Learning to Detect Event-Related Queries for Web Search

Nattiya Kanhabua
L3S Research Center
Leibniz Universität Hannover
Hannover, Germany
kanhabua@L3S.de

Tu Ngoc Nguyen
L3S Research Center
Leibniz Universität Hannover
Hannover, Germany
tunguyen@L3S.de

Wolfgang Nejdl
L3S Research Center
Leibniz Universität Hannover
Hannover, Germany
nejdl@L3S.de

ABSTRACT

In many cases, a user turns to search engines to find information about real-world situations, namely, political elections, sport competitions, or natural disasters. Such temporal querying behavior can be observed through a significant number of event-related queries generated in web search. In this paper, we study the task of detecting event-related queries, which is the first step for understanding temporal query intent and enabling different temporal search applications, e.g., time-aware query auto-completion, temporal ranking, and result diversification. We propose a two-step approach to detecting events from query logs. We first identify a set of event candidates by considering both implicit and explicit temporal information needs. The next step further classifies the candidates into two main categories, namely, event or non-event. In more detail, we leverage different machine learning techniques for query classification, which are trained using the feature set composed of time series features from signal processing, along with features derived from click-through information, and standard statistical features. In order to evaluate our proposed approach, we conduct an experiment using two real-world query logs with manually annotated relevance assessments for 837 events. To this end, we provide a large set of event-related queries made available for fostering research on this challenging task.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Query Intent; Temporal Query Classification; Events

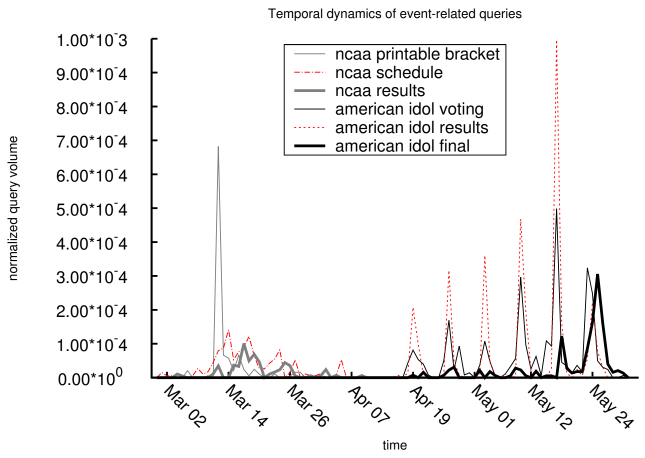

Figure 1: Temporal querying behavior of two real-world events: (1) ncaa tournament, and (2) american idol.

1. INTRODUCTION

People are naturally interested in searching about high-impact events like political elections, major sports competitions, aviation accidents or earthquakes, in order to keep track of ongoing discussions, or get updated about an anticipated event or an aftermath following the previous incident. This can be regarded as a collective attention or a crowd phenomenon observable through a significant number of queries generated around such happenings. In Figure 1, we illustrate temporal querying behavior, or changes in the popularity of queries related to two real-world events. The first event is a NCAA tournament[1], a major sports competition in the United States held annually and organized by the National Collegiate Athletic Association (NCAA).

Another event is American Idol[2], which is an American reality-singing competition. For such a case, underlying information needs change over time, which reflect the temporal aspects or topics of an event. As depicted, relevant searches for a NCAA tournament are ncaa printable bracket, ncaa schedule, and ncaa results, whereas search queries related to American Idol are american idol voting, american idol results, and american idol final. On the one hand, a pre-event activity like tournament brackets triggers a burst in query streams around the beginning of the event on March 14, 2006. On the other hand, information about schedule and results gains increased interest only after the event has

[1] http://en.wikipedia.org/wiki/NCAA_tournament
[2] http://en.wikipedia.org/wiki/American_Idol

started, with less query popularity before that. The American Idol program was displayed on a weekly basis, thus public attentions about audience voting and results exhibits periodic patterns, while the final contest is surged in important on the last program shown on May 24, 2006.

