ECO: Event Detection from Click-through Data via Query Clustering

Prabhu K. Angajala, Sanjay K. Madria, and Mark Linderman

Department of Computer Science, Missouri University of Science and Technology, MO, USA
Air Force Research Lab, Rome, NY, USA
madrias@mst.edu

Abstract. In this paper, we propose an algorithm to detect real world events from the click-through data. Our approach differs from the existing work as we: (i) consider the click-through data as collaborative query sessions instead of mere web logs proposed by many others (ii) integrate the semantics, structure, and content of queries and pages, and (iii) aim to achieve the overall objective via query clustering. The problem of event detection is transformed into query clustering by generating clusters using hybrid cover graphs where each hybrid cover graph corresponds to a real-world event. The evolutionary pattern for the co-occurrence of query-page pairs in a hybrid cover graph is imposed over a moving window period. Finally, we experimentally evaluated our proposed approach using a commercial search engine's data collected over 3 months with about 20 million web queries and page clicks from 650,000 users. Our method outperforms the most recent event detection work proposed using complex methods in terms of metrics such as number of events detected, F-measures, entropy, recall etc.

1 Introduction

The approximate size of today's indexed World Wide Web is at least 45.93 billion pages as per existing estimation [1] and is a rich collection of all the real world objects. Web is a great source of knowledge to be mined to learn about topics, stories, events etc. Event detection is becoming increasingly popular because of its applicability in several diversified areas. Therefore, the interpretation of "event" definition is context-dependent. An event can be associated with reporting how many people/cars have entered a building/freeway, web access logs, security logs, object trajectory in video surveillance and business activity monitoring for business intelligence etc. In our perspective and from the viewpoint of Web, an event can be understood as some real-world activity that stirs a large scale querying and browsing activity. That is, it is of more interest to users over a sizable window period which is unusual relative to normal patterns of querying and browsing behavior.

Web is the collaborative work of many people; a few publishing, and all of them querying and retrieving the information. Search engines record these activities in Web logs called the click-through data and reflect the query and clicks activities of users. Click-through data is more or less in the format shown in the Table 1 below:

R. Meersman et al. (Eds.): OTM 2012, Part I, LNCS 7565, pp. 323–341, 2012.

Table 1. Sample click-through data

AnonID	Query	Query Time	Item Rank	Click URL
7	Easter	2006-03-01 23:19:52	1	http://www.happy-easter.com
7	Easter eggs	2006-03-01 23:19:58	1	http://www.eeggs.com

To briefly explain the fields in Table 1, we begin with AnonID, the anonymous User ID from whom the search engine received the request, followed by the query issued by the user, the time the search engine received the request, the rank of the page item clicked, the page clicked in response to the result among the set returned by the search engine. Note that the click-through data format varies slightly from one search engine to the other.

Three Web data types identified in previous [2] efforts are: content (text and multimedia), structure (links that form a graph) and Web usage (transactions from Web log). Web data mining encompasses a broad range of research topics like improving page ranking, efficient indexing, query clustering, query similarity, query suggestions, extracting semantic relations and event detection etc. All these areas are inter-related and many use the click-through data as a starting point for their analysis. The seamless flow of advancement in developing better approaches in individual areas can be pipelined to improve existing techniques in other inter-related fields. Our effort in this paper is to integrate three Web data types and achieve the overall objective of event detection via query clustering.

1.1 Motivation

The dynamics of click-through data was previously identified in [3]. The frequency of queries and page clicks grow very fast when the real-world event approaches and become weaker gradually after. The co-occurrence of a query-page pair in a given window period is the number of times the pair appear together in the same row of Table 1 in that window period. The dynamics of co-occurrences can sense the arrival and pass over of the events. The work done by Greg [17] et al. gave an inside out perspective about the query space, query sessions, user behavior and content space. Interesting facts were revealed like about 28% of all queries are reformulations of previous queries; an average query is reformulated 2.6 times; users formulate and reformulate a series of queries in pursuit of a single task. The possibilities are new queries, add/remove word(s) to/from queries, change word(s) in a query, get more results for the same query, return to a previous queries etc. Our motivation is to cluster such similar queries with similar evolutionary pattern corresponding to real world events.

Our work differs from the existing work in one or more of the following ways:

(1) We consider the click-through data as collaborative query sessions (a time interval which consists of all the query-page pairs from all users within this time interval) rather than collection of individual entries of query-page pairs considered in [3,4]. A query session captures a series of user interactions with the search engine

with begin and end time period. The advantage of this approach is that in most of the meaningful sessions users issue a series of related queries and click through the web pages in the result set. They are semantically and temporally related to one another. These meaningful query sessions as initial clusters can correspond to real world events. User intensions are better understood by considering click-through data as query sessions. Search engines click-through data is massive and the graphs generated from the click-through data are overwhelmingly large. By considering click-through data as collaborative query sessions, we can substantially reduce the complexity of the problem.

(2) As we see, not every entry in the click-through data corresponds to real-world events. Navigational queries account for 21% of the total query frequency [17] so pruning irrelevant data can prepare a better ground for the approach. As an example, in one sample of data, we found the frequency of queries and page clicks of popular portal pages shown below in Table 2.

The frequencies are high but they really do not correspond to any real-world event. So in the data cleaning, preparation and transformation phases of the web data mining, we incorporate filtering methods to process the data. This step significantly improved the quality of the results.

