Published in final edited form as:

Data Knowl Eng. 2011 July ; 70(7): 642–660. doi:10.1016/j.datak.2011.03.009.

Extracting Hot spots of Topics from Time Stamped Documents

Wei Chen and

Current Organization: **Dept of Computer Science, University of Nebraska at Omaha**,
Organizational Email address: **wchen@mail.unomaha.edu**, Institute from where the PhD degree
was obtained: **PhD student**, Major Organizations worked with: **N/A**, Names of (up to 5) PhD
students graduated under: **N/A**, Homepage URL: <http://myweb.unomaha.edu/~wchen/>

Parvathi Chundi

Current Organization: **Dept of Computer Science, University of Nebraska at Omaha**,
Organizational Email address: **pchundi@mail.unomaha.edu**, Institute from where the PhD
degree was obtained: **Dept of Computer Science, University at Albany - State University of
New York**, Major Organizations worked with: **HP, GE**, Names of (up to 5) PhD students
graduated under: **N/A**, Homepage URL: <http://faculty.ist.unomaha.edu/pchundi/>

Abstract

Identifying time periods with a burst of activities related to a topic has been an important problem in analyzing time-stamped documents. In this paper, we propose an approach to extract a **hot spot** of a given topic in a time-stamped document set. Topics can be **basic**, containing a simple list of keywords, or **complex**. Logical relationships such as **and**, **or**, and **not** are used to build complex topics from basic topics. A concept of **presence measure** of a topic based on fuzzy set theory is introduced to compute the amount of information related to the topic in the document set. Each interval in the time period of the document set is associated with a numeric value which we call the **discrepancy score**. A high discrepancy score indicates that the documents in the time interval are more focused on the topic than those outside of the time interval. A **hot spot** of a given topic is defined as a time interval with the highest discrepancy score. We first describe a naive implementation for extracting hot spots. We then construct an algorithm called **EHE** (Efficient Hot Spot Extraction) using several efficient strategies to improve performance. We also introduce the notion of a **topic DAG** to facilitate an efficient computation of presence measures of complex topics. The proposed approach is illustrated by several experiments on a subset of the TDT-Pilot Corpus and DBLP conference data set. The experiments show that the proposed EHE algorithm significantly outperforms the naive one, and the extracted hot spots of given topics are meaningful.

Keywords

scan statistic; text mining; hot spots; topics

1. Introduction

Analyzing unstructured text documents such as blogs and news articles for temporal information and temporal sequential patterns is an important data mining activity due to the pervasive nature of these data [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Current search systems

© 2011 Elsevier B.V. All rights reserved.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

provide limited analysis capability with keyword search and temporal constraints over publication dates. However, these methods are inadequate because of the enormity and diversity of data sources.

It is common for blogs and news sources to discuss/publish a few news stories intensely for a period of time [10, 11]. The news stories may change frequently and be replaced with new stories, or they may stay active and the context surrounding the story may change. This kind of coverage results in bursty patterns of stories/topics. It has been well-recognized that identifying periods of bursty activity of a topic may provide much useful information such as the context of given topics [1, 2, 7, 8]. The periods of bursty activity may contain a lot of useful information related to the story and may also lead to other stories that are relevant to the topic burst.

In this paper, we describe methods to extract **hot spots** of a given topic from a time-stamped document set. Intuitively, a hot spot is a *sub-interval* of the time period that contains significantly more documents that discuss the topic than the rest of the time period.

It is well-known that capturing user intent by a topic represented using a simple list of keywords is a challenging task [13]. The topic structure supported by our system is powerful enough to capture multiple contexts in which a topic may appear and the logical relationships among these contexts. We consider two kinds of topics: a **basic topic**, which is simply a weighted list of keywords, and a **complex topic**, which is a logical expression of basic topics and operators *and*, *not*, *or*. Users specify their topic of interest using a basic or a complex topic.

To identify hot spots, we assign a **discrepancy score** to each of the $O(n^2)$ intervals of the time period of the document set, where n denotes the number of time points in the time period T of the document set. A discrepancy score of an interval is a numerical value that captures the discrepancy between the presence of the topic in the document set of the interval and its presence in the document set outside the interval. Scan statistics have been used to analyze temporal, spatial, and spatio-temporal data to identify abnormal clusters of events in fields such as epidemiology [14]. We use the temporal scan statistic [14] to compute the discrepancy score of an interval. Our work is perhaps the first one to use the temporal scan statistic to find hot spots of topics from a time-stamped document set.

