InZeit: Efficiently Identifying Insightful Time Points

Vinay Setty, Srikanta Bedathur, Klaus Berberich, Gerhard Weikum
Max-Planck Institute for Informatics
Saarbrücken, Germany
{vsetty,bedathur,kberberi,weikum}@mpi-inf.mpg.de

ABSTRACT

Web archives are useful resources to find out about the temporal evolution of persons, organizations, products, or other topics. However, even when advanced text search functionality is available, gaining insights into the temporal evolution of a topic can be a tedious task and often requires sifting through many documents.

The demonstrated system named INZEIT[1] (pronounced "in-sight") assists users by determining insightful time points for a given query. These are the time points at which the top-k time-travel query result changes substantially and for which the user should therefore inspect query results. INZEIT determines the m most insightful time points efficiently using an extended segment tree for in-memory bookkeeping.

1. INTRODUCTION

Thanks to improved digitization and preservation techniques, archives accessible on the Web nowadays contain large numbers of documents published during a long period of time. Prominent examples include the archive of the newspaper The New York Times [12] containing more than 13 million articles published since 1851 and the Internet Archive [6] that contains 150 billion snapshots of webpages with the earliest dating back to 1996. Such archives on the Web are highly prized for finding out about persons, organizations, products or topics and establishing their timelines. Consider, for instance, a social scientist researching key political developments in relation to an important topic such as `same sex marriage` or `health care reform`.

When using standard text search, results for a given query are returned in the order of their estimated relevance to the query or in chronological order of their publication date. Choosing either of these orderings forces the user to sift through a large number of results in order to gain insights into the temporal evolution of a topic (expressed by the query). A more sophisticated search feature such as time-travel query [3] allows the user to retrieve results for a specific time point. However, it still fails to help in a situation when the user has *no prior knowledge* about the right time points to explore for the given topic. The user may still have to pose many time-travel queries for a large number of time points in order to learn the timeline of the topic under study.

In this demonstration, we present INZEIT, a system which assists users by determining *insightful time points* to obtain quick insights on the temporal milestones of the topic of the query. This may also help the user in further examining the archive by posing targeted time-travel queries. Towards this goal, INZEIT analyzes how the top-k time-travel query result for the topic evolves over time. Insightful time points are then identified as those where the top-k query result undergoes significant change. This approach is different from existing systems (e.g., Google News Archive search) that consider merely the number of relevant document published at a time to identify interesting time points. These systems thus ignore the estimated relevance of documents to the query, and end up highlighting time points even if they contain documents from the "long tail" with low relevance score as long as there are a large number of them.

INZEIT reads query results sequentially in the order of their estimated relevance and computes insightful time points. In this process, quick response times are crucial so that the user can immediately start to learn about the temporal evolution of the topic at hand. INZEIT achieves efficiency by using two novel techniques: first, INZEIT uses an *early-terminating method* that allows it to terminate as soon as the m most insightful time points have been accurately identified – typically long before the entire query result has been read. Second, a *rank-aware dynamic segment tree* [14] is used to perform efficient in-memory bookkeeping, which additionally also has the benefit of having the top-k time-travel query results for identified insightful time points ready without needing any further query processing.

For our demonstration, INZEIT is deployed on the New York Times Annotated Corpus [12] – a real-world dataset that comprises 1.8 million newspaper articles published between 1987 and 2007. Using our own and attendee-suggested queries, we showcase the usefulness of insightful time points, compare them against time points determined by a purely frequency-based approach, and demonstrate the effectiveness of the techniques that INZEIT builds upon.

Organization. The rest of this paper is organized as follows: Related research is briefly discussed in Section 2. We lay out the techniques underlying our system in Section 3, before describing its architecture in Section 4. Section 5 illustrates the results from INZEIT and provides an outline of the proposed demonstration of the system.

2. RELATED WORK

Temporal exploration of results is recently gaining lot of attention due to the inherent intuitiveness of organizing results along the timeline [1]. Arguably the most popular of such tools is the timeline exploration interface of the Google News-archive Search (`http:`

[1]**Zeit** (German): n. Time, temporal, period (die Zeit).

Proceedings of the VLDB Endowment, Vol. 3, No. 2
Copyright 2010 VLDB Endowment 2150-8097/10/09... $ 10.00.