Table 2. Frequent query-page pairs of popular portals

Query	Click URL	Frequency
Google	http://www.google.com	14236
Yahoo	http://www.yahoo.com	181820
Aol	http://www.aol.com	4774
Myspace	http://www.myspace.com	17104
Ask.com	http://www.ask.com	2213

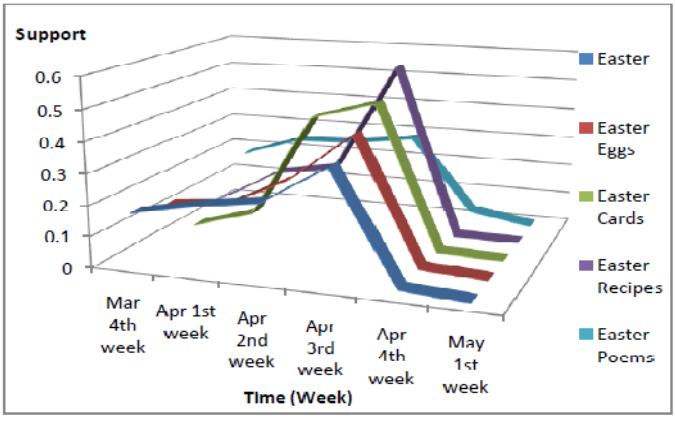

Fig. 1. Demonstration of query-page pair dynamics for “Easter” over a six week period

(3) We achieve the overall objective of event detections via query clustering. Event detection process ends with clusters of query-page pairs that are semantically and temporally related and corresponds to one or more events. We begin this process with queries because the number of queries the search engine receives (number of ways in which real-times queries are framed) are far less than the size of the Web i.e. Q<<P. By this obvious fact, we believe that clustering can be done efficiently if we begin the process with Q. Also, query keywords give some insight about the events. Queries can be formulated in several ways in different contexts, although they all mean the same and correspond to the same event. For example, Figure 1 shows the support of query-pairs {"Easter", www.happy-easter.com}, {"Easter Egg", www.eeggs.com}, {"Easter Cards", www.easter-cards.com}, {"Easter Recipes", www.easter-recipes.com} and {"Easter Poems", www.poemsforfree.com}. All the four query-page pairs have similar evolutionary pattern in the time window and correspond to the same event "Easter" on April 16, 2006. As one can observe, the support increased gradually up to the 3rd week of April and then decreased gradually. By early detection of this kind of query clusters, event detection can be done efficiently. Therefore, we incorporate query clustering into the event detection framework.

2 Related Work

In this section, we review some significant work in the literature on event detection and query clustering. The beginning of event detection originates from the initial work done on (TDT) Topic Detection and Tracking [11] to automatically detect topically related stories within a stream of news media. The objective of the work done on retrospective and on-line detection [12] is to detect stories based on two tasks: retrospective detection and online detection. The retrospective detection aims to discover previously unidentified events in accumulated collection while the on-line detection tries to identify the on-set of news events from live news feeds in real-time. An attempt on burst event detection was done by Fungs et al. [13] from chronologically ordered documents as text streams. They proposed a parameter-free probabilistic approach called feature-pivot clustering to fully utilize the time information to determine set of bursty features in different time windows. The work done by Zhao et al. [16] introduced the dynamic behavior idea to cluster web access sequences (WASs), based on their evolutionary patterns of support counts. The intuition is that often WASs are event/task- driven and partitioning WASs into clusters result in grouping of similar/closer WASs. Later their work in [3] laid to the foundation for visitor-centric approach to detect events by using click-through data. The query-page relationship is represented as the vector-based graph. On the dual graph of a vector-based graph, a two-phased graph cut algorithm is used to partition the dual graph based on (i) semantic-based similarity and (ii) evolution pattern-based similarity to generate query-page pairs that are related to events. However, their work does not perform any data pruning and have query times the number of pages in the space. Later, another approach was introduced by Chen et al. [4] by transforming the click-through data to the 2D polar space by considering the semantic and temporal dimensions of the queries. It then performs a subspace estimation to detect subspaces such that each subspace

corresponds to queries of similar semantics, thus it complicates the solutions by doing first subspace estimation and then the pruning of uninteresting spaces.

The query clustering work by Wen et al. [7] is on the Encarta encyclopedia. Their approach was based on the two principles: (1) if users clicked on the same documents for different queries then the queries are similar. (2) If a set of documents are often selected for a set of queries then the terms in these documents are related to the terms of the queries to some extent. In the effort of extracting semantic relations from query logs, Baeza-Yates et al. [8] proposed a model to project queries in a vector space and deduced some interesting properties in large graphs.

3 Event Detection Framework

The overview of our proposed event-detection framework is shown in Figure 2 and is briefly explained in this section. Given the click-through data, we perform the data cleaning, preprocessing and transformation tasks to refine the data. As shown in Table-2, some portion of the click-through data does not correspond to real-world events (like searching Google, Yahoo as keywords, or some pornographic keywords). The percentage of irrelevant data is highly random depending upon the sample chosen. Filtering this noise is a better step to prepare ground for further process. In order to analyze the dynamics of increase and decrease of co-occurrences of query-page pairs, we partition the click-through data in a sequence of collections based on user-defined time granularity. Time granularities can be like a day, week, month etc. Different time granularities are required to detect events over moving window sizes. Each collection can be represented by a bipartite graph. We summarize the co-occurrences of query-page pairs from all the collections into a summarized bipartite graph. Next, we transform the problem of event detection into query clustering while capturing the relationship among queries and pages. For this purpose, we use the hybrid cover graph [8] and employ a distance-based function that includes the semantics of the query and pages to define the criteria for clustering. The summarized support from bipartite graph (i.e, edges in hybrid cover graph, see Figure 2 and Figure 4) is used to emphasize the dynamics of the queries and pages in the clusters to detect an event.

