Detecting Hot Events from Web Search Logs

Yingqin Gu[1,2] , Jianwei Cui[1,2], Hongyan Liu[3], Xuan Jiang[1,2],
Jun He[1,2], Xiaoyong Du[1,2], Zhixu Li[1,2]

[1] Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education, China
[2] School of Information, Renmin University of China, Beijing 100872, China
{guyingqin, cjwruc, jx, hejun, duyong, lizx}@ruc.edu.cn
[3] Department of Management Science and Engineering, Tsinghua University,
Beijing 100084, China
hyliu@tsinghua.edu.cn

Abstract. Detecting events from web resources is a challenging task, attracting many attentions in recent years. Web search log is an important data source for event detection because the information it contains reflects users' activities and interestingness to various real world events. There are three major issues for event detection from web search logs: effectiveness, efficiency and the organization of detected events. In this paper, we develop a novel Topic and Event Detection method, *TED*, to address these issues. We first divide the whole data into *topics* for efficiency consideration, and then incorporate link information, temporal information and query content to ensure the quality of detected events. Finally, events detected are organized through the proposed *interestingness measure* as well as topics they belong to. Experiments are conducted on a commercial search engine log. The results demonstrate that our method can effectively and efficiently detect hot events and give a meaningful organization of them.

Keywords: Event detection, click-through data, efficient algorithm

1 Introduction

As the richness of information on the web grows, more and more researchers are motivated to discover knowledge such as topics and events from web data. The existing approaches of such event detection can be divided into three categories: *content-based* method, *structure-based* method and *log-based* method. In plain terms, the content-based method detects events through analyzing textual information of web resources using techniques such as natural language processing [1]; the structure-based method utilizes website structures and hyperlink structures to discover events [2]; and the log-based method, which is a new research direction in event detection, exploits web search logs (namely, the *click-through data*) to detect events [3] [4].

In [3], the author classifies web data into two types: *author-centric* and *visitor-centric*. Based on this classification, the former two approaches utilize the author-centric data, which are created by web publishers, and the log-based method utilizes the visitor-centric data, which can reflect users' browsing or query activities. Therefore, the log-based method can extract information such as users' activities and interestingness to various real world events, and such information may play equally or even more important role in event detection compared with the author-centric data. Besides, the click-through data are collected automatically by web search engine and are easier to obtain. Detecting events from such data has more practical significance. Because of the above reasons, our focus in this paper is to detect events form click-through data.

Table 1 shows an example of click-through data, which record the interactions between users and a web search engine. It can also be represented as a bipartite graph as showed in Fig. 1. From Table 1 we can observe that each record contains three elements: query keywords issued by users, URL of the page clicked by users and the time the query was submitted to the search engine. From the click-through data, three kinds of information can be used for event detection: *link information* (that is indicated by the queries and the corresponding clicked pages), *temporal information*(time) and *query content*(query keywords). Then our task is to utilize the above information to discover groups of click records (or query-page pairs in the bipartite graph) that correspond to real life events.

Table 1. Example of click-through data

Query	URL	Time
q1: David Blaine	p1: http://abcnews.go.com/GMA/…….	2006/5/2
q1: David Blaine	p2: http://www.katu.com/entertain......	2006/5/2
q2: Blaine	p1: http://abcnews.go.com/GMA/…….	2006/5/2
q2: Blaine	p2: http://www.katu.com/entertain......	2006/5/2
q3: David Blaine drowned	p3: http://abclocal.go.com/kabc/story...	2006/5/9

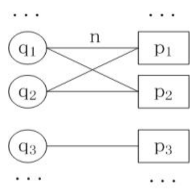

Fig. 1. Bipartite graph

In order to enhance event detection from click-through data, we deem that three issues need to be addressed. First, the click-through data are produced by users' query submission and browsing activities. It is quite noisy and sparse. Therefore, how to discover real life events with high precision and recall is a problem. Second, the web search logs usually contain at least millions of log records. The efficiency of event detection is an important issue. Last but not the least, the detected events need to be organized well so that from the results users can find the real world interesting events conveniently.