We define the hot spot extraction problem as follows: given a time-stamped document set and a topic, the hot spot is a time interval with the maximum discrepancy score. When there is more than one such interval, we arbitrarily choose one of these intervals as a hot spot in this paper, but the algorithm can be easily modified to find all intervals with the maximum discrepancy score.

Extracting a hot spot requires calculating the discrepancy score of every interval in the time period of the document set. A naive implementation runs in time $O(n^3)$, where n is the number of the time points of the document set. We discuss several strategies to improve the efficiency of the naive implementation and present an algorithm, which we refer to as the *Efficient Hot spot Extraction*, (EHE) algorithm, to compute a hot spot in time $O(n^2)$. Our experimental results show that the EHE algorithm is much faster than the naive implementation practice.

Given a topic and a document set, there are many ways to compute the presence of the topic in the given document set γ , e.g., term frequency, document frequency. We provide a flexible infrastructure based on the fuzzy set theory to incorporate different methods for computing the topic presence for different purposes. In this paper, we define a method for computing the presence of a given a topic in a document set, and functions to combine the

topic presence computed from different parts of a complex query. However, fuzzy set theory-based infrastructure for computing topic presence enables users to explore different functions easily. We describe data structures and algorithms to compute the presence of a given topic in each of the $O(n^2)$ intervals, which we call the **presence measure** of the topic. A basic topic is a simple list of keywords and weights; a complex topic is constructed using logical operators and other basic topics. For a complex topic, computing its presence measure in a document set depends on the basic topics and logical operators in that complex topic. If a complex topic contains redundant subexpressions, the presence measure of the redundant subexpression must be computed just once and reused. Otherwise, computing the presence measure of a complex topic may lead to unnecessary and expensive processing. We describe a simple procedure, based on *BDD Reduce* [15, 16], to compute a **non-redundant** complex topic that is functionally equivalent to the given complex topic. We describe an efficient procedure to compute the presence measure of a topic in each of the $O(n^2)$ intervals of the document set.

We conducted experiments using two data sets: 1) the DBLP data containing the titles of papers published in VLDB and SIGMOD conferences from 1975 to 2007, and 2) the TDT-Pilot Corpus data containing the titles of Reuters and CNN news articles published from 1994 to 1995 [17]. We compared the performance of the naive implementation and that of EHE. We studied several basic as well as complex topics and analyzed the hot spots for each of the topics from the data sets. The preliminary experiments show that the proposed methods work very well in practice.

- The experiments demonstrate the great performance improvements of the EHE algorithm described in the paper. This is due to the fact that many keywords appear in only a handful of the time points which are exploited to achieve performance improvements by the EHE algorithm.
- We studied the accuracy of the EHE algorithm to extract hot spots of topics in a given document set using a synthetic data set. The synthetic data set is generated with a hot spot set at a specific interval chosen randomly. The proposed method has more than 90% accuracy in extracting the preset hot spot. We also compared the hot spots extracted by our method with the bursty intervals reported by [7]. The results show that the proposed method is superior in that it includes more time points in the hot spot with relatively high presence of the given topic, when compared to the bursty intervals reported in [7].
- We computed the TF-IDF scores using Lucene [18] for all documents with respect to the topic to examine the relationship between relevant documents and hot spots. We found that hot spots contained a significant amount of relevant documents for many of the topics studied in the experiments. Therefore, extracting hot spots is a very meaningful activity.

The rest of the paper is organized as follows. Related work is discussed in Section 2. Some preliminary definitions including the definition of discrepancy score are given in Section 3. In Section 4, we define the presence measures of topics and describe an efficient algorithm procedure to compute the presence measure of a topic in each of the intervals. In Section 5, we describe the EHE algorithm in detail. In Section 6, we describe an efficient procedure to simplify a complex topic by removing redundancy. Experimental results are discussed in Section 7. Section 8 concludes the paper.