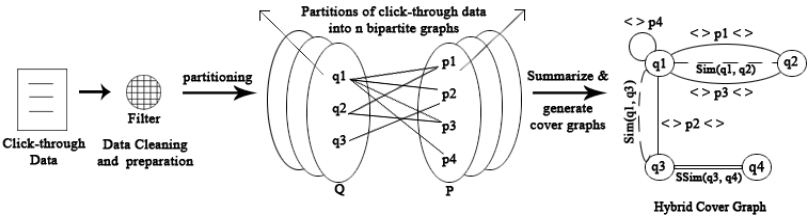

Fig. 2. Event detection framework overview

4 Data Representation

Click-through data is collected in Web logs. As mentioned earlier, we consider the click-through data as collaborative query sessions instead of individual query-page records. The reason for the same is explained earlier in Section 1.1. As informally defined earlier, a query session is wrapped by time boundaries; the beginning and the end time. We segment users' streams into sessions based on anonymous ID. Another widely used technique [14] is based on the idea that two consecutive actions (either query or click) are segmented into two sessions if the time interval between them exceeds 30 minutes.

Definition 1: (Query session) A query session S= (Q, P), where Q={q_1, q_2…q_m} is a bag of queries issued through the search engine and P = {p_1, p_2….p_n} is the set of corresponding pages clicked by the user from the search result set, where n is not m.

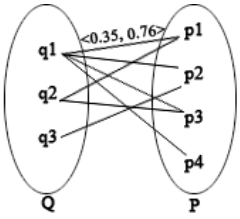

Fig. 3. Summarized bipartite graph

A Bipartite graph, G = (V, E) where nodes in V represent queries and Web pages and edges in E represent the strengths of the query-page pairs. Bipartite graphs are widely used in the Web data mining domain [5, 6] to represent the relationship between queries and pages. An edge between a query and a page is formed if the page is clicked in response to the query. Bipartite graphs can be visualized as mapping between the query set (Q) and the page set (P) as shown in Figure 3. We like in [3] partition the click-through data C into sequence of collections <C_1, C_2... C_n> based on user-defined time granularity like hour, day, week and month etc.

Definition 2: (**Strength**) of a query-page pair $P_{s,t}$ = (q_s, p_t) in collection C_i is $S_i(P_{s,t})$ = $\frac{|Ps,t(Ci)|}{\sum_{i=1}^{n}|Ps,t(Ci)|}$, where 1 ≤ i ≤ n and (s, t) is a query-page pair. Strength is the ratio of the co-occurrence of the query-page pair in collection C_i to C . This ratio so defined keeps the value ≤ 1 and is easy to process rather than showing high co-occurrence values withour normalization. Note that in Figure 3 the strength of (q1, p1) is summarized as <0.35, 0.76> for two collections. Noisy query-page pairs that appear sporadically and potentially not related to any event have very low strengths.

In order to cluster queries with pages clicked, we need the efficient data structure for their representation. Several graph algorithms are in existence which can be used for this purpose. For example, Baza-Yates et al. [15] identified several types of query graphs. In all such graphs, queries are nodes and an edge is drawn between two nodes

if (i) the queries contain the same word(s) – word graph (ii) the queries belong to the same session – session graph (iii) users clicked on the same URLs from the result sets – cover graph. Word graph is hard to use because users formulate queries in different ways but it is essential to capture the query semantics. Not all the queries from a session correspond to some event so session graph is not the option. Both word and session graphs fail to capture the semantics of pages clicked. Cover graph can be efficient because for two queries with a commonly clicked page, the edge is represented only once. Reducing the complexity of the graph structure with emphasis on page clicks can simplify the problem and helps in easy representation. We extend the notion of cover graph to hybrid cover graph, which is explained later. The notion of commonly clicked documents [15] is as follows:

Definition 3: Query Instance is a query (set of words or sentences) plus zero or more clicks related to that query. Formally: QI = (q, u*) where q = {words or phrase}, q being the query, u a clicked URL and the query instance of query q is denoted by QI_q and $QI_{c(u)}$ denotes the set of its clicked URLs.

Definition 4: URL Cover is the set of all URLs clicked for a query. So for the query q, $\cup C_p = \cup_{QIqp} QIc(u)$.

The nodes in the hybrid cover graph are queries from the click-through data. Three types of edges are possible between any two nodes: 1. Cover edge (represented by normal line) is drawn if a page is clicked in common to both the queries 2. Similarity edge (represented by dotted line) represents the similarity of the two queries; page content and user click feedback. 3. Session similarity edge (represented by double line ==) is drawn if two queries are related to each other as a result of the association rule mining of most of the sessions, referred as session inference. The criterion for similarity over the similarity edge is based on distance function and session inferences.

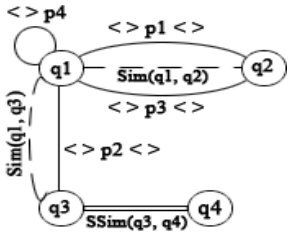

Fig. 4. Hybrid Cover Graph

The hybrid cover graph shown in Figure 4 is formed by incorporating the features of word and session graphs into the cover graph. Sim(q1, q3) is the similarity edge that represents the similarity between the queries q1 and q3, which have common URLs clicked in response to them. The vectors on each side of the page p2, represented as <>p2<> indicate the summarized support of p2 with the corresponding query nodes. SSim(q3, q4) is the session similarity between q3 and q4, which will be explained in Section 5.

5 Distance Function

Similarity between two queries correspond to nodes in a graph is based on our approach to integrate the semantics, structure, and content of queries and pages. Our distance criterion is based on work done by Wen et al. [7] to cluster queries.

5.1 Similarity Based on Query Contents

Although short queries are harder to understand, queries are better understood by considering them as keywords, words order and phrases. We perform stemming, stop words elimination, phrase recognition and synonym labeling while adding a query to the query semantics dictionary of a cluster. Let p, q be two queries.

