myanmar	burma myanmar	burma free burma	burma free burma

(a) 2007 Burmese anti-government protests

	Apr 17	*Apr 18*	*Apr 19*	*Apr 20-21*	*Apr 22 - 25*
TR	virginia tech blacksburg	cho seung hui blacksburg wayne chiang ismail ax virginia tech	cho seung hui blacksburg wayne chiang ismail ax virginia tech	cho seung hui ismail ax virginia tech	cho seung hui virginia tech
GN	virginia tech blacksburg	cho seung hui blacksburg virginia tech	cho seung hui virginia tech blacksburg	cho seung hui virginia tech	cho seung hui virginia tech

(b) Virginia Tech massacre

Figure 4: Case study: evolution of two events.

With the notion of query profile, the key challenge of predicting whether a given query is event-related is addressed by two measures, recency and clarity, defined from the temporal and topical aspects respectively. With the proposed two-stage event detection framework, query profiles are firstly grouped into event fragments and then into events. DBScan algorithm was adopted in both stages. Experiments were conducted using real data collected from Google News and Technorati in a period of 21 months. Though promising, this work has one major limitation. Recall that the popular queries were collected from Technorati and the matched documents were collected from Technorati and Google News respectively. As both service providers have their proprietary algorithms for ranking popular queries as well as ranking documents matching these queries, the experimental results were implicitly affected by their ranking algorithms. On the other hand, data collection through dedicated search engines makes it possible to reach more documents without the cost of crawling, storing, parsing, and indexing the huge volume of news articles and blog posts.

References

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *Proc. of DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[2] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proc. of SIGIR*, pages 37–45, Australia, 1998.

[3] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In *Proc. of SIGIR*, pages 390–397, Seattle, USA, 2006.

[4] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proc. of SIGIR*, pages 299–306, Finland, 2002.

[5] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *Proc. of VLDB*, pages 323–333, New York, USA, 1998.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of KDD*, pages 226–231, Portland, USA, 1996.

[7] G. P. C. Fung, J. X. Yu, H. Liu, and P. S. Yu. Time-dependent event hierarchy construction. In *Proc. of KDD*, pages 300–309, 2007.

[8] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 2008. `http://dx.doi.org/10.1038/nature07634`.

[9] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *Proc. of SIGIR*, pages 207–214, Amsterdam, 2007.

[10] V. Hristidis, O. Valdivia, M. Vlachos, and P. S. Yu. Continuous keyword search on multiple text streams. In *Proc. of CIKM*, pages 802–803, Arlington, USA, 2006.

[11] M. Hu, A. Sun, and E.-P. Lim. Event detection with common user interests. In *Proc. of WIDM*, pages 1–8, Napa Valley, USA, 2008.

[12] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining Knowledge Discovery*, 7(4):373–397, 2003.

[13] G. Luo, C. Tang, and P. S. Yu. Resource-adaptive real-time new event detection. In *Proc. of SIGMOD*, pages 497–508, Beijing, China, 2007.

[14] W. R. Mahoney, P. Hospodka, W. Sousan, R. Nickell, and Q. Zhu. A coherent measurement of web-search relevance. *IEEE Trans. on Systems, Man and Cybernetics, Part A*, 39(6):1176–1187, 2009.

[15] A. Sun, M. Hu, and E.-P. Lim. Searching blogs and news: A study on popular queries. In *Proc. of SIGIR*, Singapore, July 2008.

[16] C. Wang, M. Zhang, L. Ru, and S. Ma. Automatic online news topic ranking using media focus and user attention based on aging theory. In *Proc. of CIKM*, pages 1033–1042, Napa Valley, USA, 2008.

[17] C. Yang, X. Shi, and C.-P. Wei. Discovering event evolution graphs from news corpora. *IEEE Trans. on Systems, Man and Cybernetics, Part A*, 39(4):850 – 863, 2009.

[18] Y. Yang, J. G. Carbonell, R. D. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.

[19] Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *Proc. of SIGIR*, pages 28–36, Australia, 1998.

[20] Q. Zhao, T.-Y. Liu, S. S. Bhowmick, and W.-Y. Ma. Event detection from evolution of click-through data. In *Proc. of KDD*, pages 484–493, Philadelphia, USA, 2006.