2. Related Work

Extracting temporal information from text documents is an important research area. Detecting bursty topic patterns from text streams is an emerging problem. Fung et al. [19]

describe a parameter-free probabilistic approach to detect a set of bursty features for a bursty event from a text stream. The bursty features can then be used as a set of features for positive examples to facilitate partially supervised text classification. He et al. [20] study how to use bursty features to represent a set of documents over time. The effectiveness of the representation is studied in the context of text clustering. Mei and Zhai [21] propose probabilistic methods to discover latent themes from text, construct an evolution graph of themes, and analyze the life cycles of themes. Methods to extract correlated bursty topic patterns from multiple text streams, which can be applied even if the text streams have completely different vocabulary are studied by Wang et al. [8]. Subasic and Berendt [22] investigate how different temporal text mining techniques extract interesting facts from text documents and proposes an evaluation framework for assessing the methods. Chen et al. [23] proposes an on-line multi-resolution burst detection method to detect a set of bursty features and group them to identify a bursty event. Kleinberg [24] provide an overview of many approaches for dealing with temporal properties of information streams.

Kleinberg [7] proposes an automaton model to identify bursts in a text stream, which is used in [25] to detect bursty communities in a Weblog graph. The proposed approach is similar in spirit to [7], in that both approaches extract time intervals with an abnormal presence of a given topic. However, there are significant differences. First, our method to finding a hot spot is inspired by the temporal scan statistic method, which is extensively used to find abnormal time intervals in the public health domain. Second, our model allows users to specify topics with multiple weighted keywords and logical operators, whereas [7] allows only single keyword topics. We believe that our topic model enables users to specify their intent more precisely as a topic. Users can use the weight of a keyword to boost or lower its contribution to the presence measure of a topic. Logical operators can be used to combine all keywords related to a topic as well as those that are unrelated. Third, our model is based on fuzzy set theory in which the presence of a given topic in a time interval is computed. Measuring the presence of a topic in a time interval is an important component of finding hot spots. Different functions to model the presence may result in different amounts, thereby influencing the final outcome. Our fuzzy set theory-based method provides the users the means to explore different options for measuring the presence of a topic. In contrast, Kleinberg [7] uses a simple method of counting all documents containing the topic keyword as the number of relevant documents. Fourth, since our model is based on the temporal scan statistic, users do not specify any parameter value for extracting a hot spot. In contrast, the two-state automaton (multi-state) automaton model needs users to specify two parameters s and γ , where s is used to capture the substructure in a bursty state and γ is used to capture the cost of changing states. If the values of s and γ do not match the distribution of the given data set, the two-state automaton model may not find any bursty intervals or may report too many. We believe that identifying the precise parameters to identify bursty intervals is non-trivial and may require several iterations.

Finally, our method reports as output, a single hot spot, an interval deemed as the most abnormal in terms of the presence of the given topic, whereas the proposed methods in [7] can find multiple bursty intervals. The proposed approach can extract all non-overlapping time intervals that are statistically equally significant. However, extracting top- k intervals that are statistically significant is our future work.

The concepts of scan statistics were introduced by Joseph Naus [26] almost fifty years ago, and were further developed by Kulldorff [27]. This approach is aimed at studying the probabilities and statistical properties of spatial or temporal clusters. It is an important statistical tool to detect an abnormal cluster, and is applied in the public health community to detect trends in outbreaks of disease. The increasing concern over early disease outbreak detection is important for periodic geographical disease surveillance systems and uses scan

statistics extensively for this purpose [28, 29, 30]. Naus and Wallenstein [31] compared two groups of temporal scan statistics models for disease outbreak risks over the time line. A typical outbreak is defined as a spike (burst) standing out on its background. The two models are based on p-values and generalized likelihood ratio to compare the number of events observed in a fixed-size window to the expected number of events. Our model is based on likelihood ratio; it gradually changes the size of the scanning window to find the time interval with an abnormally high presence of events. More current applications in temporal scan statistics are discussed in [32, 33].

Chen and Chundi [1, 2] used temporal scan statistics to identify a hot spot for a given topic, expressed either as a basic or a complex topic. In this work, they introduced the notions of basic topic and complex topic. The temporal scan statistic is exploited to characterize the trends of activity patterns of the topics by Chen and Chundi [3].

Shi and Janeja [34] exploited the scan statistic approach to discover anomalous windows using a technique called Linear Intersecting Paths. In their work, a path is defined as a one-dimensional spatial coordinate marking an observation point. The scanning window travels on a subset of intersecting paths until it reaches the maximum of a likelihood ratio that indicates the degree of unusualness of this window with respect to the rest of the data.