//news.google.com/archivesearch). Although the details of how their timelines are constructed are not publicly available, it seems likely that they use the distribution of result documents along time to identify interesting time-points. From the research community, there have been a few similar efforts that use frequency of relevant documents to identify bursty or interesting times [7, 2]. InZeit differs from all these previous proposals in that instead of considering the frequency of results at any time point, it pays specific attention to qualitative change over time by computing the variation in the top-k results for the query. Also, it has significantly lower computational overhead after the back-end search engine has processed the text query. This is in marked contrast to sophisticated but computationally expensive techniques used in news event detection [8, 9].

3. IDENTIFYING INSIGHTFUL TIME POINTS

In this work we consider a document collection $\mathcal{D}$ comprising of documents which stay valid for an arbitrary period of time. Each document $D \in \mathcal{D}$ is represented as a tuple $\langle did, t_b, t_e \rangle$ where did is the unique document identifier, t_b and t_e are the timestamps of the begin and end time. When a keyword query, q, is processed on $\mathcal{D}$ we obtain a ranked result subset, $R \subset \mathcal{D}$, where each document $d \in R$ has an associated relevance score $d.s_q$ (e.g., determined using a scoring model such as Okapi BM25[11]) as well as $d.r_q$, the rank of the document d for the given query. We define a top-k result set for a given query 'q' at time point t as $top_k^q(t)$. We consider a time point $t_i \in \mathcal{T}$ insightful if $top_k^q(t_i)$ differs significantly from $top_k(t_{i-1})$ where $t_{i-1} \in \mathcal{T}$ is predecessor of t_i. We define the *insightfulness* of a time point as the sum of weights of documents in top-k set difference of two consecutive time points, t_{i-1} and t_i (note that we employ discrete notion of time in our work). It is formally defined as:

$$\zeta(t_i) = \sum_{d \,\in\, top_k^q(t_i) \setminus top_k^q(t_{i-1})} w(d),$$

where $w(d)$ is weight based on the relevance rank $d.r_q$ defined as $w(d) = \frac{1}{d.r_q}$. The *frequency based approach*, in contrast, counts the number of relevant documents for a given query which begin at a given time point t as a measure to show the importance of time point t. It is formally defined as:

$$f(t) = |\{d \in D \ \wedge \ d \in R \ \wedge \ d.t_b = t\}|$$

Consider an example in which we have six documents in the result list R for an arbitrary query 'q' as shown in Figure 1. Assuming we are looking at top-3 (k=3) documents at three consecutive time points $\{t_1, t_2, t_3\}$, we can see that time point t_2 is most insightful with $\zeta(t_2) = \frac{1}{1} + \frac{1}{5} = 1.2$ even though $f(t_2) < f(t_1)$ and $f(t_2) = f(t_3)$. An important observation here is that even though document d_5 is not starting at time point t_2 it contributes to the insightfulness at t_2 because $d_5 \in top_k^q(t_2) \setminus top_k^q(t_1)$. At time point t_3, $\zeta(t_3) = 0$ even though $f(t_3) = 1$ because there is no change in $top_3^q(t_3)$ from $top_3^q(t_2)$.

Problem Definition. We are given a user query 'q' and the corresponding list of relevant documents $R \in \mathcal{D}$ is assumed to be obtained in the decreasing order of relevance score in an incremental fashion. And $\mathcal{T}$ is a set of all time points occurring as boundaries of the valid-time intervals in R, i.e., it is defined as:

$$\mathcal{T} = \bigcup_{d \in \mathcal{D}} \{d.t_b, d.t_e\}.$$

The goal is to read the minimal subset of R and compute $\forall t \in \mathcal{T}$, $top_k^q(t)$ and at the same time determine the top-m time points with

Insightfulness	ζ(t₁)=0.25	ζ(t₂)=1.2	ζ(t₃)=0
Frequency	f(t₁)=2	f(t₂)=1	f(t₃)=1
top-k set	{d2,d3,d4}	{d1,d2,d5}	{d1,d2,d5}

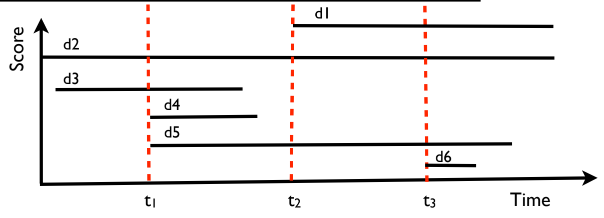


Figure 1: Computation of insightfulness

highest insightfulness value represented as:

$$top_m(q) = \{\{t_1, t_2, ...t_m\} \mid t_i \in \mathcal{T} \wedge \zeta(t_i) \geq \zeta(t_m)\},$$

where $\zeta(t_m)$ is the insightfulness of the m^{th} highest interesting time point in $\mathcal{T}$.