Temporal web dynamics are related to user querying behavior in at least two ways. First, as shown in the above example, search traffic for particular queries varies over time and might present certain temporal patterns, such as, spikes, periodicity (e.g., hourly or daily), seasonality and trends. Second, many queries are time-sensitive queries, i.e., contain underlying temporal information needs that do not exhibit a temporal pattern in search streams. More precisely, event-related queries can be regarded as queries that contain temporal information needs, which are categorized into two main classes. The first class of temporal queries is observed through temporal patterns of user search behaviors in query logs, such as, burstiness, trend or seasonality [18]. The second class of event-related queries may not exhibit such temporal patterns in a query stream, but the underlying temporal information needs can be either (1) represented by temporal expressions explicitly mentioned in queries [17], or (2) implicitly associated with a particular time period with no temporal expressions given [11]. Examples of explicit temporal queries are US Election 2012 and 2014 FIFA World Cup, whereas queries without explicit temporal expressions can be, for instance, Easter, Thailand tsunami and Germany World Cup.

In this paper, we address the problem of identifying event-related queries in search streams. This problem is crucially important to understand temporal search intent and enable the development of advanced models and algorithms within the temporal search realm. There are existing works that are relevant in this context including: 1) studying event-driven search behavior in query logs, e.g., [10, 18], 2) temporal query classification, e.g., [9, 19, 2, 23], and 3) determining time for implicit temporal queries, e.g., [11, 3].

Our work differs from the aforementioned works to some extent. We consider *two main aspects* of event-related queries: (1) such queries can be observed through temporal querying patterns in query logs, and (2) they may not exhibit such temporal patterns in a query stream, but contain underlying temporal information needs. This is contrary to most of the previous works, which have studied only on a particular aspect, e.g., to study *either* implicit *or* explicit temporal information needs, separately. On the other hand, the Temporalia challenge [8] provides a test set for temporal query intent classification (TQIC) consisting of approximately 400 temporal queries categorized into predefined classes based on their implicit or explicit temporal intent. Therefore, many recent works participating in this challenging TQIC task, for instance, the work by Burghartz and Berberich [2], are also related to our work. However, this task focuses only on the later stage of our work that does not deal with a large amount of candidate queries in a data stream setting.

Our contributions can be summarized as follows.

- We propose a two-step approach for detecting event-related queries from public web search logs, namely, 1) event candidate identification, and 2) event-related query classification.
- We evaluate our proposed approaches using a large set of queries and make it available[3] for fostering further research on this challenging task.

[3]http://www.l3s.de/~kanhabua/data/EventQueries.zip

2. OVERVIEW

Data Model. Our data consists of the sets of queries Q, URLs U and click-through information. Each query $q \in Q$ contains query keywords or $term(q)$, the timestamps of query $time(q)$ or the time point when it was issued (so-called *hitting time*), and an anonymized ID of the web search user who submitted the query. A clicked URL $u \in U_q$ refers to a web document returned as an answer for a given query q, which has been clicked by the user. The click-through information is a transactional record per query for each clicked URL, i.e., an associated query q, a clicked URL u, the position in the ranked list of results, and its timestamps. In addition, we employ a *temporal document collection*, which contains documents that are temporally-ordered and it can be represented as $D = \{d_1, \ldots, d_n\}$. A document is regarded as bag-of-words (an unordered list of terms, or features): $d = \{w_1, \ldots, w_n\}$. $PubTime(d)$ is a function that gives the publication date of the document.

A real-world event of interests is a happening that involves or attracts public attention, e.g., ceremony, competition, convention, meeting, festival, etc. An event-related query q^e corresponding to a real-world event e, where the event time $time(e)$ indicating when an event e has actually taken place. The temporal aspect of the event e represented as $time(q, e)$ can be determined by considering $time(e)$ with respect to $time(q)$, which can refer to one of the time periods *before*, *during* and *after*.

Problem Statement. This works aims at detecting event-related queries in two steps: 1) automatically extracting a set of query candidates $\{q_1, \ldots, q_n\}$ from the query logs Q, and 2) classifying candidates as a set of event-related queries $\{q^e_1, \ldots, q^e_m\}$ using machine learning, which will be described in more detail in the next section.

3. OUR APPROACH

The first step is to identify a set of event candidates or queries related to real-world events. The next step is to classify the candidates into two categories, i.e., *event* or *non event*. For this purpose, we present different features for training a query classification model.

3.1 Identifying Event Candidates

Explicit temporal queries can be automatically extracted using a time and event recognition algorithm [12, 20]. We identify implicit temporal queries by looking for those exhibiting changes in query popularity or burstiness in query streams. Due to a short time span of the query dataset, we mine seasonality patterns using an external, temporal document collection. Our method for identifying implicit queries is based on keyword-based clustering (depicted in Algorithm 1) that is chosen rather than a technique relying on query-URL bipartite graphs because a graph-based processing of millions of queries can be infeasible [22]. The proposed keyword-based clustering is performed in two steps as follows.