Fuzzy systems is an alternative to the traditional crisp set membership concept which has its origins in ancient Greek culture. Fuzzy set theory was established by L.A Zadeh [35] in 1965 to study sets whose elements have degrees of membership. Fuzzy set theory has been used in a wide range of domains in which information is incomplete or uncertain. The major successful applications of fuzzy sets are in control systems and at the leading edge of Artificial Intelligence. In the past 20 years, fuzzy set theory has captured interest from information related fields such as information retrieval, bioinformatics and data mining. Fuzzy logic systems were applied to the area of information systems in Europe in the early 1990s (see [13]), and they are shown to have better performance with relatively low computation cost. Witte and Bergler [36] provided a clustering algorithm for the analysis of document collections and detection of common and distinctive topics within a document set. The proposed algorithm showed that the clustering algorithm based on fuzzy set theory is easy to implement and has the capability to generate a highly flexible data structure for topic analysis and summarization with excellent performance. Dedek and Vojtas [37] presented a fuzzy system to provide classification of textual web reports. The major contribution of this work is to extract web information using fuzzy inductive logic. Chen et al. [38] proposed an effective Fuzzy-base Multi-label Document Clustering approach to extract the key terms (itemsets) to represent the documents in the set using a fuzzy association rule mining method. Their experimental results showed that the approach had high performance in both generating non-overlapping and overlapping clusters with higher accuracy. The method illustrated in the current paper, to compute a numeric value of relevance of a document set to a multi-keyword topic based on fuzzy set theory, was first discussed by Chen and Chundi [2].

3. Preliminaries

A *time point* is an instance of time with a given *base granularity*, such as a second, minute, day, month, year, etc. A time point can be represented by a single numerical value, specifying a given second, minute, day, etc. A time period T is a sequence of n time points $t_1, \dots, t_n$. An interval T_{ij} ($1 \leq i \leq j \leq n$) of T is a sequence of consecutive time points, starting at time point t_i and ending at time point t_j , in T . Let T_{ij} and T_{kl} be two intervals of T . T_{ij} is contained in T_{kl} if $i \geq k$ and $j \leq l$.

Suppose we observe several events during a time period T . We use the temporal scan statistic [27] to assign a numeric value to describe the amount of an event p during a time interval T_{ij} of T by comparing its presence during T_{ij} to its presence in the rest of the time period. Let m be the number of times p appears in T_{ij} , and b be the total number of times all events appear in T_{ij} . In addition, let M denote the presence of p in the entire time period T ; and B denote the total presence of all events during T . We calculate the *discrepancy score* of p in t using the following formula:

$$ds(m, b) = m \log \frac{m}{b} + (M - m) \log \frac{M - m}{B - b} \text{ if } \frac{m}{b} \geq \frac{M}{B} \quad (1)$$

Let D be a document set spanning the time period T . Let T_{ij} be a time interval of T . Let D_{ij} be the document set published during T_{ij} . Let p be a given topic. Let m_{ij} denote the presence measure of p in D_{ij} . We set M in Equation 1 to the presence measure of p in D and set B to $|D|$. Then, the discrepancy score of T_{ij} with respect to the given topic p is computed as $ds(m_{ij}, |D_{ij}|)$.

A **hot spot** of an event p is a time interval T_{ij} that has the highest discrepancy score.

3.1. Fuzzy Set Theory

Fuzzy set theory was established by L.A Zadeh in 1965 [35]. Fuzzy set theory extends the traditional set theory such that every element has a degree of membership. A fuzzy set F is a pair (A, f) , where A is a traditional set and f is a membership function $f: A \rightarrow [0, 1]$. For each $x \in A$, $f(x)$ is the grade of membership of x that belongs to the fuzzy set. If $A = \{x_1, \dots, x_n\}$ the fuzzy set (A, f) can be denoted as $\{f(x_1)/x_1, \dots, f(x_n)/x_n\}$. For element x_i , if $f(x_i) = 1$, x_i certainly belongs to F , and if $f(x_i) = 0$, then x_i certainly does not belong to F .

Traditional logical operators **and**, **or**, and **not**, denoted by $\wedge$, $\vee$ and $-$ respectively, are defined as follows.

Given two fuzzy sets $B = (A, f_1)$ and $C = (A, f_2)$:

$$B \wedge C = (A, f_3), \text{ where } f_3(x) = \min(f_1(x), f_2(x)) \quad (2)$$

$$B \vee C = (A, f_3), \text{ where } f_3(x) = \max(f_1(x), f_2(x)) \quad (3)$$

$$B - C = (A, f_3), \text{ where } f_3(x) = \max((f_1(x) - f_2(x)), 0) \quad (4)$$

4. Computing the Presence Measure of a Topic

4.1. Presence Measure of Topics

We define a **basic topic** p as a list of pairs, $\{(kw_i, w_i) | 1 \leq i \leq s\}$, where kw_i s are topic keywords and w_i s denote the topic weights, which are the values of each $w_i \in (0, 1]$. We require that the sum of all w_i in a basic topic p is 1. If p contains a single keyword, then the weight of that keyword is 1.

