10-2008

Event detection with common user interests

Meishan HU

Aixin SUN
Nanyang Technological University

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

DOI: https://doi.org/10.1145/1458502.1458504

Event Detection with Common User Interests

Meishan Hu, Aixin Sun
School of Computer Engineering
Nanyang Technological University
Singapore 639798
{hu0004an, axsun}@ntu.edu.sg

Ee-Peng Lim
School of Information Systems
Singapore Management University
Singapore 178902
eplim@smu.edu.sg

ABSTRACT

In this paper, we aim at detecting events of common user interests from huge volume of user-generated content. The degree of interest from common users in an event is evidenced by a significant surge of event-related queries issued to search for documents (e.g., news articles, blog posts) relevant to the event. Taking the stream of queries from users and the stream of documents as input, our proposed framework seamlessly integrates the two streams into a single stream of query profiles. A query profile is a set of documents matching a query at a given time. With the single stream of query profiles, the well-studied techniques in event detection (e.g., incremental clustering) could be easily applied. In our experiments using real data collected from Blog and News search engines respectively, the proposed technique achieved very high event detection accuracy.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*;
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*

General Terms

Experimentation, Design

Keywords

Event Detection, Query Profile, Popular Queries, Blog

1. INTRODUCTION

The user-generated content, such as queries, blog posts, and comments, has become an ideal source for understanding the interests of common users. Taking the common user interests as one input, many Information Retrieval (IR) tasks can potentially be more user-oriented by addressing the information needs of a large number of users. Most existing IR tasks, however, have largely ignored such valuable information sources. *Event detection* is one of them, which has always been an important research topic in IR and has many useful applications.

1.1 Motivation

In event detection, the objective is to detect events from a temporally-ordered stream of documents (e.g., news articles) and organize these documents according to the events they describe [2]. Most existing solutions addressed the problem using unsupervised learning approaches and no input from common users was considered in the process. Two potential problems exist for these solutions: (i) many documents that receive no interest from common users are processed, resulting in unnecessary computation overhead, and (ii) many detected events are less useful to most users as the events are not of their interests. Without knowing which document deserves more interests from common users than others, all documents are processed with equal importance. In the news domain, Google News[1] and Yahoo! News[2] are currently tracking 4500 and 5000 sources respectively [11], and it is computationally expensive to process every document received from these sources. For blogs, an event detection system would have to deal with millions of blogs as sources. In reality, however, most users are only interested in a small subset of news articles and blog posts. If we can determine the user-interested documents and use only these documents in event detection, we can significantly reduce the amount of documents to be processed and also effectively detect events that are of the interests of common users.

In this research, we determine common user interests manifested at most news/blog sites as users often search for news/blogs of their interests using keyword queries. Given the nature of news/blogs, many of these keywords are event-related [13, 16]. When a bursty event happens, a large number of queries related to the event is often issued by a large number of users around the world, making them the popular queries in that period. Figure 1 captures the popular queries published by Technorati[3] on 28 Dec 07, one day after the `Assassination of Benazir Bhutto`[4]. The first 4 most popular queries are all related to this event. A sharp increment in the number of blog posts mentioning *Benazir Bhutto* right after the event's happening was also observed. Such a surge in the number of blog posts mentioning the keyword suggests

WIDM'08, October 30, 2008, Napa Valley, California, USA.

[1] `http://news.google.com`
[2] `http://news.yahoo.com`
[3] `http://www.technorati.com/pop/`
[4] `http://en.wikipedia.org/wiki/Assassination_of_Benazir_Bhutto`

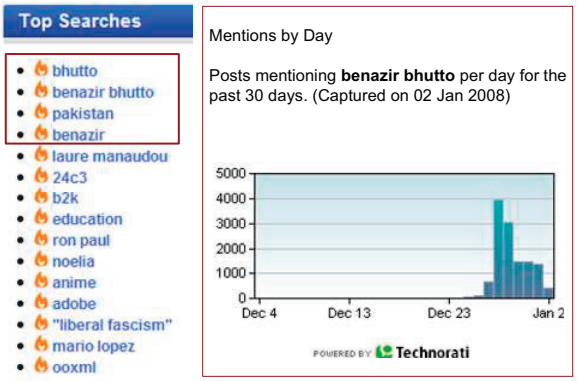


Figure 1: Top searches from Technorati.com

strong user interests in the event. Even for general-purpose search engines, it was observed that when an event happens, the number of related queries increases dramatically [25].

To summarize, information provided by common users, e.g., queries, gives strong indications of the events they concern or are interested in. With common user interests identified, an event detection process can be guided to process only those documents that are more likely to receive attention from common users and produce events of common user interests. At the same time, many documents that are lack of interests from common users can be safely ignored in the detection process, reducing the computational overhead.

1.2 Research Objectives and Contributions

In this paper, we focus on **detecting events of common user interests**. Given *a stream of documents* each attached with its publication time, and *a stream of popular queries* representing common user interests, our task is to group documents into events that are concerned by users. An **event of common user interests** to be detected therefore consists of:

- A set of documents extracted from the document stream that describes the event as defined in most existing event detection tasks.
- A set of representative keywords describing the event for easy browsing and searching. These keywords can be derived from the popular queries, the documents belonging to the event, or both.
- A time period within which common users are interested in the event. This time period maybe different from the start and end time detected in most existing event detections. In our setting, common users may pay attention to an event even before its happening (e.g., a release of a movie) for instance.