Similarity based on Keywords or Phrases

$Sim_{keyword}$ (p, q) = KN (p, q)/Max (kn (p), kn (q))

where KN (p, q) = the number of common keywords in the queries p and q, kn (p) = number of keywords in p.

Similarity based on String Matching

The comparison is the string-matching problem and can be computed by edit distance, i.e. number of edit operations required to unify two strings.

For letters and characters, Sim_{edit} (p, q) = EditDistance (p, q)

$Similarity_{content} = Sim_{keyword} / Sim_{edit}$ ($Sim_{keyword}$ is based on the keywords)

5.2 Similarity Based on Session Feedback

A query can be expressed as a point in high-dimensional space [15] where each dimension corresponds to a unique URL i.e. a query can be given a vector representation based on all the different URLs in its cover. A weighted representation that takes document frequencies into account is used. If p and q are two queries then Sim_{vector} (from the author-centric point of view) is computed as:

$$Sim_{vector} = \frac{\overline{p}}{|p|} . \frac{\overline{q}}{|q|}$$

Session feedbacks from meaningful query sessions can help to relate topically similar URLs. A simple way to take user feedbacks into consideration is by taking the normalized value to see the similarity in terms of the commonly clicked URLs for the queries. Sim_{doc} represents visitor-centric (generated by users browsing activity) point of view and

Sim_{doc}= RD (p, q) / Max (|Cover (p)|, Cover (q)|) where RD (p, q) is the number of commonly clicked URLs and |Cover (p)| is the number of URLs clicked for query p.

Sim_{vector} is presented from the author-centric (generated by publishers) point of view, whereas Sim_{doc} is from visitor-centric point of view. This tells how users have chosen to click on these pages.

$Similarity_{feedback} = Sim_{vector} * Sim_{doc}$

Content-based measures tend to cluster queries with the same or similar terms whereas session feedback-based measures tend to cluster page clicks related to the same or similar topics.

Similarity (p, q) = α $\text{Similarity}_{\text{content}}$ + (1- α) * $\text{Similarity}_{\text{feedback}}$ where α is the weight factor.

Distance (p, q) = 1 / Similarity (p, q)

Larger the similarity, smaller the distance and the weights for content and session feedback similarities are adjusted to obtain better results. An edge between two queries p and q in the hybrid cover graph is drawn if Distance (p, q) ≤ D_{min}, where D_{min} is the minimum distance.

Association Rules [9] can be applied to find queries that are asked together in many of the query sessions. In the problem of finding related queries from query set Q, we are interested in associations like X=>Y, where X, Y are subsets of Q, X ∩Y=Ø. The rule X=>Y should have a support ≥ S_{min} and confidence ≥ C_{min}, where S_{min} and C_{min} are minimum support and confidence values. Suppose the rule q1=> q4 | given S and C where S ≥ S_{min} and C ≥ C_{min} is found then include the rule in the hybrid cover graph.

6 Clustering Process

For each query q ∈ Q, find the clusters (for first query, there is no cluster to compare so it forms its own cluster, for subsequent queries, find distances among the clusters obtained so far) with which the minimum distance condition is satisfied. Assign q to those clusters. Two queries q1 and q2 fall into the same cluster if the distance between q1 and q2, D(q1,q2)<=Dmax. If the threshold distance condition is not satisfied with any of the existing clusters then start a new cluster beginning with q.

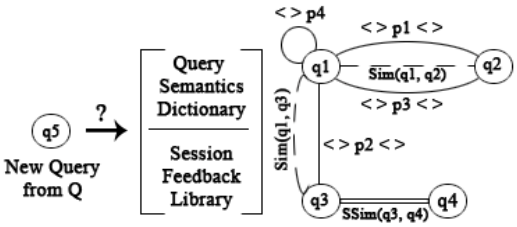

Fig. 5. Clustering Process

For example, as shown in Figure 5, when a new query q5 comes in, its content is compared with the semantics of the query dictionary formed from existing queries - q1, q2, q3, q4. Then its page clicks from the summarized bipartite graph are compared with the session feedback library of all the pages - p1, p2, p3, p4 for a given cluster. If the distance D is ≤ D_{min} then the query is added to the cluster, the query semantics are added to the query semantics dictionary and its page clicks are added to the session feedback library. If not, the query begins forming a new cluster.

6.1 Event Detection Algorithm

There are several challenges in designing a query clustering technique. It should be able to handle all types of attributes, scalable on massive datasets, work with high dimensional data, handle outliers, have reasonable time complexity, be independent of data order, and start without initial parameters (for example, the number of clusters). DBSCAN [10] algorithm and its incremental version meet all the required conditions and its average time complexity is O (n*log n).

```
Input: Set of Queries Q
Output: Set of Clusters Θ
Initialization: Θ = Ø, cluster=0;
ClusteringAlg(Q)
Begin
For each query qϵQ do
  If cluster!=0  // clusters existing
    then
      For each existing Cluster Ci
        For each query qj ϵ Ci
            if distance(q, qj) ≤ Eps
             Then Ci = Ci U q  // add to cluster
                  Θ = Θ U Ci // update cluster set
            End if
        End for
      End for
  End if
  Else
      // no clusters are existing condition
      Then Cnew U = q   // form a new cluster
           Θ = Θ U Ci   // add to cluster set
           cluster++;   // cluster count tracked
  End Else
End for
GenerateEventSub-graphs (Θ)   // pass cluster set
End
```

Algorithm 1. ECO – Clustering Process

```
Input: Set of Clusters Θ
Output: Set of hybrid cover graphs

GenerateEventSub-graphs(Θ)
Begin
    For each Cluster Ci ϵ Θ
        For all Qi ϵ Ci
            DrawCoverGraph();
            Check-Comprehensive-Reachable();
            Check-Comprehensive-Connected();
        End for
    End for
End
```

Algorithm 2. Event Detection ECO – Hybrid Cover Graph

Our algorithm though inspired by the DBSCAN differs significantly from it. First, the function in our approach is not density-based but distance-based and second, we require only one scan of the queries through the click-through data. The criterion for distance function is explained previously in Section 5. The event detection algorithm is presented in two steps. Algorithm1 is for the clustering process and the later one is for generating the hybrid cover graphs. The hybrid cover graphs are drawn with respect to the comprehensive-reachable and comprehensive connected conditions of the DBSCAN algorithm for the terminal nodes. The algorithm runs at different time granularities to detect events of different window size like day, week and month etc.