¹Presence measure of a topic is discussed in Section 4.1.

We use function $S_{kw}(p)$ to obtain the set of keywords in the given topic p . Function $Ex(kw_i, d_j)$ denotes whether the keyword kw_i appears in the document d_j ; if it exists, the function returns a 1, otherwise it returns a 0.

Given a basic topic $p = \{ (kw_i, w_i) | 1 \leq i \leq s \}$ and a document d_j , we define a **membership function**, denoted by mb , as follows:

$$mb(p, d_j) = \sum_{i=1}^s w_i Ex(kw_i, d_j) \quad (5)$$

Given a document set D and a topic p , we define the **related document set** $Rt_{p,D}$ as a fuzzy set (D, mb) , where D is the given document set and mb is the membership function defined by Equation 5. For each document $d \in D$, $mb(p, d)$ reflects how a document d in D is related to p .

We define the **presence measure** of a topic p over a document set D as the sum of the membership of each document in D to topic p .

$$m(p, D) = |Rt_{p,D}| = \sum_{d_i \in D} mb(p, d_i) \quad (6)$$

If p is a single keyword, then w_i is 1 and $m(p, D)$ is the number of documents containing the keyword [1]. If document d_i contains all keywords in $S_{kw}(p)$, its $mb(p, d_i) = 1$.

The presence measures of a topic over different document sets qualitatively describe the relevance of the document sets to the specific topic.

A **complex topic** is formed by connecting two or more basic or complex topics with logical operators **and**, **or** and **not**.

Based on the concept of fuzzy logic, we give the formulas to compute the membership function of the related topic set $Rt_{p,D}$, where p is a complex topic, on document set D .

Given topics p_1 and p_2 , $d_j \in D$, when $p = p_1 \wedge p_2$

$$mb(p, d_j) = \min(mb(p_1), mb(p_2)) \quad (7)$$

When $p = p_1 \vee p_2$

$$mb(p, d_j) = \max(mb(p_1), mb(p_2)) \quad (8)$$

When $p = p_1 - p_2$

$$mb(p, d_j) = \min((mb(p_1) - mb(p_2)), 0) \quad (9)$$

The **presence measure** of a complex topic is defined similarly. Let $Rt_{p,D}$ be the related document set of a complex topic p over a document set D . Then the presence measure of p over D is simply the sum of $mb(p, d_j)$ for each $d_j \in D$.

The following is an example that shows how to compute the presence measure of a complex topic using the above formulas:

Example 4.1 (Presence measure of Basic and Complex Topics). Suppose the document set and topics are given as: $D = \{a, b, c, d, e, f\}$, $p_1 = \{(Bill, 0.5), (Clinton, 0.5)\}$, and $p_2 = \{(Clinton, 0.5), (congress, 0.5)\}$.

Suppose Bill is contained by $\{a, b, d, f\}$, Clinton by $\{b, f\}$, and congress by $\{b, d, e\}$.

For basic topics, we get:

$$Rt_{p_1,D} = \{0.5/a, 1/b, 0.5/d, 1/f\}, m(p_1, D) = 3$$

$$Rt_{p_2,D} = \{1/b, 0.5/d, 0.5/e, 0.5/f\}, m(p_2, D) = 2.5.$$

So the presence measure of topic “Bill Clinton” is greater than the presence measure of topic “Clinton congress”.

When using Equations 7, 8, and 9, we get:

$$Rt_{p_1 \wedge p_2,D} = \{1/b, 0.5/d, 0.5/f\}, m(p_1 \wedge p_2, D) = 2,$$

$$Rt_{p_1 \vee p_2,D} = \{0.5/a, 1/b, 0.5/d, 0.5/e, 1/f\}, m(p_1 \vee p_2, D) = 3.5$$

$$Rt_{p_1 - p_2,D} = \{0.5/a, 0.5/f\}, m(p_1 - p_2, D) = 1.$$

Hence, the presence measure of the topic $p_1 \wedge p_2$ is less than the presence measure of the topic $p_1 \vee p_2$, and is equal to the presence measure of $p_1 - p_2$.