Detecting events of common user interests is challenging for two reasons. Firstly, *not all popular queries from users at a given time are event-related*. The observation that event-related queries increases dramatically after a major event happens does not necessarily imply that all popular queries are event-related. Identifying only those event-related queries is therefore a key challenge. Secondly, *multiple queries may be related to the same event, and the same query issued at different times may be related to different events*. As illustrated by our example in Figure 1, all 4 top queries refer to the same event. However, the query *Pakistan* may refer to another event happens in Pakistan if searched six months later.

We summarize our contributions in this work as follows:

1. We formally define the problem of *event detection of common user interests*. To address the problem, we propose a framework that nicely integrates a stream of queries representing common user interests and a stream of documents into a stream of query profiles. The well studied techniques in event detection can then be easily applied. In simple words, a query profile is a set of documents matching a given query at a given time.
2. We propose the notion of *temporal query profile* and define four measures, namely, *cohesiveness*, *time-span*, *content-similarity*, and *novelty*, to describe the properties of query profiles and their relationships. All these measures can be derived with minimum computational cost, making them ideal for online processing.
3. We conducted experiments using real data collected from Technorati and Google News. Other than showing events of common user interests can be effectively detected, our experiments also indicate that the proposed method can be easily integrated to any existing search engine for news/blogs.

1.3 Paper Organization

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 gives an overview of the proposed framework. The temporal query profile and its measures are described in Section 4, followed by the event detection algorithm in Section 5. Our experiments are presented in Section 6. We conclude the paper in Section 7.

2. RELATED WORK

Most event detection works address tasks defined in Topic Detection and Tracking (TDT), including new event detection, topic tracking and retrospective event detection [1, 2, 20]. The goal is to group documents (e.g., news articles) received from one or more temporally-ordered stream(s) according to the events that they describe. Our work is more related to new event detection and topic tracking.

One common approach is to model event detection as an online incremental clustering task [1, 2, 19, 20]. Documents received from a stream are processed in the order of their arrival. For each document, its similarities to the existing events (clusters of documents) are computed, and the document is assigned to either an existing event or a new event based on predefined criteria. Methods in this approach vary mainly in the way of computing the similarity between a document to an existing event [2, 3, 11, 12, 21, 22, 23].

Another approach for event detection is to identify bursty features from a document stream [7, 8, 10]. Features sharing similar bursty patterns in similar time periods are grouped together to describe events and also determine the periods of the bursty events. The detected bursty events are therefore described by a set of features, not a cluster of documents.

Event detection based on pre-given user queries is best represented by [6, 9]. Fung *et al* proposed a retrospective event detection by constructing an event hierarchy for a

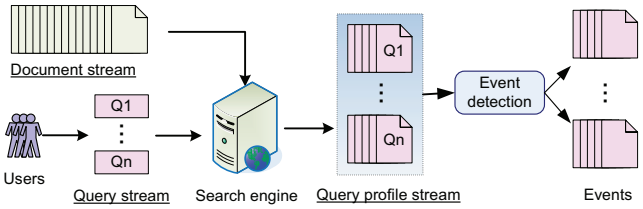


Figure 2: The proposed framework

given user query [6]. Bursty features related to the user query are first identified from the document collection. Documents related to those bursty features are then extracted from the collection and organized in an event hierarchy. In [9], a user specifies an event (or a topic) of interest using several keywords as a query. Multiple textual streams (e.g., news feed, emails) are monitored and the result to the query is a combination of streams that (i) sufficiently correlated, and (ii) collectively contain all query keywords within a time period. Different from that in [6, 9], in our problem setting, we aim to detect those events interest to many users in real-time and the queries from users are not pre-given.

Our work is also related to the studies on query logs of general search engines (e.g., MSN) [18, 24]. Event detection using click-through data proposed in [25] is based on two observations: (i) when an event occurs, the number of related queries increases dramatically, and (ii) there is strong co-occurrence between this set of queries and a set of webpages clicked. This work is similar to ours, as both aim to detect events based on user queries. However, three major differences exist. Firstly, in [25], only click-through data is used for event detection but not the web page content. In our work, we consider the content of the documents (news/blogs) without accessing the click-through data. Secondly, the query logs used in [25] are from a general search engine, which are quite different from news/blog search engines in nature and unavailable to the public. Thirdly, the proposed method in [25] is for retrospective event detection whereas ours is online event detection.

The notion of *temporal query profile* proposed in this paper is quite related to those reported in [4] and [5]. In [5], a *temporal query profile* is defined by the probability that a query is generated at a particular time. Our definition differs as we define a query profile to be the set of documents matching the query at a given time. Moreover, query profiles in [5] are not used for event detection. Chieu and Lee [4] also studied the problem of detecting events based on queries. In their problem setting, important events are those reported in a large number of news articles and each event is constructed according to one single query and represented by an extracted set of sentences. In our problem setting, we aim to detect events that are concerned with by a large number of common users. More importantly, events are formed online according to the affinity between queries measured using their temporal profiles and represented by the detected set of query profiles.