Step 1. Given all distinct queries, we partition query log streams by a *weekly* granularity in order to roughly group queries from the same events together. Our intuition is that event-related queries are usually issued around the same time within a short period, e.g., days or a week. However, a time-based partition cluster often contain queries from multiple, unrelated events, so we further perform a clustering step based on lexical similarity in order to find a set of queries corresponding to an individual event.

Algorithm 1 *KeywordClusteringByTime(Q)*

```
1: INPUT: Set of real-world search queries $Q$
2: OUTPUT: Clusters of queries C
3: for each $\{Q_{partition} \in Q\}$ do
4:     $Q_{sort} \leftarrow sort(Q_{partition})$                // sort alphabetically
5:     $S \leftarrow set()$                                      // create empty set
6:     for each $\{q_i \in Q_{sort} \wedge freq(q_i) \geq \alpha \wedge freq(q_i) \leq \beta\}$ do
7:         $S.put(q_i)$
8:         if $JaccardSim(q_i, q_{i+1}) \geq \theta$ then
9:             $S.put(q_j)$
10:        else
11:            $C.add(S)$                        // add S to C as a query cluster
12:        end if
13:    end for
14: end for
15: return $C$
```

Step 2. Given the sets of queries grouped by time, we sort all queries alphabetically and identify a pair of sorted queries by searching for any two *adjacent* queries. For each pair, we apply stop-word removal and stemming, then compute their similarity using the Jaccard similarity coefficient. We assign each query pair into the same cluster if their similarity score is above a certain threshold, which will be determined empirically. The clustering process will be repeated until all query partitions are processed. Our final results contain a set of clusters, where each of them is corresponding to an event. An example cluster generated using our algorithm comprising a set of queries related to the event Easter are shown below.

> Easter - *easter 2006, easter 2007, easter 20crafts, easter activities, easter animation, easter animations, easter background, easter basket, easter bread, easter bucket, easter bunny, easter bunny decorations, easter bunny lights*

3.2 Event-Related Query Classification

Our method is based on a machine learning technique trained using a feature set aimed at detecting periodic, trending, sporadic (rare), and unseen events. It is composed of time series features from signal processing [1, 4, 7], along with features derived from click-through information [5], and standard statistical features [9, 13]. For a given query, we extract *daily* time series data from three data sources: 1) Q, the normalized query volume aggregated across all users over time, 2) U, the normalized click frequency for the query accumulated from all URLs and users, and 3) D, the temporal distribution of number of top-k search results retrieved from an external document collection. For each query candidate q issued at hitting time $time(q)$, we extract a set of time series data $\mathbb{Y} = \{Y_Q, Y_U, Y_D\}$ as input for our feature extraction process, where Y_i corresponds to data from the aforementioned sources. In the following, we present our feature set.

Seasonality is a temporal pattern that indicates how periodic is an observed behavior over time. Previous work [19] employs *seasonality* for detecting seasonal queries corresponding to events that repeat every year, e.g., **US Open**, **Halloween** and **Christmas**. However, we leverage this feature for detecting more *fine-grained periodic* events that recurring on a weekly basis, such as, a TV show program. In order to extract the seasonality component, time series data can be decomposed by different statistical techniques [4, 7], called a time-series decomposition process. Given a time series $Y = \{y_1, ..., y_N\}$ where N is the total number of observed values, we apply Holt-Winters adaptive exponential smoothing [7] to construct the statistical decomposition of time series input, which can be either the aggregated query volume or the distribution of top-k retrieved documents over time. The output consists of three components: *level* L, *trend* T and *seasonality* S. The trend represents the long-term direction of the time series. We use *trending scope* and *trending amplitude* as features for classification.