We can use the presence measure of a topic on the document set $m(p, D)$ to describe the degree of affinity of document set D to topic p . In the example, $m(p_1, D) > m(p_2, D)$ shows that document set D contains more documents about Bill Clinton; $m(p_1 \wedge p_2, D) > m(p_1, D) > m(p_1 \wedge p_2, D)$ shows that there are more documents containing the topic (Bill Clinton or Clinton congress) than Bill Clinton.

4.2. Computing the Presence Measure of a Topic at a Time Point

To compute the presence measure of a topic in an interval, we first need to compute the presence measure of a topic in every time point of the interval. According to the definition in the previous section, the related document set, $Rt_{p,D}$ of a topic p must be computed first at each time point t using the set of documents appearing in that time point. In $Rt_{p,D}$, every document d is associated with a membership value $mb(p, d)$. The presence measure of the topic p at a time point t is simply the sum of memberships of documents in $Rt_{p,D}$.

We use an inverted index structure to compute the membership values. We assume that the given document set is pre-processed into a structure where each time point is associated with an inverted index over the documents appearing in that time point. Each keyword in the index points to a list containing the document IDs in which it appears.

Suppose p is a basic topic. We simply look up each keyword in the inverted index and collect the related documents into one list which we refer to as the **membership list**. Let $doc-mbv$ denote a pair consisting of $\langle docid, mbv \rangle$, where $docid$ denotes the document's id and mbv denotes the membership value of $docid$ to p . Then, the membership list consists of a series of time points, where each time point connects to a list of $doc-mbv$ pairs. The $doc-mbv$ pairs list of a time point t_i is referred to as $doc-mbv(t_i)$. The presence measure of the basic topic at a time point is the sum of every mbv value in $doc-mbv(t_i)$.

A typical complex topic has the format of $p = p_h \text{ op } p_j$, where p_h and p_j are basic or complex topics. Assume the membership lists of subtopics p_h and p_j are available. For every time point t_i , we first extract the $\text{doc-mbv}(t_i)$ lists of p_h and p_j . If a document id d_k appears in the $\text{doc-mbv}(t_i)$ lists of both p_j and p_h , the equation corresponding to op (refer to Equations 7, 8, 9) is applied to the mb , the membership values of d_k from both lists, and the result is stored as the new membership value of d_k in the new $\text{doc-mbv}(t_i)$ list of the complex topic p . If d_k appears in only one list, the operation is applied to the membership value of d_k and 0. After processing $\text{doc-mbv}(t_i)$ lists of p_h and p_j at each point, we have a new membership list for p with a doc-mbv list for each time point.

4.3. Computing the Presence Measure of a Topic in a Interval

The presence measure of a topic p in a interval T_{ij} is the sum of its presence measure at each time point in T_{ij} . The presence measure of a topic p in each interval can be computed as follows:

We first compute the presence measure of p in each time point t_i by simply summing up the mbv in the $\text{doc-mbv}(t_i)$. We store these values in the mlist of p which stores the pairs (i, m_i) , where i is the ID of the i^{th} time point and m_i is the presence measure value of p at time point t_i .

Based on the mlist , we build a list clist containing cumulative presence measure values. The i^{th} pair in the clist , $\langle i, \text{cm}_i \rangle$, contains the ID i of the time point and the sum of the presence measure values of p from time points 1 through i . Note that clist also contains only those time points in the mlist with non-zero presence measures of the topic. Let $\langle t_1, m_1 \rangle$ be the first tuple in mlist . Then, the first element in clist is $\langle 1, m_1 \rangle$, because $\text{cm}_1 = m_1$. And $\text{cm}_i = \text{cm}_{i-1} + m_i$, where m_i is the presence measure of the topic in the i^{th} time point, showing how the other elements are determined.

We use the clist to retrieve the value of presence measure of the topic in any interval T_{ij} , where the presence measure of topic in T_{ij} is $\text{cm}_j - \text{cm}_{i-1}$.

Building the clist costs $O(n * k)$ running time, where n is the number of time points in the mlist and k is the number of basic topics in the given topic. With the clist , we can retrieve the presence measure of the topic in any interval at constant time, which is $O(1)$.

5. Method to Extract Hot Spots

Let us assume that the time period of the given document T contains n time points. A naive implementation to compute a hot spot for a given topic explores each interval of T , computes the discrepancy score of the interval, and then finds an interval with the maximum discrepancy score. Let b_i denote the number of documents in the time point t_i . For a given topic p , vector m contains the $m_i(p)$ values and vector b contains the b_i values for each of the n time points in T . Figure I outlines the algorithm. The algorithm takes two vectors $m_i(p)$ and b of size n as input.