3. EVENT DETECTION FRAMEWORK

Our proposed framework takes two streams as input: (i) a *stream of documents*, and (ii) a *stream of popular queries* representing common user interests, as shown in Figure 2. Note that, in contrast, most existing event detection takes a stream of documents as the only input. To avoid processing documents that may not be interesting to common users, our proposed framework is mainly driven by the stream of popular queries. These queries can be obtained periodically (e.g., every 3 hours in our data collection). For each query, the subset of documents matching the query at that time is then retrieved from the document stream through a search engine, as shown in Figure 2. The retrieved subset of documents forms the query's temporal profile. Based on its properties, a query profile is either placed into the *query profile stream* if the query is likely to be event-related, or dropped otherwise. The query profile stream is then fed into the event detection module. Note that, by integrating the two streams into one stream of query profiles, our problem fits well into existing event detection framework, offering us the advantage of utilizing the well-studied techniques in detecting events. In this paper, we therefore adopt the simple online incremental clustering algorithm, commonly used in the event detection community, for grouping query profiles into events.

In summary, with the notion of query profile and the utilization of a search engine, our framework integrates the document stream and the popular query stream into one query profile stream, in a seamless manner. During the integration, *documents that do not match common user interests are naturally ignored* since they are not included in the search results of popular queries, leading to reduced computational cost. At the same time, *the popular queries are now attached with their query profiles where more semantics can be derived* to better facilitate event detection.

4. TEMPORAL QUERY PROFILE

In this section, we formally define temporal query profile, and then discuss properties of temporal query profile.

DEFINITION 1. **Temporal Query Profile**
Let $\mathcal{S}$ be a document stream where each document $d_i \in \mathcal{S}$ is associated with its time-stamp t_i. Let $D_{q,t}$ be the set of documents matching query q from document stream $\mathcal{S}$ at time t. The **temporal query profile** *of query q issued at time t over document stream $\mathcal{S}$, denoted by $P_{q,t}$, is the set of (at most) N_p most recent documents from $D_{q,t}$ that are created on or before t; i.e., $P_{q,t} \subseteq D_{q,t}$, $|P_{q,t}| \leq N_p$, and $\forall d_i \in P_{q,t}$, $\nexists d_j \in (D_{q,t} - P_{q,t})$ s.t. $t_j > t_i$.*

In simple words, a query profile is a set of recently published documents matching the given query. In the above definition, $|P_{q,t}|$ denotes the size of the temporal query profile. N_p is a predefined constant (N_p=50 in this paper). A query profile may be smaller than N_p if there are fewer documents matching q at a given time.

4.1 Query Profile Measures

We propose four measures to describe a query profile (i.e., *cohesiveness* and *time-span*) and the relationships between query profiles (i.e., *content-similarity* and *novelty*).

- *Cohesiveness* is a measure of the intra-similarity among the documents of a query profile, similar to that defined for clustering analysis [17]. We define cohesiveness by the averaged similarity between each document to the centroid of the query profile. Let v_i be the term vector derived from document d_i. The centroid $C(P_{q,t})$

and cohesiveness $H(P_{q,t})$ of a query profile $P_{q,t}$ are defined by Equations 1 and 2 respectively, where $cos(,)$ denotes the cosine similarity metric.

$$C(P_{q,t}) = \frac{1}{|P_{q,t}|} \sum_{d_i \in P_{q,t}} v_i \quad (1)$$

$$H(P_{q,t}) = \frac{1}{|P_{q,t}|} \sum_{d_i \in P_{q,t}} cos\,(v_i, C(P_{q,t})) \quad (2)$$

- *Time-span* refers to the length of the minimum time period covering the time-stamps of all documents in a query profile. Formally, the time-span $TSpan(P_{q,t})$ of query profile $P_{q,t}$ is given in Equation 3.

$$TSpan(P_{q,t}) = \max_{d_i \in P_{q,t}} (t_i) - \min_{d_j \in P_{q,t}} (t_j) \quad (3)$$

- *Content Similarity* between two query profiles is the similarity based on the content of the documents included in the query profiles. We use the cosine similarity between the centroids of the two query profiles $P_{q,t}$ and $P'_{q,t}$ for its simplicity and efficiency, denoted by $CSim(P_{q,t}, P'_{q,t})$. A similar measure has been proposed to compute the similarity between two short text snippets [14].

$$CSim(P_{q,t}, P'_{q,t}) = cos(C(P_{q,t}), C(P'_{q,t})) \quad (4)$$

- *Novelty* of query profile $P_{q,t}$ with respect to $P'_{q,t}$ measures the amount of information contained in $P_{q,t}$ that is not already known in $P'_{q,t}$. In this paper, it is defined by the ratio of the documents that are contained in $P_{q,t}$ but not in $P'_{q,t}$ (see Equation 5). Note that novelty measure is asymmetric.

$$Novelty(P_{q,t}, P'_{q,t}) = \frac{|P_{q,t} - P'_{q,t}|}{|P'_{q,t}|} \quad (5)$$

With the two measures of content similarity and novelty, it is relatively easy to tell the relationship between two profiles $P_{q,t}$ and $P'_{q,t}$:

1. $P_{q,t}$ is not relevant to $P'_{q,t}$ if $CSim(P_{q,t}, P'_{q,t})$ is small;
2. $P_{q,t}$ is relevant to $P'_{q,t}$ but contains no new information, if $CSim(P_{q,t}, P'_{q,t})$ is large but not $Novelty(P_{q,t}, P'_{q,t})$;
3. $P_{q,t}$ is relevant to $P'_{q,t}$ and contains new information to $P'_{q,t}$, if both $CSim(P_{q,t}, P'_{q,t})$ and $Novelty(P_{q,t}, P'_{q,t})$ are large.

4.2 Query Profiles and Events

In this section, we state some observations on searching document stream and then discuss our method using query profiles and the proposed measures.