Solution. The key idea underlying our solution to the above problem is that we can exploit the score order of documents to materialize the $top_k^q(t)$, $\forall t \in \mathcal{T}$. Each document $d \in R$ can be visualized as an interval spanning from, $d.t_b$ to $d.t_e$ in the time axis. Assuming we have n documents to be processed, there are at most $2n$ unique end points. Since these are the only time points where there is any change in $top_k^q(t)$, it is clear that only these $2n$ points can have $\zeta(t) > 0$. *segment tree*[4] is a data structure which efficiently indexes the intervals by partitioning them at their end points. In segment tree if there are n intervals with $2n$ distinct end points then we can have at most $2n+1$ atomic segments (where no interval starts or ends). The segment tree over these sequence of atomic segments is a balanced binary search tree. Each node N of the segment tree can be described by an extent interval $interval(N)$ (spanning between lower bound lb and upper bound ub) which is the union of all atomic segments under N's subtree. This property of segment tree can be used to materialize the top-k results for each atomic segment in the segment tree and also compute insightfulness of consecutive intervals. Unlike the standard segment tree, we additionally need to dynamically insert and materialize the top-k results at each node. For this purpose, we make use of a novel extension of segment tree called *Rank Aware Dynamic Segment Tree (RADST)* that was proposed in [14] which supports dynamic insertions and maintenance of top-k set at these intervals. In this extended segment tree, the $top_k^q(t)$ are materialized using a priority queue, $N.topk$, at each node.

Early Termination Algorithm. Even though *RADST* helps in dynamically indexing and materializing top-k results for all time points in $\mathcal{T}$, in practice not all time points may have k documents alive. In such cases we end up reading all the documents in R making the identification of insightful time points slower. To overcome this problem, we propose the use of an NRA ("No Random Access")[5] style early pruning mechanism to minimize number of entries read from R. It is easy to see that insightfulness $\zeta(t)$ is monotonically decreasing as we go down in the document ranked list (increasing rank values). Using this property and assuming that we obtain R incrementally in increasing order of rank, we bound the best possible insightfulness for any time point. This helps in early termination of our algorithm.

Algorithm 1 describes the steps involved in dynamically inserting document into *RADST* (line 5) and updating insightfulness of node intervals (line 9) whenever a document is not inserted in previous interval (line 8). After updating the insightfulness we obtain a bound on best insightfulness that a time point can achieve (line 10). In each iteration, we compare if the worst insightfulness we have seen so far is better than best insightfulness that can be achieved

Algorithm 1 Top-m insightful time points identification

```
Input: Score sorted result list $R$ for keyword query $q$, $m$, $k$
Output: $top_m(q)$
 1: top-$m \leftarrow \varnothing$; min-$m \leftarrow 0$;
 2: $RADST\ tree \leftarrow \varnothing$; $PriorityQueue\ candidates \leftarrow \varnothing$;
 3: while $R$.size != 0 $\wedge$ top-$m$.size != $m$ do
 4:   $d \leftarrow \text{argmax}_d\{d.s_q | d \in R\}$
 5:   $tree$.insert($d$)
 6:   for all nodes $N$ in $tree$ such that $d.t_b \leq N.lb \wedge d.t_e > N.ub \wedge N.topk.size < k$ do
 7:     $N.topk \leftarrow N.topk \cup d$
 8:     if $d \notin N' \wedge N'.ub = N.lb$ then
 9:       $\zeta(N.lb) \leftarrow \zeta(N.lb) + \frac{1}{d.r_q}$
10:       $best\text{-}\zeta(N.lb) \leftarrow \zeta(N.lb) + \sum_{i=1}^{k-N.topk.size} \frac{1}{d.r_q+i}$
11:       if $\zeta(N.lb) > min\text{-}m$ then
12:         top-$m \leftarrow$ top-$m$ - $\text{argmin}_t\{\zeta(t) | t \in \text{top-}m\}$
13:         top-$m \leftarrow$ top-$m \cup N.lb$
14:         min-$m \leftarrow \min\{\zeta(t) | t \in \text{top-}m\}$
15:       else if $best\text{-}\zeta(t) >$ min-$m$ then
16:         $candidates \leftarrow candidates \cup N.lb$
17:       end if
18:     end if
19:   end for
20:   $threshold \leftarrow \max\{best\text{-}\zeta(t) | t \in candidates\}$
21:   if $threshold \leq$ min-$m$ then
22:     return top-$m$
23:   end if
24: end while
```