This paper aims to address the above three issues. In terms of the effectiveness, we propose a novel notion called *click-through burst region* as the event indicator for event detection and incorporate link information, temporal information and query content simultaneously to ensure the quality of detected events. As for the efficiency, we propose an efficient algorithm, which divides the whole data into several small semantic-relevant subsets, each of which is called a *topic*. Then the event detection is performed in each topic, making the whole process efficient. In order to organize the results orderly, we introduce an *interestingness* measure, which is similar to the work in [4], to rank the detected events. Besides, since we discover topics first, events extracted from one topic can be clustered naturally.

The proposed log-based event detection method is performed on a real life click-through dataset collected from MSN search engine. We also manually labeled a list of events to evaluate the quality of the detected events. The results show that by incorporating the three kinds of information the effectiveness is improved evidently, and from the performance on datasets of different size our algorithm shows high efficiency.

The rest of this paper is organized as follows: Section 2 reviews related work. In Section 3, we introduce our proposed log-based method for event detection. Experimental results on real life data are shown in Section 4. Finally, we draw conclusions and future work in Section 5.

2 Related Work

In this section, we will introduce some previous works in event detection area and discuss difference between previous works and our work.

Event detection is part of a broader area called Topic Detection and Tracking (TDT). TDT is a famous research project which can be divided into two categories [5] as Retrospective Event Detection (RED) [5] [6] and New Event Detection (NED) [7] [8]. At the moment, our work is more related to RED. RED is first proposed in [5] by Yang et al., and an agglomerative clustering algorithm called Group Average Clustering was proposed in that paper. In [6], the author proposed a multi-model RED algorithm, which incorporated both the content and the time information, and an approach was proposed to determine the approximate number of events. Besides TDT, many research works focus on detecting events by identifying burst patterns of content-based features from a text stream [9] [10] [11]. [9] introduced how to extract burst features from text streams, where the text stream is modeled as an infinite state automation and the bursts are modeled as transition states. In [10], the author proposed a parameter-free method to group related busty features into one event. In [11], a mixture density-based approach is proposed to detect bursts of word features for event detection. The difference between our work and above TDT research or content based methods is that we detect events based on different data source. From the visitor-centric data, we may find hotter and more interesting events.

Recently, with the development of search engine and Web2.0 technology, many researches have focused on exploiting web search logs for various purposes such as query expansion [12], named entity mining [13] and web search improvement [14], etc. However, to the best of our knowledge, there are only two of such works focusing on event detection [3] [4]. The work done by Zhao et al. [3] is the first effort, in which two phases of clustering based on semantic and temporal similarity respectively are performed to group similar query-page pairs corresponding to real life events. While in [4], the authors use query session as indicator of event. They first map each query session to a polar space based on content and time similarity, then group query sessions that are similar in content and close in time to represent an event. The general method of our work is different from the above two works. Besides, both the above two works run some time consuming algorithms in the whole click-through data and fail to mention the efficiency in their papers; while the algorithm we propose is quite efficient and applicable.

3 Framework of Our Method

The framework of our method, *TED*, for exploiting web search logs to detect events is presented in Fig. 2.