Observation 1. *If a query q issued at time t is related to some event, there is likely a large number of documents describing the event matching q published within a short period before t.*

Here, we assume that the document stream to be searched contains enough documents. For instance, 4500 sources are subscribed by Google News, even if each source reports the event in one article, the number of articles matching a query 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 reporting. Observation 1 suggests a way to determine whether a user query q at time t is event-related. Recall that, a query profile $P_{q,t}$ is the set of (at most) N_p most recent documents matching q at t. The query profile for an event-related query should demonstrate (1) large cohesiveness, and (2) small time-span since the maximum size of a query profile is fixed at N_p.

Observation 2. *If both queries q and q' are related to the same event at time t, there are likely a large number of documents describing the event matching both q and q' published close to t.*

Referring to the real example given in Figure 1, users often search documents relevant to the event using name variants of the person involved in the event, and the location of the event. As all these information are likely to be mentioned together in every article reporting the event, all the four queries in Figure 1 are likely to share similar set of documents. In other words, if two queries q and q' are both related to the same event at time t, the content similarity between their query profiles, i.e., $CSim(P_{q,t}, P_{q',t})$ is expected to be higher than that between queries not related to the same event.

Observation 3. *If a query q is related to an event that lasts for some time, the documents matching q for two searches at time t_1 and t_2, both within the period of the event, are likely to describe the evolution of the event.*

An event often lasts for some time, e.g., several days or even weeks, and it is common for a user to search for the updates about the event using the same query related to the event at different time points. Searching the document stream with the same query q at different time points t_1 and t_2 ($t_2 > t_1$) leads to two query profiles, i.e., P_{q,t_1} and P_{q,t_2} respectively. $CSim(P_{q,t_1}, P_{q,t_2})$ might be large if (i) the time difference between t_1 and t_2 is relatively small, say one or two hours; or (ii) the time difference between t_1 and t_2 is relatively large, but the event evolves at very slow pace. In both cases, $Novelty(P_{q,t_2}, P_{q,t_1})$ might be low if not many new documents relevant to the event were published between t_1 and t_2. On the other hand, if time difference between t_1 and t_2 is relatively large and the event evolves very fast, $CSim(P_{q,t_1}, P_{q,t_2})$ could be small despite they are relevant to the same event.

In summary, the three observations, with the proposed measures, can be used for query profiles and our event detection.

5. ONLINE EVENT DETECTION

With the measures defined on query profiles, we propose an event detection process utilizing the simple online clustering algorithm. The process consists of three major modules, namely, *event-related query identification*, *event assignment*, and *event archive*.

5.1 Event-Related Query Identification

As event evolution takes time, it is unnecessary to evaluate popular queries too frequently. We partition the time into continuous and non-overlapping intervals T_σ ($T_\sigma = 3$ hours in our experiments). At the end of each time interval, the

set of popular queries[5] is examined and their profiles are built.

Let t_σ be the end time of the last time interval and Q_σ be the set of popular queries obtained. For each query $q \in Q_\sigma$, its query profile P_{q,t_σ} is built through blogs/news search engine(s) as these search engines match queries with both content similarity and recency [13]. Based on the observations discussed in Section 4.2, we identify event-related queries based on both the cohesiveness and the time-span of the query profiles. Specifically, a query q at time t_σ is expected to be event-related if $H(P_{q,t_\sigma}) \geq H_\theta$ and $TSpan(P_{q,t_\sigma}) \leq T_\theta$, where both H_θ and T_θ are pre-defined threshold. Values for H_θ and T_θ can be determined by analyzing the data collected.

5.2 Event Detection Algorithm

5.2.1 Event Assignment

We adopt the simple online clustering algorithm to assign query profiles to events. That is, an event is a subset of query profiles. Given the stream of event-related query profiles, the algorithm takes the first query profile to form one event. For each new-coming query profile, $P_{q,t}$, its similarities to all existing events are computed. Let E_k be the nearest neighbor of $P_{q,t}$, and $Sim(P_{q,t}, E_k)$ be their similarity, $Novelty(P_{q,t}, E_k)$ be the novelty of $P_{q,t}$ with respect to E_k. The query profile $P_{q,t}$ is processed with the following rules, where S_θ and V_θ are pre-defined thresholds.

- If $Sim(P_{q,t}, E_k) < S_\theta$, a new event is created using $P_{q,t}$.
- If $Sim(P_{q,t}, E_k) \geq S_\theta$ and $Novelty(P_{q,t}, E_k) < V_\theta$, then query profile $P_{q,t}$ is dropped, since inclusion of $P_{q,t}$ does not introduce new information to E_k.
- If $Sim(P_{q,t}, E_k) \geq S_\theta$ and $Novelty(P_{q,t}, E_k) \geq V_\theta$, then query profile $P_{q,t}$ is assigned to E_k. The representation of E_k is updated accordingly.

In the above discussion, $Novelty(P_{q,t}, E_k)$ is defined by the minimum novelty between $P_{q,t}$ to any query profile contained in E_k, shown in Equation 6.

$$Novelty(P_{q,t}, E_k) = \min_{P'_{q,t} \in E_k} Novelty(P_{q,t}, P'_{q,t}) \quad (6)$$

5.2.2 Similarity Computation

To compute the similarity between a query profile and an event, one simple option is to represent an event using the centroid of all query profiles (or documents) contained in the event. However, an event may last for a long time and evolve at a fast pace, the new coming query profiles may not be similar to some old query profiles contained in the same event.