The seasonality component S is significantly important for estimating the seasonal characteristic of a given time series Y. The final score can be calculated using a similarity metric, i.e., the cosine similarity between the original time series Y to its derived seasonality component S as follows.

$$Seasonality(Y, S) = \frac{\vec{Y}\vec{S}}{\parallel \vec{Y} \parallel \cdot \parallel \vec{S} \parallel} \tag{1}$$

Autocorrelation is the cross correlation of a signal with itself or the correlation between its own past and future values at different times. We employ autocorrelation as an additional feature for detecting a trending event, which can be categorized by its predictability. When an event contains strong inter-day dependencies, the autocorrelation value will be high. Given observed time series values $\{y_1, ..., y_N\}$ and its mean $\bar{y}$, autocorrelation is the similarity between observations as a function of the time lag l between them.

$$Autocorrelation(Y, l) = \frac{\sum_{i=1}^{N-k}(y_i - \bar{y})(y_{i+l} - \bar{y})}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{2}$$

In this work, we consider autocorrelation at the one time unit lag only ($l = 1$), which shifts the second time series by one day. This is called the first-order autocorrelation of the first $N - 1$ observations $\{y_1, ..., y_{N-1}\}$ and the next $N - 1$ observations $\{y_2, ..., y_N\}$. In addition, we also employ the features **global trend** and **local trend** for detecting popular events.

In order to detect *unseen* events that become popular recently, we employ the feature **prediction error**, which has been proposed by Radinsky et al. [18] for capturing *surprises* in querying behavior streams. Specifically, a surprise represents an unplanned event happening when there is a significant error in the residuals of a prediction model, i.e., by computing the sum of squared errors of prediction (SSE). The score implies how *unplanned* is the time series at a given time point. Given a time series $Y = \{y_1, ..., y_N\}$, we estimate its predicted values $\hat{Y} = \{\hat{y}_1, ..., \hat{y}_N\}$ using a simple linear regression model. The surprise score of Y with respect to $\hat{Y}$ at $time(q)$ can be calculated as:

$$SSE(Y, \hat{Y}) = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3}$$

Click entropy is a measurement of query click variation over a set of URLs $U = \{u_1, ..., u_j\}$ for a given query q, which is originally proposed in [5] and used for improving the personalized web search task. Intuitively, smaller click entropy means less variation among individuals, that is, users agree with each other on a small number of web pages, whereas higher click entropy indicates that many web pages were clicked for the query. We regard this measurement as a signal of *temporal content dynamics* [14], which is an indirect indication of events in query streams.

$$ClickEntropy(U_q) = \sum_{u \in U_q} -P(u|q) \log P(u|q) \quad (4)$$

where U_q is a set of clicked URLs for a given query q at time t, and the final value will be aggregated over the last n days with respect to the hitting time $time(q)$. $P(u|q)$ is the probability of u being clicked among all the clicks on q:

$$P(u|q) = \frac{|click(u,q)|}{\sum_{u_i \in U_q} |click(u_i,q)|} \quad (5)$$

We take into account other statistical features on time series data, such as, burstiness, kurtosis and temporal KL-divergence. We make use of various **burstiness** characteristics, i.e., the number of bursts, the maximum weight, the longest duration, and the distance between the maximum burst and hitting time, using the burst detection algorithm proposed in [13].

Temporal KL-divergence was proposed in [9] to determine temporal classes of queries and it is measured as the difference between the distribution over time of top-k retrieved documents D_q with respect to q and the document collection D. Similarly, we use temporal KL-divergence for capturing the temporality of a given query with respect to its top search results. Note that, this feature is only generated from an external document collection as follows.

$$TemporalKL = \sum_{t \in T} P(t|q) \log \frac{P(t|q)}{P(t|T)} \quad (6)$$

where T is the set of all publication dates in the document collection D. $P(t|T)$ is the probability of a publication date t in the collection D. $P(t|q)$ is the probability of generating a publication date t given q by considering the top-k retrieved documents D_q and their relevance scores. In this paper, our retrieval model is based on *relevance language modeling* [15], that is, the top-k retrieved documents D_q are considered and weighed according to the document's probability of relevance.

$$P(t|q) = \sum_{d \in D_q} P(t|d) \frac{P(q|d)}{\sum_{d' \in D_q} P(q|d')} \quad (7)$$

where $P(q|d)$ is a retrieval score of d for a particular ranking model. $P(t|d)$ is equal to 1 if $PubTime(d) = t$, and 0 if $PubTime(d) \neq t$.

Inspired by the previous work [9], we employ **kurtosis**, which is a basic statistic method to measure the *peakedness* or *skewness* of a probability distribution. Intuitively, kurtosis can capture sporadic or spiky events, e.g., breaking news, celebrities, and short-span events. It quantifies how much of the probability distribution is contained in the peaks, and how much in the low-probability regions. Kurtosis is calculated as the ratio of the fourth moment and variance squared.