The above naive implementation contains three nested loops. Variable i takes on values from 1 to n in the outermost loop and fixes a starting for the interval. Variable j takes on values from i to n in the loop nested within the outermost loop and fixes an ending point for the interval. The innermost loop is executed once for each value of k between (including) i and j and computes the m_{ij} and b_{ij} values for interval $[i, j]$. Therefore, the naive implementation takes $O(n^3)$ time to find a hot spot for topic p .

We outline several strategies to improve the efficiency of this implementation. First, we preprocess the *mlists* to the *clists* so that $m_{ij}(p)$ can be accessed in constant time. Similarly, we can access b_{ij} , the number of documents in interval $[i, j]$, in constant time.

As our second strategy to efficiently compute a hot spot, we predict the highest possible discrepancy of an interval and all of its sub-intervals; if the predicted value is smaller than the current maximum value, then we can skip that interval. In order to predict the maximum possible discrepancy value, we prove the following lemma.

Lemma 5.1. *Given an interval T_{ij} such that $m_{ij} \leq M$ and $\frac{m_{ij}}{b_{ij}} \geq \frac{M}{B}$. The discrepancy score of any sub-interval of T_{ij} cannot be greater than $(M - m_{ij}) \log\left(\frac{M - m_{ij}}{B - m_{ij}}\right)$.*

Proof: Consider a sub-interval T_{gh} of T_{ij} where $i \leq g \leq h \leq j$. Then the following conditions are true: $0 \leq m_{gh} \leq m_{ij}$ and $0 \leq b_{gh} \leq b_{ij}$. The discrepancy score decreases when the value of m stays the same, but the base population b increases. Therefore, $ds(m_{gh}, b_{gh}) \leq ds(m_{gh},$

$m_{gh})$. Now we need to show $ds(m_{gh}, m_{gh}) \leq ds(m_{ij}, m_{ij})$ or $\frac{\partial ds(m, m)}{\partial m} \geq 0$ when $0 < m \leq M \leq B$. Let $f(m) = ds(m, m)$, and $f'(m) = \frac{\partial ds(m, m)}{\partial m} = \frac{M - m}{B - m} - \log \frac{M - m}{B - m} - 1$. Let $x = \frac{M - m}{B - m}$. Then, $g(x) = f'(m) = x - \log x - 1$. It is easy to see that $g(x) \geq 0$ when $0 < x \leq 1$. Hence $f'(m) \geq 0$ when $0 < m \leq M \leq B$. Therefore, the discrepancy score of any sub-interval cannot be greater than $ds(m_{ij}, m_{ij})$ which is $(M - m_{ij}) \log\left(\frac{M - m_{ij}}{B - m_{ij}}\right)$.

We describe an algorithm, which we refer to as the *EHE*, *Efficiently Hot Spot Extraction* algorithm, in Figure II. The EHE uses the above observation to estimate the highest possible discrepancy value of an interval which is then used to limit the number of intervals explored. The algorithm takes as input cm and cb . $cb[i]$ is a cumulative vector similar to cm , and contains the number of documents in interval $[1, i]$.

Recall that n denotes the number of time points in the time period T of the document set. The algorithm computes the discrepancy score for each interval T_{ij} with loop variable i starting at 1 and j starting at n . Each time the algorithm predicts the upper bound on the discrepancy score for interval T_{ij} . If the predicted value is smaller than the current maximum, then no sub-interval of T_{ij} needs to be explored further. To facilitate this, the value of j is stored in the variable tmj so that interval T_{itmj} will be skipped for future values of i . The algorithm uses the cumulative vectors to look up the presence measure and total number of documents for each interval efficiently; hence, the running time of the algorithm is $O(n^2)$.

For any time point that contains a zero presence measure of topic p but may contain some baseline value or $m_{ii}(w) = 0$ but $b_{ii} \geq 0$, we call it the zero-presence-measure point. It is easy to show that it always can find a hot spot without a zero-presence-measure point as its starting or ending point.