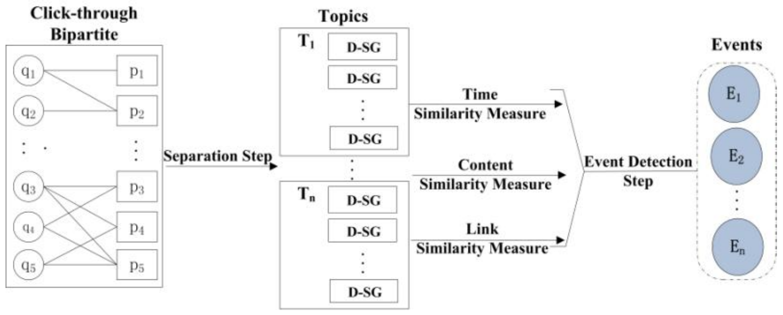

Fig. 2. The framework of exploiting click-through data for event detection

Given the click-through data, which can be represented as a bipartite graph, we first divide this graph into several sub-graphs each of which corresponds to a topic by applying a quite efficient algorithm called *Shingle* and utilizing query content information. Then, we propose a novel notion called *click-through burst region* as event indicator to further detect events from topics. For each topic, the events can be detected by clustering the similar burst regions. Here the link information, temporal information and query content are combined to measure the burst regions' similarity. Finally, we organize the detected events through their interestingness and the topics they belong to.

3.1 Dividing Click-through Data into Topics

In this step, we tend to divide the whole click-through data into several sub-graphs so that events can be detected efficiently in each small sub-graph. To do this, we implement an algorithm called *Shingle* [15]. The substance of algorithm *Shingle* is to use the Jaccard coefficient to measure similarities of nodes in a graph so that the nodes with similar neighbors tend to be grouped into one dense sub-graph. While its method is based on recursive stages of fingerprinting the graph, which after multiple applications can turn dense sub-graphs of arbitrary size into fingerprints of constant size. It is very efficient and is capable of handling graphs with tens of billions edges [15].

After running the algorithm Shingle, we get dense sub-graphs of click-through bipartite graph only based on the link information. While work in [16] illustrates that using link information alone will produce a high precision at a low recall level, which means that click records belong to one event may be separated into different dense graphs. To solve this problem, we further consider query content of each dense sub-graph and merge the content-similar sub-graphs into one larger sub-graph to reflect *topic* information. It can be done through comparing the similarity of query content in each dense sub-graph, here the classical *tfidf* and *cosine similarity* techniques are used to measure the similarity of queries, and the *inverted indexing* is used to speed up this step. Then a pruning strategy based on an interestingness measure which will be introduced in Section 3.3 is used so that uninteresting topics need not be considered anymore.

3.2 Event Detection

In this section, we will introduce our method to detect events from each topic. From each topic, we first propose a novel event indicator called *click-through burst region* (*burst region* for short). Then link information, temporal information and query content are combined linearly to measure the similarity between burst regions. Finally burst regions can be clustered as events.

3.2.1 Click-through Burst Region

Based on the intuitive premise used in [9], the appearance of an event can be signaled by a "burst of activity". Therefore, a sharp increase in time series of query-page pair's click-through (the number of click-throughs) tends to indicate a real life event. Based on this, we choose burst region defined below as event indicator.

Definition 1(*click-through burst region*) *click-through burst region* is a period in the time series of a query-page pair's click-through, in which the click through increased sharply than average click-through of the time series.

We calculate a threshold called *cutoff* by adding average and stand derivation of the click-through time series for detecting burst regions. Each of detected burst regions *B* can be represented by (T_{st}, T_{ed}), where T_{st} and T_{ed} is the start time and the end time of burst region B respectively.

3.2.2 Burst Region Similarity

In this section, we introduce the method to measure the similarity of two *click-through burst regions* by combining link, content and time similarity.

The link-based similarity between two burst regions B_i and B_j is denoted as *linkSim*(B_i, B_j). We take advantage of the method in [3] to calculate *linkSim*(B_i, B_j). First, bipartite graph of each topic is converted to a dual graph. Each edge between a query *q* and an URL *p* in the bipartite graph is converted to a node <*q*, *p*> of the dual graph. Two nodes in the dual graph are linked by an edge if they have the same query or URL. Then, we run algorithm *SimFusion* [17] to calculate the link similarity between different nodes to represent link-based similarity of burst regions as shown in equation (1). In equation (1), P_x is a query-page pair containing burst region B_i, and P_y contains burst region B_j.