The summarized support values for the query-page pairs are analyzed using histograms to ensure that the hybrid cover graph has an evolutionary pattern. The higher ranking of nodes in the hybrid cover graph can be given for the connected dominating set (nodes that essentially connect the graph), nodes with least distance and with higher supports with their corresponding edges. The page rank of the edge can be obtained as the ItemRank from the click-through data. The edges with better ranks can be regarded as high quality Web pages clicked in relation to events.

Pruning irrelevant data is very important because the click-through data has millions of queries and pages. We reduced the size of the graph qualitatively and quantitatively by eliminating: 1. Queries and pages that have low support values. By doing so, some edges and nodes can be removed from the graph. These queries and pages can be seen sporadically in the data. 2. Multi-topical URLs (pages that talk about several topics or a very generic topic) by removing edges of low weight obtained from criteria in Section 5. Low weight edges are more likely to represent poor quality semantic relations.

7 Working Example

In this section, we explain the overall process by continuing the example initiated in Section 1.1. Figure 1 shows the support of query-pairs P1 {“Easter”, www.happy-easter.com}, P2 {“Easter Egg”, www.eeggs.com}, P3 {“Easter Cards”, www.easter-cards.com}, P4 {“Easter Recipes”, www.easter-recipes.com} and P5 {“Easter Poems”, www.poemsforfree.com}. The co-occurrence, support for the query page pairs for the 6 week window period is shown in Tables 3 and 4. We show the similarity computation for the queries “Easter” and “Easter Eggs”.

Table 3. Co-occurrence of query-page pairs over a 6 week window period

	31-March	7-April	14-April	21-April	28-April	04-May
P1	7000	8700	9900	1510	600	0
P2	9200	10500	16900	2740	1000	200
P3	300	1500	8200	9300	100	0
P4	1000	2900	3500	6900	0	0
P5	9100	8300	8500	9500	1200	0

Table 4. Support of query-page pairs over a 6 week window period

	31-March	7-April	14-April	21-April	28-April	04-May
P1	0.169	0.210	0.239	0.365	0.014	0
P2	0.141	0.161	0.259	0.420	0.015	0
P3	0.015	0.077	0.422	0.479	0.005	0
P4	0.058	0.170	0.205	0.564	0	0
P5	0.181	0.247	0.252	0.282	0.035	0

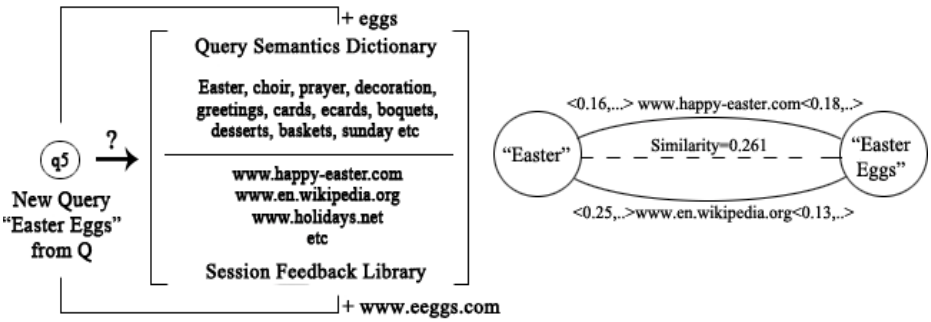

Fig. 6. Illustration of "Easter" and "Easter Eggs" clustering

$Sim_{keyword}$ =1/2=0.5; Sim_{edit}=4, $Similarity_{content} = Sim_{keyword} / Sim_{edit}$=0.125
We computed Sim_{vector}= 1.2, Sim_{doc}= 177/569=0.311,
$Similarity_{feedback} = Sim_{vector} * Sim_{doc}$= 0.373
Similarity ("Easter", "Easter Eggs") = α $Similarity_{content}$ + (1- α) * $Similarity_{feedback}$, where α is the weight factor and assume α=0.45.
Similarity ("Easter", "Easter Eggs") = 0.261,
Distance (p, q) = 1 / Similarity (p, q)

Let Distance = 1/0.261=3.83. Assume D_{min}=3 then the queries "Easter" and "Easter Eggs" should fall into the same cluster. The process is illustrated in Figure 6. Note that only the portion of hybrid cover graph with nodes "Easter" and "Easter Eggs" is shown because of the complexity of the graph. All the four query-page pairs are semantically and temporally related and have similar evolutionary patterns in the window period and corresponding to the same event "Easter" on Aril 16, 2006. As one can observe, the support increased gradually in the 3rd week of April and then decreased gradually. The criterion for distance function is explained in Section 5 and the clustering process is explained in Section 6.

8 Performance Study

In this section, we study the performance of our event detection approach. First, we describe the characteristics of the dataset used in experiments. Then we present the experimental results and their comparison with some of the existing work.

8.1 Data Set

A real click-through dataset obtained from AOL search engine is used in our experiments. The data is from March 2006 to May 2006, comprised of 500k query sessions, consisting ~20 web million queries and click-through activities from 650k users. As described in [17], each line in the data represents one of two types of activities: (i) a query that was not followed by the use clicking on a result item. (ii) a click through on an item in the result list returned from a query. In the later case, the pages appear

as successive entries in the data. In our approach, as a query session is obtained as successive pages corresponding to the same query from the same user. The timestamp of the first page click in a query session is taken as the start time of the session.