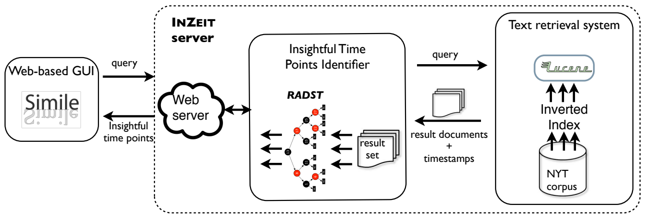


Figure 2: System architecture of InZeit

(lines 20-23) and we terminate the algorithm and return top-m insightful time points we have seen so far. Due to lack of space, we omit detailed description of the algorithm's workings which is available from [14].

Consider the same example in Figure 1 suppose we are interested in top-2 insightful time points and we have read the result list R until d_4 (4th-rank) from the top. Now we have two time points t_1 and t_2 in the *top-m* list, we have to decide if time point t_3 can make it to top-m. Since t_3 already has two documents d_1 and d_2 only one document can make it to $top_k^q(t_3)$, which in the best case is d_5. Using this knowledge we obtain a bound on best insightfulness t_3 can achieve $best\text{-}\zeta(t_3) = 0.2$. However we observe that $\zeta(t_1) = 0.25$ the worst insightfulness in top-m is better than $best\text{-}\zeta(t_3)$. Now we can stop reading more documents from R as we have already obtained the *top*-2 most insightful time points we are looking for.

4. SYSTEM ARCHITECTURE

In this section we describe InZeit's system architecture and its main components as illustrated in Figure 2. The three main components of InZeit are (1) Web-based GUI, (2) Insightful Time Points Identifier (ITPI), and (3) Text retrieval system (TRS).

The **Web-based GUI** is implemented using simple HTML forms, JavaScript and SIMILE visualization library [15]. A screenshot of the interface is shown in Figure 3. As indicated in the figure, the visualization comprises of the following parts: (a) a text box where the user can enter the text query using the full Lucene keyword query syntax [10], (b) an input field where the use can enter the number of insightful time points required (default value set to 100), (c) an optimal input area to enter the time period to *zoom into*, (d) timeline visualization of insightful time points, (e) similar visualization for the result of the traditional frequency counting approach, and (f) the ranked list of top-k documents at any time point along the timeline – shown when the user clicks on the time point. The Web-based GUI sends the query along with the optional parameters to the backend Java server. After processing the query, the backend computes and returns the insightful time points along with their insightfulness values, $\zeta(t)$. For the purpose of this demonstration, it also returns the frequency values at all time points $f(t)$. These results are dynamically visualized using timeline plots from SIMILE. When the user clicks on a specific time point to inspect, the top-k documents from that time point are retrieved from the server. The visualization also highlights those documents which have the same begin time as the time point queried for.