$$linkSim(B_i, B_j) = SimFusion(P_x, P_y) \tag{1}$$

The content-based similarity is donated as *contentSim*(B_i, B_j) and is calculated by the query content similarity as introduced in Section 3.1.

$$conetentSim(B_i, B_j) = Sim(q_x, q_y) \tag{2}$$

The third type similarity we considered is the time-based similarity, which evaluates how near two burst regions happened by Equation (3):

$$timeSim(B_i, B_j) = \begin{cases} \dfrac{-distance(B_i, B_j)}{|B_i| + |B_j| + distance(B_i, B_j)} & \text{if } distance(B_i, B_j) >= 0 \\ \dfrac{1}{2}\left(\dfrac{overlap(B_i, B_j)}{|B_i|} + \dfrac{overlap(B_i, B_j)}{|B_j|}\right) & \text{otherwise} \end{cases} \quad \textbf{(3)}$$

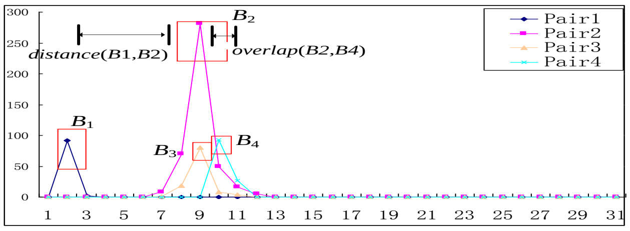

Fig. 3. Time-based similarity of four burst regions

In Equation (3), *distance*(B_i, B_j) describes how far two burst regions B_i and B_j are, which can be measured by the difference between the starting time of B_j and the ending time of B_i as shown in Fig. 3. *overlap*(B_i, B_j) is the time intersection of two burst regions, calculated by the difference between the ending time of B_j and the starting time of B_i. For example, in Fig. 3, *overlap*(B_2, B_4) is equal to T_{ed} of B_2 minus T_{st} of B_4. $|B_i|$ describes how long a burst region lasts. For example, B_1 in Fig.3 lasts four days and so $|B_1|$ equals 4. Therefore, by Equation (3), if two burst regions have overlap, the larger *overlap* they have, the more similar they are. Otherwise, the larger *distance* they have, the less similar they are.

Based on the three type similarity introduced above, the similarity between burst regions B_i and B_j, *Sim*(B_i, B_j), can be calculated by Equation (4):

$$Sim(B_i, B_j) = \alpha \cdot linkSim(B_i, B_j) + \beta \cdot contentSim(B_i, B_j) + \gamma \cdot timeSim(B_i, B_j) \quad \textbf{(4)}$$

α, β and γ are three parameters used to tradeoff the effectiveness of link, content and time similarity. How to set the three parameters will be discussed in the following experimental section. After getting the similarity between *click-through burst regions,* we utilize a parameter free hierarchy-based clustering method [19] to cluster similar burst regions, which uses *silhouette coefficient* [18] to judge when to terminate the clustering procedure. Each cluster obtained after this step corresponds to an event.

3.3 Organizing Detected Events

After detecting the events, effective organization of the detection results is important for improving the utility of our method. To do this, we develop an *interestingness measure* to indicate how hot and interesting a detected event is. An event is assumed to be more interesting if it can attract more users' attention and has a more obvious pattern of "burst of activity". So we introduce the following definition to measure the interestingness.

Definition 2(*attention degree*) Given a query-page pair *p* and event *E* it belongs to, the *attention degree* of *p*, denoted as *ad*(*p*), and the attention degree of *E*, denoted by *ad*(*E*), are defined as:

$$ad(p) = |p| / |E| \tag{5}$$

$$ad(E) = |E| / |D| \tag{6}$$

Here $|p|$ is the click-through count of p, $|E|$ is the click-through count of E to which p belongs to, and $|D|$ is the click-through count of all click-through. So, a higher *attention degree* means this query-page pair or event attracts more people's attention.