Among all the query profiles contained in event E_k, let $\mathcal{P}_k$ be the set of query profiles that are created within W days (W=10 in our experiments) with respect to the most recent query profile in E_k. We represent E_k using the centroids of query profiles in $\mathcal{P}_k$, denoted by $C(E_k)$, (see Equation 7).

$$C(E_k) = \frac{1}{|\mathcal{P}_k|} \sum_{P_{q,t} \in \mathcal{P}_k} C(P_{q,t}) \quad (7)$$

The similarity between a query profile $P_{q,t}$ and an event E_k consists of two components: *content similarity* and *query similarity*. The content similarity is defined by the cosine similarity between the centroid of $P_{q,t}$ and that of E_k. The query similarity is applicable only if there exists a query profile in E_k whose query q matches the query in $P_{q,t}$. Let $P_{q,t'}$ be the latest created query profile in E_k matching the query of $P_{q,t}$. Let $Diff(t, t')$ be the time difference between t and t' in number of days. The query similarity between $P_{q,t}$ and E_k is defined in Equation 8.

$$QSim(P_{q,t}, E_k) = exp^{-\frac{Diff(t,t')}{2}} \quad (8)$$

Note that $QSim(P_{q,t}, E_k)$=0 if there is no existing query profile in E_k has the same query as $P_{q,t}$.

User may use the same query to search for what is going on for an event. In the case that the event evolves very fast, the content similarity may fail to decide whether a new coming query profile should be assigned to that event. Query similarity is therefore designed based on the heuristics that the same query issued within a short period is likely related to the same event.

Hence, the similarity between $P_{q,t}$ and E_k is computed as in Equation 9.

$$Sim(P_{q,t}, E_k) = cos(C(P_{q,t}), C(E_k)) + QSim(P_{q,t}, E_k) \quad (9)$$

5.2.3 Event Archive

Typically users are interested in an event for a finite period of time. Hence, events that are relatively old can be archived and removed from the main memory to speedup the computation. In our experiments, if an event has not received any new query profile for a certain number of days (e.g., 10 days in our experiments), the event is recorded to a database. This can significantly reduce the number of events stored in main memory and hence reduce the cost in computing the similarities for a new-coming query profile.

6. EXPERIMENTS

6.1 Data Collection

We collected the 15 most popular queries published by Technorati every 3 hours from 2006-11-08 1AM to 2008-03-31 10PM (17 months). Each of these popular queries was submitted to Technorati (**TR** for short) and Google News (**GN** for short) independently. The first 50 results returned by each search engine were recorded for building query profiles[6]. The title and snippet of each search result are extracted as the content of each document. These documents were preprocessed by punctuation removal, stopword removal and stemming using Lucene[7] and KStem[8]. Note that the query profiles built from GN and TR are used separately in the event detection process.

6.2 Dataset Analysis

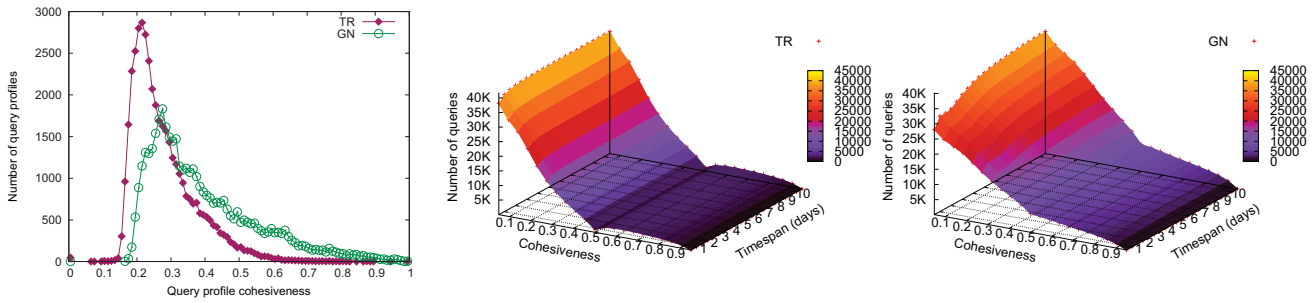
We first report some statistics on the query profiles built from each search engine. Note that all our discussions here-

[5]As we do not have access to query log of any commercial search engine, in our experiments, we subscribed the popular queries published by Technorati.com. The details on how to obtain these popular queries can be found in [16].

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

[7]`http://lucene.apache.org/`

[8]`http://ciir.cs.umass.edu/`



(a) Profile cohesiveness distribution (b) Technorati (c) Google News

Figure 3: Cohesiveness and Query Profiles

Table 1: List of parameters used in event detection

Parameter	Symbol	Value for TR	Value for GN
Cohesiveness	H_θ	0.25	0.35
Time-span(hour)	T_θ	60	48
Similarity	S_θ	0.35	0.40
Novelty	V_θ	0.5	0.5

after are based on query profiles having size of 50 (the predefined query profile size N_p).

Figure 3(a) plots the distribution of query profiles against their cohesiveness, from TR and GN respectively. On average, cohesiveness of the query profiles built from GN is higher than that from TR. The median of the query profile cohesiveness from TR and GN is 0.27 and 0.36 respectively. Given the nature of blogs being noisy and informal, the discussions in blogs are expected to be more diverse than that in news articles.