We also consider information about named entities (i.e., person, location and organization) and temporal expressions. Consequently, we leverage such information as features with a binary value, e.g., whether a query contains a temporal expression, or it refers to a name of person, organization or location. To this end, we exploit the statistics of a cluster c derived from the method explained in Section 3.1, e.g., the number of distinct queries in c, as well as the maximum number, the average, and the sum of query frequencies in c.

Table 1: Statistics of the two query datasets.

Dataset	AOL	MSN
Queries	18,614,850	6,015,318
Unique queries	1,087,259	268,271
Unique URLs	545,223	788,441
Click-through	1,689,712	4,894,418
Explicit temporal queries		
Automatically identified	19,531	2,728
Labeled as events	*256*	*117*
Implicit temporal queries		
Automatically identified	121,176	17,009
Labeled as events	*305*	*159*

4. EVALUATION

In the following, we explain our experimental setting including the description and statistics of query log datasets, relevance assessment, parameters as well as studied methods for comparisons. Finally, we discuss experimental results that show the performance of our approach.

4.1 Experimental Setting

Query Log Datasets. We used two real-world query log datasets publicly available. The first dataset is the AOL query logs, which consists of more than 30 million queries covering the period from March 1, to May 31, 2006. The second dataset we used is the MSN query logs[4] composed of about 15 million queries sampled from May 2006. User information is anonymized and adult search queries are ignored from the study. We followed data filtering steps suggested in [24] as follows: 1) removed all non-English queries, 2) converted queries into lower case, 3) ignored queries with frequency less than 5 (for filtering noisy data), 4) ignored those with frequency more than 15,000 (i.e., navigational queries, which are not useful for our study), and 5) removed click-through data with frequency less than 3.

After applying the pre-processing steps, we further filtered the candidates (identified by the method described in Section 3.1) using a heuristic based on the cluster statistics (e.g., the number of queries and the sum of query frequencies). In total, we randomly selected over 20,000 candidates for manually labeling and identified 837 event-related queries. A query is regarded as *event-related* if it refers to, or related to a real-world event happening in the past, present or future. In addition to assign a query to each of temporal classes, we also constructed the temporal fact, e.g., information about time and locations of events, using external sources, such as, Wikipedia, a news archive portal and a search engine. Knowing the precise temporal information of events is crucially important for validating our time-aware ranking method. The statistics of our datasets are shown in Table 1, and the overview of existing datasets in this line of work is presented in Table 2.

Parameter Setting. We employed HeidelTime [20] for temporal expression extraction. The cleansing step of event candidate identification requires parameters: Jaccard similarity threshold was 0.2; edit distance threshold was 3; and the overlap n-gram was 2 terms. For burstiness features, we used the burst detection technique provided by CISHELL[5]) with default parameters.

[4]`http://research.microsoft.com/~nickcr/wscd09/`
[5]`http://wiki.cns.iu.edu/display/CISHELL/Burst+Detection`

Table 2: Comparison of existing temporal query datasets.

Kulkarni et al. [14]	Bing (25/03/2010 - 28/05/2010) 100 temporal queries
Shokouhi [19]	Bing (2006 - 2010) 74 seasonal queries
Radinsky et al. [18]	Bing (15/12/2010 - 25/04/2011) 504 dynamic queries 330 temporal-reformulation queries 1,836 alternating queries
Kairam et al. [10]	Bing (19/07/2012 - 30/08/2012) 393 trending queries 218 (filtered) trending queries
Our work	AOL (01/03/2006 - 31/05/2006) **561 event-related queries** MSN (01/05/2006 - 31/05/2006) **276 event-related queries**

Metrics. We employed several classifiers, i.e., support vector machine (SVM), AdaBoost, decision tree (J48), and neural network (NN). For assessing the performance of classification, we measured accuracy, precision, recall and F-measure. All classification methods were performed using 10-fold cross validation.

4.2 Experimental Results

Feature Selection Results. For getting a first idea of high impact features we start our feature selection analysis. In general there are three main strategies for feature selection, filter, wrapper and embedded. While the filter strategy is independent from the classifier and can be computed in a pre-processing step, wrapper and embedded methods depends on the classifier in a sense that the wrapper strategy is selecting the best feature-subset with the best performance and the embedded feature selection is directly bonded in the learning algorithm. In this work, we focus on the filter strategy with an univariate method since we want to investigate the importance of the features independent from the classification method.