Definition 3(*burst rate*) Given a query-page pair p and the event E it belongs to, *burst rate* of p and E are defined as follows:

$$br(p) = \max(|p_i|) / |p| \tag{7}$$

$$br(E) = \sum_i (br(p_i) \cdot ad(p_i)) \tag{8}$$

Where p_i is the ith query-page pair in event E. Based on the above definitions, we give the notion of *interestingness measure*.

Definition 4 (*interestingness measure*) The *interestingness measure* of an event E considers both the attention degree factor and burst rate factor of this event, which is defined as follows:

$$im(E) = ad(E) \cdot br(E) \tag{9}$$

We calculate each event's interestingness measure to rank the detected events so that the more interesting an event is, the topper position it will be in the ranked event list. Furthermore, since we detect event in each topic, events detected from one topic can be organized together and listed by order of the timestamp of each event.

4 Experimental Results

4.1 Dataset

Our experiments are conducted on a real life click-through dataset collected from MSN search engine from May 1 to May 31, 2006. Our click-through dataset contains about 14 million clicks, 3.5 million distinct queries, 4.9 million distinct urls and 6.8 million distinct query-URL(query-page) pairs.

Table 2. List of labeled events

Topic	Event	Timestamp
David Blaine	Submerged in an aquarium	May 2, 2006
	Failed to break world record	May 9, 2006
Immigration Bill	Senate reached deal to revive immigration bill	May 11, 2006
	Senate passed immigration bill	May 25, 2006
Charles Barkley	Admit having gambling problem	May 4, 2006
	Called Bryant selfish	May 17, 2006
Chris Daughtry	Eliminated from American Idol	May 10, 2006
Stephen Colbert	"Attack" Bush at a dinner	May 1, 2006
Record Hammerhead	A world-record hammerhead shark caught	May 26, 2006
Ohio Bear Attack	Woman attacked by escaped bear	May 22, 2006
Hoffa Dig	Feds digging for Jimmy Hoffa	May 18, 2006
Pygmy Rabbit	Last male of pygmy rabbit dead	May 18, 2006
Kentucky Derby	Barbaro won the Kentucky Derby	May 6, 2006

To evaluate effectiveness of our method, we manually labeled 10 real life topics and 13 corresponding events from the dataset. Each labeled topic and event contains a set of query-page pairs. Given a topic, we label query-page pairs to represent the topic and the corresponding events by the following steps:

1. Select a set of queries, denoted as *Q*, which were frequently submitted and highly correlated to the topic.
2. The click-through data are represented as a bipartite graph, which contain a number of *connected sub-graphs*. From them, a set *G* of connected sub-graphs that contain the queries in *Q* are extracted.
3. For each connected sub-graph in *G*, filter out the queries and URLs whose frequencies are less than a given minimum threshold.
4. For each query-page pair in the filtered *G*, we manually check whether it belongs to the topic according to its page content. After that, a set of query-page pairs, denoted as *T*, are chosen to represent the topic.
5. For query-page pairs in *T*, if there is more than one event for the topic, we further annotate them into several parts to represent different events. Otherwise, there is only one event in *T*.

Based on the above method, we extract 13 real life events from the click-through dataset. The complete list of topics and corresponding events are shown in Table 2. Then from the labeled click-through records, we randomly choose other unrelated records from the whole datasets to generate larger and noisier data. We extracted four datasets: dataset1, dataset2, dataset3 and dataset4, whose numbers of click-through records are 250k, 350k, 600k and 1,000k respectively.

4.2 Parameter Setting

In this section, we learn how to set parameters in equation (4) and the experiment is performed on dataset1. Methods based on different linear combinations of link information, temporal information and query content are listed in Table 3.