Figures 3(b) and 3(c) plot the number of query profiles with respect to cohesiveness and time-span filtering. Each point on the plotted surface indicates the number of query profiles that would be identified as event-related when cohesiveness threshold H_θ is set to its x-axis value and time-span threshold T_θ set to its y-axis value. Both figures suggest that the time-span is likely short for query profiles that have high cohesiveness. On the other hand, for those with low cohesiveness, the time-span could be up to several days. Compared to that of TR, query profiles built from GN are more likely to have long time-span. This, however, might be attributed to the ranking mechanism used by the two search engines.

Values for all parameters in our experiments are listed in Table 1. The cohesiveness and time-span thresholds, i.e., H_θ and T_θ, were set mainly based on the median/mean of query profiles from TR and GN respectively. In strict online setting, these values can be estimated when a large number of query profiles have been processed. For the similarity threshold S_θ, we experimented by varying S_θ for TR and GN respectively. Our results showed that, within certain range (i.e., [0.25, 0.45] for TR, and [0.30, 0.50] for GN), different values for S_θ had no significantly different impact on the duration of the detected events. Hence, we set 0.35 and 0.40 respectively for TR and GN.

6.3 Performance Evaluation

Evaluating the proposed technique is challenging, as there is no ground truth labels for the detected events. In our experiments, by referencing to a single information repository, i.e., Wikipedia[9], we manually examined the detected events, where quality of the verification is ensured. Nevertheless, it is well understood that not all events are worth recording in Wikipedia. For each detected event whose duration is not shorter than 1 day, we searched Wikipedia using the queries describing the event and labeled the events with one of the four labels:

- *True event*: If the event is recorded in a Wikipedia article and the time of the recorded event is within a short period of the detected duration[10].
- *Segment event*: If the detected event is wrongly split from a detected true event to which it should belong;
- *Mixed event*: If the detected event contains queries and documents from two or more events recorded in Wikipedia.
- *Unknown event*: If we cannot locate an entry recording the event in Wikipedia. Most events mainly containing non-English queries were also labeled as unknown due to the lack of understanding about their meaning.

Due to the limited resource, we only labeled the events detected using the parameter setting given in Table 1. From our experimental results, we found that the proposed method and the set of parameters used could detect events with high accuracy.

Table 2 reports the number of labels in the detected events using profiles built from TR and GN respectively. For each row in the table, the numbers of events with different labels are derived from all the events having the minimum duration specified in the left-most cell. Overall, the true-event detection rate is fairly high, with 71% of TR and 61% of GN for events with minimum duration of 1 day. For events lasted for a long period, e.g. 4 weeks, many are mixed or unknown events and the percentage of true events reduced to 63% and 49% respectively. Compared with that from GN, the events detected from TR achieved higher accuracy. One possible reason might be that the queries were collected from blog search and some of the events may not be well covered by news. We argue that the true event detection rate is fairly high, as many events were labeled "unknown", as not all events are worth recording in Wikipedia. Nevertheless, we believe the proposed technique would enjoy better accuracy if the detected events were evaluated by common users during the detection process.

[9] `http://en.wikipedia.org/`

[10] There could be time delay from the event occurrence to its attracting attention from common users.

In our labeling process, we observed that events having high bursty rate were often accurately detected. We define event bursty rate by the ratio of its size (i.e., the number of query profiles it contains) against its duration. In another words, bursty events are those attracted many popular queries in a short period. We hence report the top 20 events that have the highest bursty rates and lasted for at least 10 days, shown in Tables 3 and 4 for TR and GN respectively, ordered by bursty rate. For each event, we show its related queries followed by its number of profiles. The *true*, *segment*, *mixed*, and *unknown* events are indicated by '✓', '♯', '†', and '?' respectively (e.g., E_2 is a segment of E_4 in Table 4). The URL of the Wikipedia entry[11] is provided for *true* events.

Among the top 20 events from TR in Table 3, 16 were correctly detected. Given the nature of blog posts being noisy and diverse, it is natural for bloggers to discuss several related events in a single post. In such cases, our method may therefore fail to separate events from one another (e.g.,*windows vista*, *iphone*, *wii* in E_{16}). Among the 20 events from GN in Table 4, 17 were correctly detected, excluding the segment event E_2. The events about which we are not very clear are E_9 and E_{10}, consisting of queries *pool* and *none* respectively. Comparing the events detected from TR and GN, not surprisingly, we observed that the discussions in blogs is much more informal than that in news and may not be well supported by facts. For instance, *wayne chiang* (who was wrongly identified as the shooter), was included in the event `Virginia Tech massacre` detected from TR but not in the same event detected from GN.

In summary, we evaluated the proposed event detection approach using real world data, i.e., query stream collected from blog search and document streams retrieved from two search engines respectively (i.e., TR and GN). Our experimental results showed very high accuracy in terms of true event detected. In particular, events having high bursty rate were often accurately detected probably attributed to strong interests of common users.

7. CONCLUSION

In this paper, we studied the problem of detecting events that are of interests of common users. An event attracted much user interests is evidenced by a large number of queries related to the event issued by users. Taking popular queries as indication of common user interests, we proposed a framework that seamlessly integrates the stream of documents and the stream of popular queries with the utilization of a search engine. For the integration, we proposed the notion of *temporal query profile* and defined four measures on it. With the stream of query profiles, our problem well fits into existing event detection framework and a few well-studied techniques can therefore be utilized to detect events. In specific, we employed online incremental clustering algorithm, a commonly-used technique in event detection. Using real data, our experimental results showed that the events could be detected with very high accuracy, from both news articles and blog posts. More importantly, our proposed method can be easily implemented on top of or integrated to existing search engines for document stream without any modification to their implementations.