For this purpose we used the InfoGainAttributeEval from the WEKA tool, which evaluates the features by considering the information gain of the feature with respect to the class and rank them based on their weight. The resulting top-20 features are shown in Table 3, where Q, U, and D denote features extracted from the aggregated query volume, click-through information, and the temporal document collection, respectively.

We notice that the most discriminative features cover features derived from the temporal document collection, and the query streams. Regarding top-10 features in the list, *temporalKL* and *kurtosis*, which are measured using the distribution over time of top-k retrieved documents, are among the most influential ones. In addition, trend-based features or those categorizing trending events, such as, *autocorrelation*, *burst weight*, and *trending level*, also play an important role for query classification. Interestingly, the click-through features are outperformed by simple features like the maximum number, and the sum of queries in a cluster. We also observe that the *seasonality* computed from the query logs has less impact than the one extracted from the document collection. Our explanation is that the longer the time span, the better the seasonality pattern can be recognized.

Table 3: Top-20 features ranked by the information gain.

Rank	Feature	Weight	Source
1	TemporalKL	0.336	D
2	Autocorrelation	0.336	D
3	Kurtosis	0.225	D
4	Seasonality	0.222	D
5	Max. Query Freq.	0.090	Q
6	Burst Weight	0.072	Q
7	No. of Queries	0.050	Q
8	Kurtosis	0.049	Q
9	Trending Level	0.039	Q
10	Sum Query Freq.	0.029	Q
11	Click Entropy (14 days)	0.028	U
12	Person Entity	0.027	Q
13	SSE	0.024	Q
14	Autocorrelation	0.023	Q
15	Avg. Query Freq.	0.021	Q
16	Seasonality	0.016	Q
17	Click Entropy (3 days)	0.012	U
18	Max. Query Freq.	0.010	Q
19	Distance Max Burst	0.008	Q
20	Trending Scope	0.006	Q

Table 4: Performance of different classifiers.

Model	Accuracy	Precision	Recall	F-Measure
SVM	0.677	0.781	0.676	0.551
AdaBoost	0.792	0.788	0.792	0.783
NN	0.834	0.832	0.834	0.833
J48	0.904	0.904	0.905	0.904

Classification Results. The overall results for query classification are shown in Table 4. It can be seen that the decision tree model (J48) gives the best performance (of about 90% in all metrics), while SVM is the worst performing model compared to other classification algorithms. To understand further the impact of individual feature the event-related query classification performance, we compute the performance of each individual feature from the top-k features ranked by the information gain method. Figure 2 illustrates the classification performance (using J48) by incrementally adding an individual feature for training the model. When using only *temporalKL*, we obtain the accuracy of approx. 84%. It gains an increased performance by adding the features *autocorrelation*, *kurtosis*, and *seasonality*. However, the performance has dropped after adding *max. query freq.* and so on. To summarize, our experimental results identify a core set of features that can capture the most discriminative representatives for event-related query classification. This is valuable, not only for building models more efficiently in large scale systems, but also for figuring out the directions we need to concentrate in future studies for a more fine-grained understanding of temporal information needs and event-related query detection methods.

5. RELATED WORK

Detecting *event-related* queries is emerging as an important task in web search and retrieval. Strötgen and Gertz [21] have explored the notion of event-based retrieval, returning events instead of documents. Zhang et al. [23] addressed the

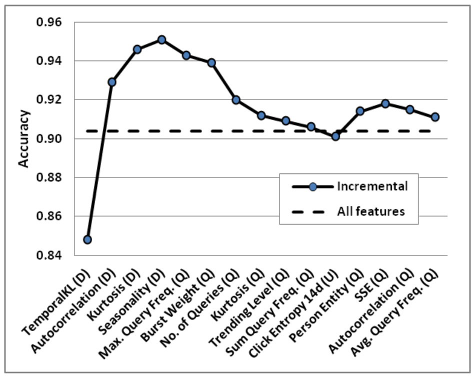

Figure 2: Incremental performance of the J48 classifier.

classification of recurrent event queries. Metzler et al. [16] proposed mining query logs in order to identify implicit temporal queries. Moreover, they presented a ranking method concerning temporal information needs that are not provided by a query as such. Kulkarni et al. [14] studied how users' information needs change over time, and Shokouhi [19] employed different time series analysis methods for detecting seasonal queries. Ghoreishi and Sun [6] introduced a binary classification approach for detecting popular event-related queries. While this previous work also determined the event-relatedness of queries, they mainly focused on *popular queries*. We, on the other hand, not only focus on identifying popular queries, but also *less popular* queries with underlying temporal intent, in which changes in query popularity cannot be observed, such as, *commemorative* or *anticipated* events.