Table 3. Detection schemes based on different combinations of informations

Notation	Explanation
Link	Detection solely based on link information
Time	Detection solely based on temporal information
Content	Detection solely based on query content
Link_Time	Detection based on linear combination of link information and temporal information
Link_Content	Detection based on linear combination of link information and query content
Time_Content	Detection based on linear combination of temporal information and query content
Link_Time_Content	Detection based on linear combination of link information, temporal information and query content

The parameter α, β and γ in equation (4) are set in the following way:

- For Detection based on Link_Content schema, γ is set to zero and α is set to 0.1, 0.3, …, 0.9 and β is set to 0.9, 0.7, …, 0.1 accordingly. It corresponds to five parameter setting cases: para1, para2, para3, para4 and para5 respectively. Parameters of Link_Time schema and Time_Content schema can be set similarly.
- For Link_Time_Content schema, α, β and γ are set according to the values that produce best results for Link_Content and Link_Time schemas.

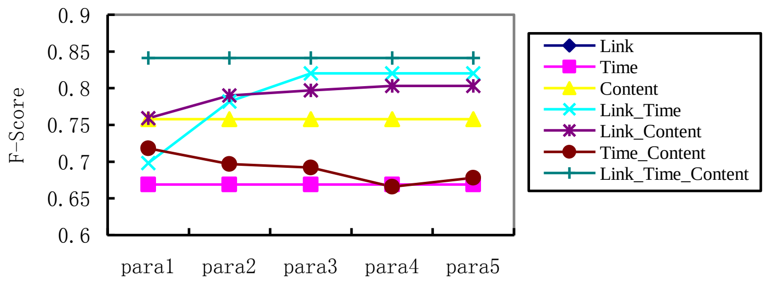

Fig. 4. Performance of different parameters setting

Fig. 4 shows the result of event detection on dataset1. F-score is used to measure the quality of detected events. The F-score of Link schema is quite low and it cannot be shown in the figure. By analyzing the detected events, we find that this is because most events detected by this schema have high precision but very low recall, which is consistent with the observation in [16]. Though the performance of the method *only based on link information* is poor, combining link information with temporal or content information often leads to better performance, and the performance based on the combination of all the three kinds of information is best.

4.3 Efficiency Evaluation

Setting the parameters according to the above experiment, we run our algorithm *TED* on the other three datasets. The experiment is conducted on a HP dc7700 desktop with an Intel Core2 Duo E4400 2.0 GHz CPU and 3.49 GB RAM. Figure 5 and 6 shows the runtime of separation step and F-score of our algorithm on different datasets.

From Figure 5 we can see that our algorithm is quite efficient. It can be run in minutes. Besides, with the augment of the datasets, the runtime of the algorithm increase linearly. The Figure 6 shows that with the noise data added, there is almost no reduce of the F-score of detected events. These two figures testify that our algorithm has quite high efficiency with good effectiveness.

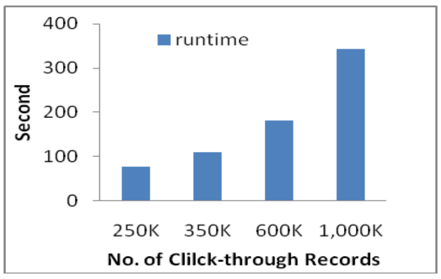

Fig. 5. Efficiency on four datasets

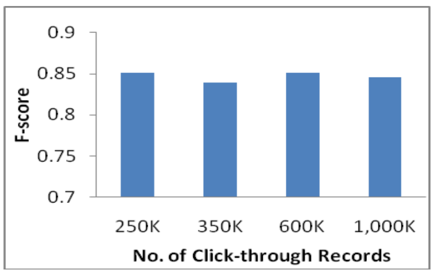

Fig. 6. Effectiveness on four datasets

4.4 Illustrative Examples

By running our algorithm *TED* on the dataset4, we get a list of ranked detected events. Table 4 shows the top 10 ten hot events. And all the labeled events can be found in the top 100 events.