8.2 Result Analysis

Our approach can also detect pre and post period events, where the current period is referred to March through May, 2006. As discussed in Section 1.2 the co-occurrence of query-page pairs corresponding to an event does not stop abruptly right after the event but slows down at a faster rate. So pre and post period events can be detected by analyzing such kind of a behavior. For example pre-period event "Winter Olympics Torino, Italy" happened during February 10 through 26. We observed significant interest decreasing at a faster rate in regard to this in early March data. Post-period event "FIFA World Cup, Germany" during June 9 through July 9 is detected with increasing interest at the end of the May data.

Our algorithm can detect events of different time granularity like day, week and month. For an event, the traffic spreads around the event juncture like few days, weeks, and months in time granularity before and after the event. Day events like the death of Jack Wild, a famous British actor on March 1, the St. Patrick's Day on March 17 etc. are detected. Week events like the Philadelphia flower show, (a big indoor flower show) during the week March 5 through 12, the Fleet week (public can see USA Navy and Coast guard ships) during the week May 24 through 30 etc. Monthly events span across bigger time frames and appeared throughout the data. The famous American Idol 5 episode appeared March 1 through May 24 (finale), the Internal Revenue Service (IRS) tax filing appeared March 1 through 31. Note that some of the events are regular and previously known like the St. Patrick's Day; Good Friday etc. which recur every year. Some are previously unknown; like Simon Lindley, an Organist received the "Coveted Spirit of Leeds" award on May 3, the release of the movie V for Vendetta on March 17 etc. These events are not periodic and do not recur. Our approach could detect both types of events and of different time granularities. Our approach detected a lot of events that are not recognized previously by the existing work [3, 4] on the same dataset. Due to space restrictions we are not able to include the full list of events detected. The complete list of events detected is shown in Appendix.

8.3 Experimental Analysis

DECK [4] outperformed two-phase-clustering algorithm [3] so we compare the performance of ECO with the DECK, DECK-NP [4] and DECK-GPCA [4] on the same dataset. Number of events detected is a simple way to compare different approaches. ECO could detect 96 events where as DECK detected only 35 events. ECO could not detect 5 events in the list of 35 events detected by the DECK. On the other hand, DECK did not detect 61 events that ECO could detect. On time granularity comparison, ECO could detect 80 day events, 8 week events and 8 month events. In the events listed by DECK, 32 are day events, 3 are week events and no month events.

As mentioned earlier, our approach could detect 1 pre-period, 83 current period and 12 post period events. The experimental results are shown in Figures 7 below.

The evaluation metrics, precision, recall, F-measure (F-1 score) and entropy are used along with the number of events detected to compare the performance. Precision is the ratio of number of correctly detected events to the overall discovered clusters. Recall is the ratio of number of correctly detected events to the total number of events. F-measure is computed based on the precision and recall as the weighted harmonic mean of precision and recall. F-measure = 2 * precision * recall / (precision + recall). For each generated cluster i, we compute Pij as the fraction of query-page pairs (query sessions) representing the true event j. Then the entropy of the cluster i is $E_i = -\sum_j Pij \log Pij$. The total entropy can be calculated as the sum of the entropies of each cluster weighted by the size of each cluster: $E = \sum_i^m \frac{ni*Ei}{n}$, where m is the number of clusters, n is the total number of query-page pairs (query sessions) and ni is the size of the cluster i. The experimental results are shown in Figures 8. ECO did fairly well in terms of precision and recall for up to half of the data size. As the number of query sessions increased, the number of query patterns increased so the number of noisy query clusters increased which resulted in slight down fall in precision but not recall and increased in entropy.