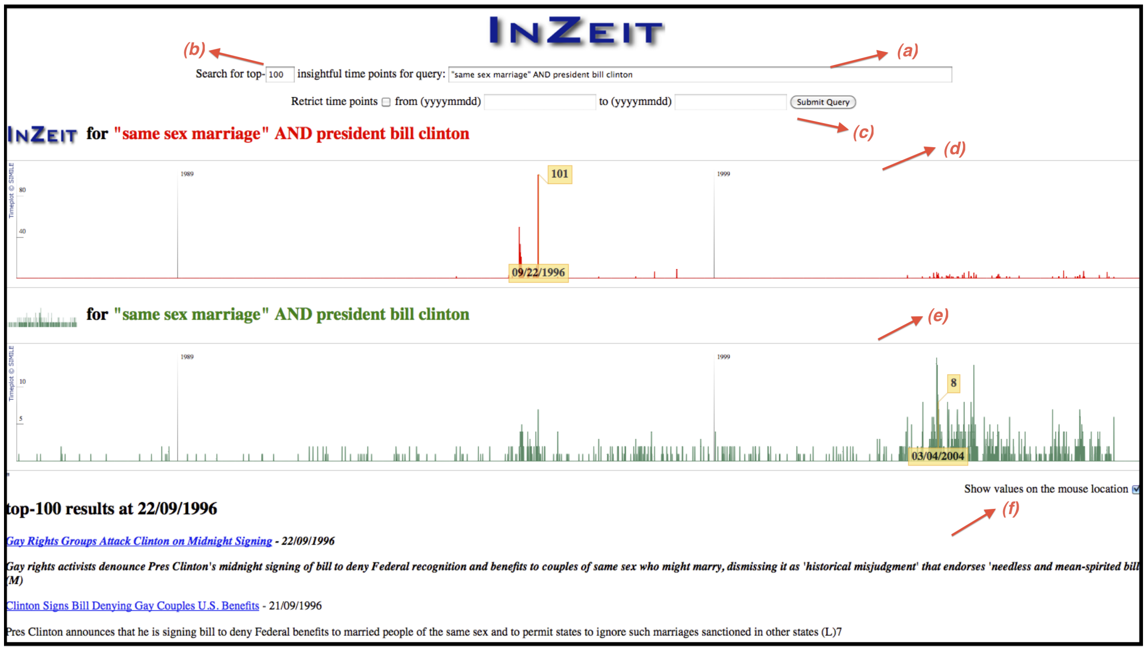
ITPI. This is the core component of the InZeit system. The interface to this component is via an Apache web server, which forwards the queries from the GUI to the background Java service that computes the results. This service, passes the user query to the backing text retrieval system, and starts pulling the relevant results from it in score order in an incremental fashion, terminating early using the Algorithm 1. As results are read, ITPI concurrently builds the *RADST* and maintains it in memory for the remaining interactions. In this demonstration, this component also computes the frequency measure at each time point (without any early termination). The results are sent back to the GUI for visualization. When the user selects a time point in the visualization, then the ITPI component performs a *stabbing query* on the *RADST* [14] and returns the corresponding top-k documents in their score order.

TRS. The text query is evaluated to obtain a relevance ranked list of documents by this module. In InZeit's implementation we used Lucene [10] as the text retrieval engine, by extending its inverted list posting structure to include the publication time stamp of documents. In practice it can be replaced by any search engine that can answer keyword queries in relevance order along with timestamps of the result documents. We also extended the underlying relevance scoring model of Lucene to Okapi BM25 using an open source plug-in [13].

5. DEMONSTRATION DESCRIPTION

To demonstrate the utility of InZeit in exploring the timelines of a topic, we make use of a large annotated news corpus recently made available by The New York Times [12]. This corpus comprises of about 1.8 million daily articles published in the New York Times newspaper during the 20-year period between 1987–2007. We indexed the New York Times Annotated Corpus using Lucene, after setting for each document a fixed lifetime of 90-days since its time of publication. This is done to reflect the real world setting where the news articles are publicly available only for a limited period from their publication, and to coarsely model the commonly used time-decaying relevance model for news articles.

Figure 3 shows a screenshot of the InZeit's GUI, visualizing the results for the query `''same sex marriage'' AND president bill clinton`. As we mentioned above, results from our approach as well as frequency counting approach are visualized using separate SIMILE timelines. When one compares the outputs from both the approaches, following observations can be made: first of all, the InZeit approach clearly ranks the time points based on their $\zeta(t)$ value, thus enabling the user to get the bird's eyeview of the temporal evolution of the query. On the other hand, the frequency-based approach generates a large number of "peaks" thus making it difficult to see the key time points easily. Secondly, the InZeit identifies the key real-world event where President Bill Clinton signed a bill denying the federal benefits to same sex couples as the most insightful time point for the query.



Search for top-100 insightful time points for query: "same sex marriage" AND president bill clinton

Retrict time points ☐ from (yyyymmdd) to (yyyymmdd) Submit Query

INZEIT for "same sex marriage" AND president bill clinton

for "same sex marriage" AND president bill clinton

Show values on the mouse location ☑

top-100 results at 22/09/1996

***Gay Rights Groups Attack Clinton on Midnight Signing* - 22/09/1996**

Gay rights activists denounce Pres Clinton's midnight signing of bill to deny Federal recognition and benefits to couples of same sex who might marry, dismissing it as 'historical misjudgment' that endorses 'needless and mean-spirited bill (M)