Table 4. List of top 10 hot events

Rank	Events
1	David Blaine submerged in an aquarium
	Failed to break world record
2	Stephen Colbert "attack" Bush
3	No-event
4	Barbaro won in Kentucky Derby
5	Chris Daughtry eliminated from American Idol
6	Holloway case
7	Videotaped killing
8	Michelle Rodriguez
9	Judicial Watch
10	Oprah Winfrey's legend ball

From Table 4 we can find that events of rank 1, 2, 4 and 5 are events in the labeled dataset, and two events about David Blaine are organized together in chronological order. Events of rank 6, 7, 8, 9 and 10 are the newly detected events, among which events of rank 6 and 7 correspond to two criminal case news and event of rank 10 is about a legend ball held by Oprah Winfrey in honor of 25 African-American women in the field of art, while pages related to events of rank 8 and 9 corresponding to a movie star and a government organization are not available now or have been updated, so we have no idea what exactly happened at that time.

By analyzing the result, we find that our algorithm still has some shortcomings. For example, the detected event of rank 3, consists of few click records and it does not correspond to any real life event. This problem may be relieved through a more critical pruning strategy. Besides, although we consider the query content to solve the low-recall problem in separation step, an event may still be possible to be separated into several parts. For instance, we find another event about Kentucky Derby in the top 100 hot events.

4.5 Comparative Experiment

Work in [3], which proposed a two-phase-clustering method, is the one that is most similar to our method for event detection. In this section, we compare our method with it (we call it two-phase-clustering) on efficiency and effectiveness.

Since the two-phase-clustering method runs a time-consuming algorithm *SimFusion* on whole dataset, it cannot process very large click-through data in our experiment environment. To do the comparison, we generate another four smaller datasets: no_noise, noise_2, noise_4 and noise_8, which stand for labeled datasets with no noise, one time noise, three times noise and seven times noise respectively. Dataset of no_noise contains 35k click records.

For efficiency, we compare the runtime of first phase of [3] with the runtime of the separation step of *TED*. Both methods' second steps are performed on subset data and are relatively efficient. Fig. 7 shows the experimental result, where we can see that the runtime of our method increases much slower than that of Two-phase-clustering. In noise_8, our method consume about one minute while First Phase consume four days. So our method is much more efficient than the first phase of [3] in dealing with click-through data. As for effectiveness, Fig. 8 shows the result of F-score for four datasets. We can see that our method has a much higher F-score for each dataset, which means our event detection method performs better than that of [3].

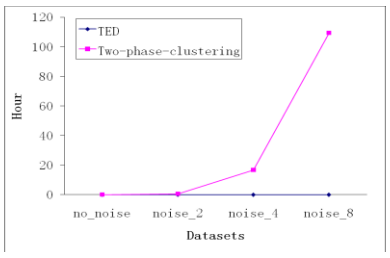

Fig. 7. Efficiency comparison

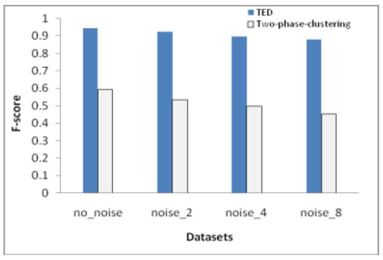

Fig. 8. Effectiveness comparison

5 Conclusion and Future Work

In this paper, we propose an efficient log-based method, *TED*, which incorporates link information, temporal information and query content for topic and event detection. Two major steps are contained in our method. The first is a separation step, in which we use algorithm *Shingle* to divide the whole data into dense sub-graphs and cluster query content similar sub-graphs as topics. Then in each topic, the event detection step is performed, in which we use the burst region as indicator of events and clustering similar burst regions by linearly combing link information, temporal information and query content. The experiments conducted on real life dataset show that our method can detect events effectively and efficiently from click-through data.

Practice asks for online detection of emerging events. For future work, we will extend our Retrospective Event Detection algorithm to New Event Detection. Moreover, we will explore how to utilize the page content to further improve the performance of our method.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grant No. 70871068, 70621061 and 70890083. We also thank Microsoft Research Asia for providing the real data and funding this research.