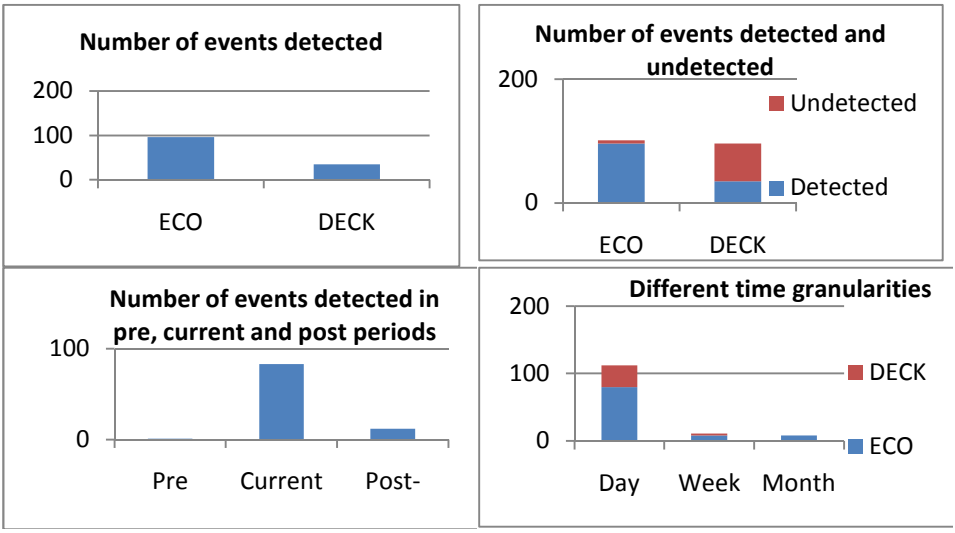

Fig. 7. Comparison of ECO with DECK on number of events detected

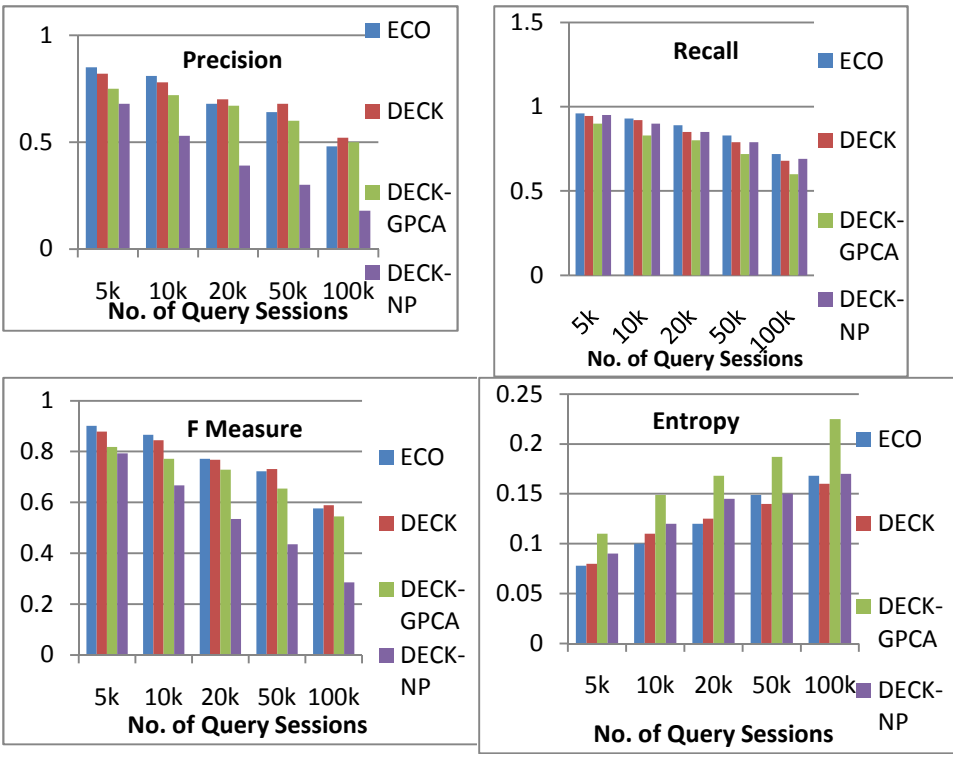

Fig. 8. Precision, recall, F-measure and entropy of ECO and DECK

8.4 Effect of α

The factor α decides the weights for content-based similarity and feedback-based similarity. We ran experiments varying the value of α, which is shown in Figure 9. The number of events detected varied accordingly. At α=0.15, 31 events are detected. As the weight for feedback-based similarity increased we started identifying new clusters of events. At α=0.45 we got the best results in terms of events detected. As the weight for feedback-based similarity increased further, the performance degraded.

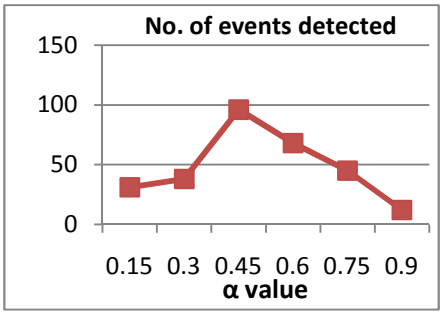

Fig. 9. Impact of α on Event Detection

9 Conclusions

In this paper, we proposed an approach called ECO for detecting events from the click-through data. Firstly we performed data cleaning, transformation and preparation process to filter the noise and then partitioned the click through data into collections of user defined granularity. Then we transformed the problem into query clustering, simultaneously trying to integrate the content, structure and semantics of the queries and clicked URLs. We introduced the hybrid cover graph to efficiently represent the clusters of query- page pairs. The evolutionary pattern of the query-page pairs is embedded into the hybrid cover graph as vectors over the edges to incorporate the dynamics. Our results outperform the existing work [3,4] in terms of the number of detected events, entropy measure, F-measure and recall.

References

[1] De Kunder, M.: The size of the World Wide Web. World Wide Web Size (September 04, 2009), http://www.worldwidewebsize.com
[2] Baeza-Yates, R.: Web Mining in Search Engines. In: Proceedings of the 27th Australasian Conference on Computer Science, New Zealand, vol. 26 (2004)
[3] Zhao, Q., Liu, T.-Y., Bhowmick, S., Ma, W.-Y.: Event Detection from Evolution of Click-through Data. In: Proceedings of KDD, Philadelphia, PA, USA (2006)
[4] Chen, L., Hu, Y., Nejdl, W.: DECK: Detecting Events from Web Click-Through Data. In: Eighth IEEE International Conference on Data Mining (ICDM), pp. 123–132 (2008)
[5] Beeferman, D., Berger, A.: Agglomerative clustering of a search engine query log. In: SIGKDD (2000)
[6] Xue, G.-R., Zeng, H.-J., Chen, Z., Yu, Y., Ma, W.-Y., Xi, W., Fan, W.: Optimizing web search using web click-through data. In: ACM Proceedings of CIKM, pp. 118–126 (2004)
[7] Wen, J., Mie, J., Zhang, H.: Clustering user queries of a search engine. In: Proceedings of the 10th International World Wide Web Conference (2001)
[8] Baeza-Yates, R., Tiberi, A.: Extracting Semantic Relations from Query Logs. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 76–85 (2007)
[9] Federal, B.F., Fonseca, B.M., De Moura, E.S.: Using Association Rules to Discover Search Engines Related Queries. In: Proceedings of the 1st Conf. on Latin American Web Congress (2003)
[10] Ester, M., Kriegel, H.-P., Jörg, S., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: 2nd International Conference on Knowledge Discovery, pp. 226–231 (1996)
[11] Allan, J., Rapka, R., Lavarenko, V.: On-line New Event Detection and Tracking. In: SIGIR (1998)
[12] Yang, Y., Pierce, T., Carbonell, J.G.: A Study of Retrospective and On-line Event Detection. In: SIGIR 1998 (1998)
[13] Fung, G.P., Yu, J.X., Yu, P.S., Lu, H.: Parameter Free Bursty Events Detection in Text Streams. In: Proceedings of VLDB (2005)
[14] White, R.W., Drucker, S.M.: Investigating Behavioral Variability in Web search. In: Proceedings of WWW, pp. 21–30 (2007)