6. CONCLUSIONS

In this paper, we studied the problem of detecting event-related queries in search streams. We considered two main classes of event-related queries, i.e., *explicit* and *implicit* temporal queries. We proposed a method for event detection consisting of two main steps: 1) event candidate identification, and 2) event-related query classification. We employed machine learning techniques trained with a set of features, which were derived from query logs as well as an external document collection. In our experiments, we evaluated the performance of different classification algorithms trained using our proposed features. To this end, we have conducted a feature selection technique in order to identify a core set of features, which can capture the most discriminative representatives of different feature categories.

Acknowledgments This work was partially funded by the European Commission for the FP7 project ForgetIT and the ERC Advanced Grant ALEXANDRIA under the grant numbers 600826 and 339233, respectively.

7. REFERENCES

[1] G. E. P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[2] R. Burghartz and K. Berberich. MPI-INF at the NTCIR-11 Temporal Query Classification Task. In *Proceedings of the 11th NTCIR Conference*, 2014.

[3] R. Campos, G. Dias, A. Jorge, and C. Nunes. GTE: A distributional second-order co-occurrence approach to improve the identification of top relevant dates in web snippets. In *Proceedings of CIKM '12*, 2012.

[4] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. STL: A seasonal-trend decomposition procedure based on loess (with discussion). *Journal of Official Statistics*, 6:3–73, 1990.

[5] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of WWW '07*, 2007.

[6] S. N. Ghoreishi and A. Sun. Predicting event-relatedness of popular queries. In *Proceedings of CIKM '13*, 2013.

[7] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

[8] H. Joho, A. Jatowt, and R. Blanco. NTCIR temporalia: a test collection for temporal information access research. In *Proceedings of WWW '14*, 2014.

[9] R. Jones and F. Diaz. Temporal profiles of queries. *ACM Trans. Inf. Syst.*, 25, July 2007.

[10] S. R. Kairam, M. R. Morris, J. Teevan, D. J. Liebling, and S. T. Dumais. Towards supporting search over trending events with social media. In *Proceedings of ICWSM '13*, 2013.

[11] N. Kanhabua and K. Nørvåg. Determining time of queries for re-ranking search results. In *Proceedings of ECDL '10*, 2010.

[12] N. Kanhabua, S. Romano, and A. Stewart. Identifying relevant temporal expressions for real-world events. In *Proceedings of the SIGIR 2012 Workshop on Time-aware Information Access (TAIA '12)*, 2012.

[13] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of SIGKDD '02*, 2002.

[14] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics. In *Proceedings of WSDM '11*, 2011.

[15] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of SIGIR '01*, 2001.

[16] D. Metzler, R. Jones, F. Peng, and R. Zhang. Improving search relevance for implicitly temporal queries. In *Proceedings of SIGIR '09*, 2009.

[17] S. Nunes, C. Ribeiro, and G. David. Use of temporal expressions in web search. In *Proceedings of ECIR '08*, 2008.

[18] K. Radinsky, K. Svore, S. Dumais, J. Teevan, A. Bocharov, and E. Horvitz. Modeling and predicting behavioral dynamics on the web. In *Proceedings of WWW '12*, 2012.

[19] M. Shokouhi. Detecting seasonal queries by time-series analysis. In *Proceeding of SIGIR '11*, 2011.

[20] J. Strötgen and M. Gertz. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of Workshop on Semantic Evaluation*, 2010.

[21] J. Strötgen and M. Gertz. Event-centric search and exploration in document collections. In *Proceedings of JCDL '12*, 2012.

[22] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Query clustering using user logs. *ACM Trans. Inf. Syst.*, 20(1):59–81, Jan. 2002.

[23] R. Zhang, Y. Konda, A. Dong, P. Kolari, Y. Chang, and Z. Zheng. Learning recurrent event queries for web search. In *Proceedings of EMNLP '10*, 2010.

[24] X. Zhu, J. Guo, X. Cheng, P. Du, and H.-W. Shen. A unified framework for recommending diverse and relevant queries. In *Proceedings of WWW '11*, 2011.