[11]Only the part following `http://en.wikipedia.org/` is shown in the Tables.

8. REFERENCES

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *Proc. 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'98*, pages 37–45, Melbourne, Australia, 1998. ACM.

[3] T. Brants and F. Chen. A system for new event detection. In *Proc. of SIGIR'03*, pages 330–337, Toronto, Canada, 2003. ACM.

[4] H. L. Chieu and Y. K. Lee. Query based event extraction along a timeline. In *Proc. of SIGIR '04*, pages 425–432, Sheffield, United Kingdom, 2004.

[5] F. Diaz and R. Jones. Using temporal profiles of queries for precision prediction. In *Proc. of SIGIR '04*, pages 18–24, Sheffield, United Kingdom, 2004.

[6] G. P. C. Fung, J. X. Yu, H. Liu, and P. S. Yu. Time-dependent event hierarchy construction. In *Proc. KDD'07*, pages 300–309, San Jose, California, USA, 2007. ACM.

[7] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *Proc. VLDB'05*, pages 181–192, Trondheim, Norway, 2005. VLDB Endowment.

[8] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *Proc. SIGIR'07*, pages 207–214, Amsterdam, The Netherlands, 2007. ACM.

[9] V. Hristidis, O. Valdivia, M. Vlachos, and P. S. Yu. Continuous keyword search on multiple text streams. In *Proc. CIKM'06*, pages 802–803, Arlington, Virginia, USA, 2006. ACM.

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

[11] G. Luo, C. Tang, and P. S. Yu. Resource-adaptive real-time new event detection. In *Proc. of SIGMOD'07*, pages 497–508, Beijing, China, 2007. ACM.

[12] J. Makkonen, H. Ahonen-Myka, and M. Salmenkivi. Simple semantics in topic detection and tracking. *Information Retrieval*, 7(3-4):347–368, 2004.

[13] G. Mishne and M. de Rijke. A study of blog search. In *Proc. of ECIR'06*, pages 289–301, London, UK, 2006.

[14] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. of WWW'06*, pages 377–386, Edinburgh, Scotland, 2006. ACM.

[15] I. Soboroff and D. Harman. Novelty detection: the trec experience. In *Proc. of HLT'05*, pages 105–112, Vancouver, British Columbia, Canada, 2005. ACL.

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

[17] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, June 2006.

[18] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proc. SIGMOD'04*, pages 131–142, Paris, France, 2004. ACM.

[19] 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.

[20] Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *Proc. SIGIR'98*, pages 28–36, Melbourne, Australia, 1998. ACM.

[21] Y. Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned novelty detection. In *Proc. KDD'02*, pages 688–693, Edmonton, Alberta, Canada, 2002. ACM.

[22] K. Zhang, J. Z. Li, and G. Wu. New event detection based on indexing-tree and named entity. In *Proc. SIGIR'07*, pages 215–222, Amsterdam, The Netherlands, 2007. ACM.

[23] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *Proc. SIGIR'02*, pages 81–88, Tampere, Finland, 2002. ACM.

[24] Q. Zhao, S. C. H. Hoi, T.-Y. Liu, S. S. Bhowmick, M. R. Lyu, and W.-Y. Ma. Time-dependent semantic similarity measure of queries using historical click-through data. In *Proc. of WWW'06*, pages 543–552, Edinburgh, Scotland, 2006. ACM.

[25] Q. Zhao, T.-Y. Liu, S. S. Bhowmick, and W.-Y. Ma. Event detection from evolution of click-through data. In *Proc. of KDD'06*, pages 484–493, Philadelphia, PA, USA, 2006. ACM.

Table 2: Distribution of the labeled events from TR and GN

	Events detected from TR					Events detected from GN				
Duration	True	Segment	Mixed	Unknown	Total	True	Segment	Mixed	Unknown	Total
1 day	371 (71%)	1	23	129	524	441 (61%)	5	15	265	726
2 days	279 (69%)	1	20	104	404	338 (58%)	4	12	228	582
3 days	229 (69%)	1	17	83	330	275 (57%)	3	11	193	482
1 week	135 (66%)	1	15	54	205	170 (54%)	2	9	134	315
2 weeks	67 (64%)	1	12	24	104	92 (53%)	1	4	76	173
3 weeks	37 (64%)	0	6	15	58	57 (52%)	0	3	50	110
4 weeks	26 (63%)	0	4	11	41	39 (49%)	0	3	38	80