References

1. Yang, Y., Zhang, J., Carbonell, J., Jin, C.: Topic-conditioned Novelty Detection. In: 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 688—693. ACM Press, Edmonton (2002)
2. Sun, A., Lim, E.P.: Web Unit Mining: Finding and Classifying Subgraphs of Web Pages. In: 2003 ACM CIKM International Conference on Information and Knowledge Management, pp. 108—115. ACM Press, New Orleans(2003)
3. Zhao, Q., Liu, T.Y., Bhowmick, S.S., Ma, W.Y.: Event Detection from Evolution of Click-through Data. In: 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 484—493. ACM Press, Philadelphia(2006)

4. Chen, L., Hu, Y., Nejdl, W.: Deck: Detecting Events from Web Click-through Data. In: 8th IEEE International Conference on Data Mining, pp. 123—132. IEEE Press, Pisa(2008)
5. Yang, Y., Peirce, T., Carbonell, J.: A Study of Retrospective and On-line Event Detection. In: 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 28—36. ACM Press, Melbourne(1998)
6. Li, Z., Wang, B., Li, M., Ma, W.Y.: A Probabilistic Model for Retrospective News Event Detection. In: 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 106—113. ACM Press, Salvador(2005)
7. Allan, J., Papka, R., Lavrenko, V.: On-line New Event Detection and Tracking. In: 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 37—45. ACM Press, Melbourne(1998)
8. Luo, G., Tang, C., Yu, P.S.: Resource-adaptive Real-time New Event Detection. In: ACM SIGMOD International Conference on Management of Data, pp. 497—508. ACM Press, Beijing(2007)
9. Kleinbert, J.: Bursty and Hierarchical Structure in Streams. In: 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 91—101. ACM Press, Edmonton (2002)
10. Fung, G.P.C., Yu, J.X., Yu, P.S., Lu, H.: Parameter Free Bursty Events Detection in Text Streams. In: 31st International Conference on Very Large Data Bases, pp. 181—192. ACM Press, Trondheim(2005)
11. He, Q., Chang, K., Lim, E.P.: Analyzing Feature Trajectories for Event Detection. In: 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 207-214. ACM Press, Amsterdam(2007)
12. Cui, H., Wen, J.R., Nie, J.Y., Ma, W.Y.: Probabilistic Query Expansion Using Query Logs. In: 11th international conference on World Wide Web, pp. 325—332. ACM Press, Honolulu(2002)
13. Xu, G., Yang, S.H., Li, H.: Named Entity Mining from Click-Through Data Using Weakly Supervised Latent Dirichlet Allocation. In: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, PP. 1365-1374. ACM Press, Paris(2009)
14. Zhu, G., Mishne, G.: Mining Rich Session Context to Improve Web Search. In: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, PP. 1037—1046. ACM Press, Paris(2009)
15. David, G., Ravi, K., Andrew, T.: Discovering Large Dense Subgraphs in Massive Graphs. In: 31st International Conference on Very Large Data Bases, pp. 721—732. ACM Press, Trondheim(2005)
16. Filippo, M.: Combining Link and Content Analysis to Estimate Semantic Similarity. In: 13th international conference on World Wide Web, pp. 452—453. ACM Press, New York(2004)
17. Xi, W., Fox, E., Fan, W., Zhang, B., Chen, Z., Yan, J., Zhuang, D.: SimFusion: Measuring Similarity using Unified Relationship Matix. In: 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 130—137. ACM Press, Salvador(2005)
18. Leonard, K., Peter, J.R.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley-Interscience(1990)
19. Cui, J., Li, P., Liu, H., He, J., Du, X.: A Neighborhood Search Method for Link-Based Tag Clustering. In: 5th International Conference on Advanced Data Mining and Application, pp. 91-103. Springer Press, Beijing(2009)