[15] Baeza-Yates, R.: Graphs from Search Engine Queries. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 1–8. Springer, Heidelberg (2007)
[16] Zhao, Q., Bhowmick, S.S., Gruenwald, L.: CLEOPATRA: Evolutionary Pattern-Based Clustering of Web Usage Data. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 323–333. Springer, Heidelberg (2006)
[17] Pass, G., Chowdhury, A., Torgeson, C.: A Picture of Search. In: the First ACM International Conference on Scalable Information Systems, Hong Kong (2006)

Appendix: List of Events Detected

Event	Time-stamp
Pre-period events	
Winter Olympics (Torino 2006)	February 26th
Current-period events	
Ash Wednesday	March 1st
Jack Wild died	March 1st
World Baseball Classic	March 3rd-20th
48th Annual Heard Museum Fair	March 4th, 5th
78th Academy Awards	March 5th
Triple Six Mafia won Academy Award	March 5th
Philadelphia flower show	March 5th-12th
Dubai Tennis Open ends	March 6th
Big 12 Women's Basketball Championship	March 7th-12th
Big Ten Conference Men's Basketball Tournament	March 9th-12th
NCAA men's Division I Basketball Tournament	March 14th-April 3rd
Ides of March	March 15th
John West salmon commercial	March 15th
Ram Bahdur Bomjon disappeared	March 16th
V for Vendetta movie released	March 17th
Saint Patrick's day	March 17th
NCAA Women's Division I Basketball Tournament	March 18th-April 4th
Los Angeles Marathon	March 19th
Washington D.C. Cherry Blossom Festival	March 25th
27th Annual Young Artist Awards	March 25th
Buck Owens died	March 25th
Rocio Durcal died	March 25th
Bataan Memorial Death March	March 26th
Indy racing league season started	March 26th
Solar eclipse in North Africa	March 29th
Basic Instinct 2 movie released	March 31st
April fool's day	April 1st
Liberty Bell Classic	April 2nd
140th anniversary of Baptist Union Baptist Church	April 2nd
Good Friday	April 14th
Scary movie 4 released	April 14th
Easter	April 16th

Boston Marathon	April 17th
Stanley Cup Playoffs	April 21st
Launch of lucky lines by Oregon Lottery	April 23rd
Italian Social Republic	April 25th
Dolphins Massacre at Zanzibar	April 28th
Steve Howe died	April 28th
The 33rd Annual Day-time Emmy Awards	April 28th
Pleasant valley baseball tournament	April 29th
The Hobbit movie started	April 31st
27th Sports Emmy Awards	May 1st
David Blaine perfor-mance at Lincoln Center	May 1st
Brooklyn Academy added to NHRP	May 2nd
10000 days album re-lease	May 2nd
Simon Lindley received "Coveted Spirit of Leeds" award	May 3rd
National Teachers day	May 4th
Advanced Placement Test	May 1st-10th
Cindo de Mayo	May 5th
Men's World Ice Hockey Championship	May 5th-21st
132nd Kentucky Derby	May 6th
29th Annual Five Boro Bike Tour	May 7th
Fort Collins Old Town Marathon	May 7th
Chris Daughtry eliminat-ed from American Idol 5	May 10th
Alligator attacks	May 14th
Mother's day	May 14th
Tony Awards nomi-nations	May 16th
The Amazing Race finale	May 17th
Penny saved 1000$ worth	May 17th
Cannes Film Festival	May 17th-28th
Big Island Film Festival	May 18th-21st
The Davinci Code movie release	May 19th
82nd Air Borne Division show	May 20th
NASCAR Sprint All-Star Challenge	May 20th
Strawberry Festival	May 21st, 22nd
10.5 Apocalypse Movie release	May 21st
41st Annual Country Music Awards	May 23rd
American Idol 5 ends	May 24th
Fleet week	May 24th-30th
Africa day	May 25th
31st Annual Million Dollar Beauty Ball	May 26th
Ultimate Fighting Championship 60: Hughes vs. Gracie	May 27th
The 90th Indianapolis 500	May 28th
Memorial day	May 29th
Post-period events	
The Omen movie release	June 6th
06/06/06 Doomsday	June 6th
FIFA World Cup (Ger-many)	June 9th
National Golden glove boxing championship	June 9th-13th
60th Annual Tony Awards	June 11th
Juneteenth Day	June 17th
Antique car show in Alabama	June 20th

USA Outdoor Track and Field Championships	June 21st-25th
Air shows New England	June 24th, 25th
Ann Arbor art fair	July 19th-21st
58th Annual Primetime Emmy Awards	August 27th
Albuquerque Baloon Festival	October 6th-15th
Month events	
NBA Basketball playoff	March, April
The Shoe show series aired on Resonance FM	March, April, May
American Idol	March, April,
	May
Annual walleye run in Fremont Ohio	March, April, May
IRS tax filing	March, April
Greenland ice melt by 250%	March, April
College Student Survey	March, April
1199 home care worker pay increase negotiations	March, April
Business Opportunities	
Summer - restaurants, resorts, cruises, islands etc	April, May