Clinton Signs Bill Denying Gay Couples U.S. Benefits - 21/09/1996

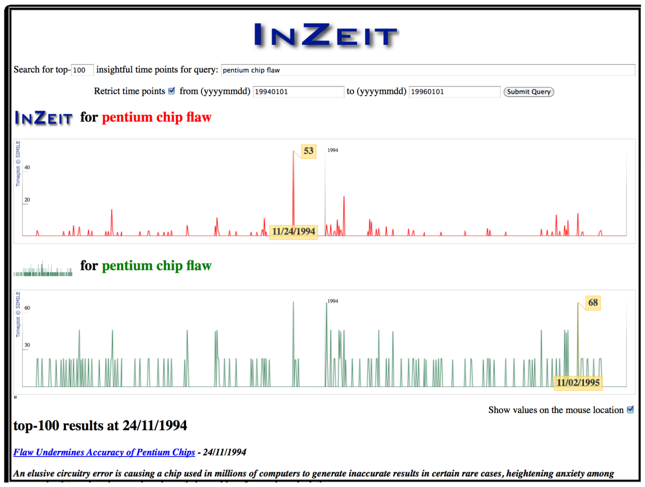
Pres Clinton announces that he is signing bill to deny Federal benefits to married people of the same sex and to permit states to ignore such marriages sanctioned in other states (L)7

Figure 3: INZEIT over the complete history of NYT

On the other hand, the frequency based method highlights a later, but irrelevant time point due to a large number of documents that are only marginally relevant for the query.

Temporal Zoom-In. The INZEIT also enables user to *zoom into* a relatively smaller time period, in order to unearth insightful time points better. This functionality also helps to improve the quality of visualization, which is highly dependent on the resolution of the screen. Figure 4 illustrates the temporal zoom-in feature, using the query `pentium chip flaw` with time period restricted to the range 1/1/1994–1/1/1996. Once again, our insightfulness based approach manages to highlight the period when the infamous floating point bug in the Pentium series of CPUs from Intel were discovered and reported in the New York Times. Although the frequency based approach, also identifies this as an important time for the query, it is buried within a number of other time points that are considered equally important. In fact, some of these (e.g., the one pointed to in the bottom timeline) correspond to the introduction of improved Pentium Pro chips!

Finally, during the demonstration of our system we invite the participation of all visitors to a hands-on session trying out their favorite topic for which they seek insightful time points over INZEIT.



Search for top-100 insightful time points for query: pentium chip flaw

Retrict time points ☑ from (yyyymmdd) 19940101 to (yyyymmdd) 19960101 Submit Query

INZEIT for pentium chip flaw

for pentium chip flaw

Show values on the mouse location ☑

top-100 results at 24/11/1994

***Flaw Undermines Accuracy of Pentium Chips* - 24/11/1994**

An elusive circuitry error is causing a chip used in millions of computers to generate inaccurate results in certain rare cases, heightening anxiety among

Figure 4: INZEIT with temporal zoom-in

6. REFERENCES

[1] Omar Alonso, Michael Gertz, and Ricardo A. Baeza-Yates. Clustering and exploring search results using timeline constructions. In *Proc. of CIKM*, 2009.

[2] Nilesh Bansal and Nick Koudas. BlogScope: A System for Online Analysis of High Volume Text Streams. In *Proc. of VLDB*, 2007.

[3] Klaus Berberich, Srikanta Bedathur, Thomas Neumann, and Gerhard Weikum. A time machine for text search. In *Proc. of SIGIR*, 2007.

[4] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 2008.

[5] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Proc. of PODS*, 2001.

[6] Internet archive. http://www.archive.org.

[7] Jon Kleinberg. Bursty and Hierarchical Structure in Streams. In *Proc. of KDD*, 2002.

[8] Giridhar Kumaran and James Allan. Text classification and named entities for new event detection. In *Proc. of SIGIR*, 2004.

[9] Zhiwei Li, Bin Wang, Mingjing Li, and Wei-Ying Ma. A probabilistic model for retrospective news event detection. In *Proc. of SIGIR*, 2005.

[10] Lucene. http://lucene.apache.org/.

[11] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[12] New York Times Annotated Corpus. http://corpus.nytimes.com/.

[13] Joaquín Pérez-Iglesias, José R. Pérez-Agüera, Víctor Fresno, and Yuval Z. Feinstein. Integrating the Probabilistic Models BM25/BM25F into Lucene. *CoRR*, abs/0911.5046, 2009.

[14] Vinay Setty. Efficiently identifying interesting time points in text archives. Master's thesis, Universität des Saarlandes, FR Informatik, 2010.

[15] SIMILE timeplot. http://www.simile-widgets.org/timeplot/.