Table 3: Top 20 events based on event burst rate detected from TR

Id	Label	Start, end dates	Size	Queries
E_1.	✓	2007-04-17 2007-04-27	136	cho seung hui(52), virginia tech(47), ismail ax(23), blacksburg(11), wayne chiang(3) [Virginia_Tech_massacre]
E_2.	✓	2006-12-30 2007-01-15	213	saddam execution video(40), saddam execution(39), saddam hanging(33), saddam(32), saddam video(32), saddam hussein(23), saddam hanging video(11), hussein(2), saddam hussein execution(1) [Execution_of_Saddam_Hussein]
E_3.	✓	2007-09-24 2007-10-06	105	burma(53), myanmar(27), free burma(25) [2007_Burmese_anti-government_protests]
E_4.	✓	2007-07-10 2007-08-07	236	harry potter(152), harry potter spoiler(50), deathly hallows(29), harry potter spoilers(3), harry potter ending(2) [Harry_Potter_and_the_Deathly_Hallows]
E_5.	✓	2007-04-30 2007-05-10	79	silverlight(58), mix07(21) [Microsoft_Silverlight]
E_6.	✓	2007-04-09 2007-04-19	71	imus(68), don imus(3) [Don_Imus]
E_7.	✓	2007-08-28 2007-09-05	53	larry craig(53) [Larry_Craig]
E_8.	✓	2007-02-09 2007-02-26	108	anna nicole smith(108) [Anna_nicole_smith]
E_9.	✓	2007-06-05 2007-06-15	61	sopranos(46), sopranos finale(15) [The_Sopranos]
E_{10}.	✓	2007-05-05 2008-03-06	1862	ron paul(1862) [Ron_Paul_presidential_campaign_2008]
E_{11}.	✓	2007-07-03 2007-07-11	48	transformers(48) [Tranformers_(film)]
E_{12}.	✓	2006-11-21 2006-12-03	71	michael richards(53), kramer(18) [Michael_richards]
E_{13}.	†	2008-03-11 2008-03-21	57	spitzer(23), ashley alexandra dupre(15), ashley dupre(10), dupre(4), eliot spitzer(3), myspace(1), kristen(1)
E_{14}.	✓	2007-03-04 2007-03-12	45	ann coulter(35), matt sanchez(10) [Matt_Sanchez]
E_{15}.	?	2007-05-25 2008-03-31	1734	youtube(804), naruto(274), yahoo(134), rss(101), seo(62), skype(51), ipod(40), ipod touch(38), wii(28), google(18)
E_{16}.	†	2006-11-08 2007-04-04	811	zune(225), youtube(209), windows vista(171), iphone(33), wii(32), apple tv(32), macworld(31), google(12), vista(12), emi(9)
E_{17}.	†	2007-02-22 2007-06-01	519	sanjaya(174), american idol(169), antonella barba(158), melinda doolittle(17), barba(1)
E_{18}.	✓	2008-01-14 2008-01-24	51	macbook air(19), macworld(17), macworld 2008(5), apple(4), steve jobs(3), ipod touch(2), engadget(1) [Macbook_air]
E_{19}.	✓	2007-05-20 2007-06-01	61	starcraft 2(61) [StarCraft_II]
E_{20}.	✓	2006-12-01 2006-12-12	55	james kim(55) [James_kim]

Table 4: Top 20 events based on event burst rate detected from GN

Id	Label	Start, end date	Size	Queries
E_1.	✓	2007-04-17 2007-04-27	133	virginia tech(69), cho seung hui(53), blacksburg(11) [Virginia_Tech_massacre]
E_2.	♯	2006-12-30 2007-01-10	133	saddam video(59), saddam execution video(41), saddam hussein video(14), saddam hanging video(10), video(9) [Execution_of_Saddam_Hussein]
E_3.	✓	2007-09-24 2007-10-05	102	burma(61), myanmar(27), free burma(14) [2007_Burmese_anti-government_protests]]
E_4.	✓	2006-12-30 2007-01-29	269	saddam(176), saddam execution(37), saddam hanging(33), saddam hussein(12), hussein(7), saddam hussein execution(4)[Execution_of_Saddam_Hussein]
E_5.	✓	2007-04-09 2007-04-19	85	imus(76), don imus(9) [Don_Imus]
E_6.	✓	2007-08-28 2007-09-05	59	larry craig(59)[Larry_Craig]
E_7.	✓	2007-07-10 2007-08-07	197	harry potter(153), deathly hallows(28), harry potter spoiler(12), harry potter spoilers(3), harry potter ending(1) [Harry_Potter_and_the_Deathly_Hallows]
E_8.	✓	2007-03-25 2007-04-09	105	iran(105) [Nuclear_program_of_Iran]
E_9.	?	2006-11-13 2006-11-26	86	pool(86)
E_{10}.	?	2007-12-05 2007-12-19	90	none(90)
E_{11}.	✓	2007-02-09 2007-02-26	109	anna nicole smith(109) [Anna_nicole_smith]
E_{12}.	✓	2008-01-25 2008-02-05	67	jerome kerviel(62), societe generale(5) [Jerome_kerviel]
E_{13}.	✓	2006-11-21 2006-12-03	70	michael richards(46), kramer(24) [Michael_richards]
E_{14}.	✓	2007-07-03 2007-07-11	42	transformers(42) [Transformers_(film)]
E_{15}.	✓	2007-03-01 2007-04-08	196	dell(196) [Dell]
E_{16}.	✓	2007-06-05 2007-06-15	51	sopranos(39), sopranos finale(12) [The_Sopranos]
E_{17}.	✓	2007-01-08 2007-03-01	261	iphone(175), apple(36), macworld(33), apple tv(15), apple phone(1), ipod(1) [Iphone]
E_{18}.	✓	2007-09-09 2007-09-17	39	madeleine mccann(36), mccann(3) [Madeleine_Mccann]
E_{19}.	✓	2006-11-08 2007-01-08	291	britney spears(146), britney(133), spears(8), britney spears crotch(2), britney spear(2) [Britney_Spears]
E_{20}.	✓	2007-01-17 2007-03-07	216	obama(167), hillary clinton(40), barack obama(9) [2008_U.S._presidential_election]