Lemma 5.2. *Given an interval T_{ij} , it can always find a hot spot that starts and ends with non-zero-presence-measure points.*

Proof: For give time interval T_{ij} , it is assumed that time point i_0 is a zero-presence-measure point, but time point $i_0 + 1$ is not, and the density of time period (i_0, j_0) , $\frac{m_{i_0, j_0}}{b_{ij}} \geq \frac{M}{B}$. So m_{i_0, j_0}

$= m_{(i_0+1)j_1}, b_{i_0j_0} \geq b_{(i_0+1)j_0}$. According to the property of the discrepancy score function, it is decreasing with b if m is fixed, so $ds(m_{i_0j}, b_{i_0j}) \leq ds(m_{(i_0+1)j}, b_{(i_0+1)j})$. In the end, the period which starts with a zero-presence-measure point gets a lower or equal discrepancy score than those that do not, similarly, the period that ends in a zero-presence-measure point is higher or equal to those that do not, and so we can always find a hot spot that starts and ends with a non-zero measure points.

It is obvious to conclude that the zero-presence-measure points can be skipped. Therefore, we do not need to consider intervals beginning or ending at zero-presence-measure points.

Vector cm can be generated from $clist$ and compactly store only those time points with non-zero presence-measure values, thereby reducing the number of time points to be considered, which further improves the running time in the experiments described in the later section.

To capture all intervals with the maximum discrepancy score, only a few changes need to be made to the EHE algorithm. An extra stack is added to store all intervals with the current maximum discrepancy score, which is $dmax$ in Figure II. Once d_{ij} is computed, if $d_{ij} = dmax$, then (i, j) is pushed onto the stack; if $d_{ij} > dmax$, then the stack is cleaned and (i, j) is stored in the stack as the first entry.

6. Simplification of Complex Topics

A topic is either basic or complex; a complex topic is one in which two or more basic topics are connected by logical operators. We assume that a topic is represented as a rooted directed acyclic graph, which we refer to as **topic DAG**². Nodes in the DAG are operators and basic topics in the complex topic. Basic topics are represented as sink nodes in the DAG. There is a directed edge from a logical operator node u to another node v in the DAG if v is an operand of the operator u . Figure Va shows an example. It presents topic DAG of the complex topic $p = (((p1 \text{ and } p2) \text{ not } (p2 \text{ or } p3)) \text{ or } (p2 \text{ or } p3))$, in which $p1, p2$ and $p3$ are basic topics.

A topic DAG represents the computation required to compute the presence measure of the corresponding topic. The presence measure of basic topics at sink nodes can be computed first from the document set and then these values can be propagated up the DAG to the root of the topic to compute the presence measure value of the topic. Therefore, the number of nodes and the structure of the topic DAG represent the number of steps and the order of computation required to compute the presence measure of the topic in each interval.

A complex topic may consist of many basic topics and operators. Different subexpressions of a complex topic may contain the same basic topics and/or a subexpression may appear many times in a complex topic. This means that the same basic topics may appear in different sink nodes of the DAG, and subgraphs of the topic DAG may be the same in terms of the basic topics and operators. We wish to compute the presence measure of a basic topic/subexpression just once and reuse the presence measure value whenever the basic topic/subexpression repeats itself. By simplifying the topic DAG, we are also reducing the number of steps required to compute the presence measure.

We devised a procedure, originating from the idea of the BDD (Binary Decision Diagram) **Reduce** function [15, 16], to simplify a topic DAG into a functionally equivalent topic DAG that does not contain redundant operations. We refer to the simplified topic DAG as a **non-redundant topic DAG**.

²A topic DAG is much like a query tree in database systems [39].

The simplification procedure uses the following equalities to recognize redundant subexpressions.

$$\begin{aligned} mb(p_1 \wedge p_2, d_j) &= mb(p_2 \wedge p_1, d_j), mb(p_1 \vee p_2, d_j) = mb(p_2 \vee p_1, d_j) \\ mb(p_1 \wedge p_1, d_j) &= mb(p_1, d_j), mb(p_1 \vee p_1, d_j) = mb(p_1, d_j) \\ mb(p_1 \wedge \emptyset, d_j) &= \emptyset, mb(p_1 \vee \emptyset, d_j) = mb(p_1, d_j) \\ mb(p_1 - p_1, d_j) &= \emptyset, mb(p_1 - \emptyset, d_j) = mb(p_1, d_j), mb(\emptyset - p_1, d_j) = \emptyset \end{aligned}$$

The simplification procedure takes a topic DAG(p), which is a rooted DAG given o pointer to the root, as input and then generates the corresponding non-redundant topic DAG as output. The procedure first traverses the topic DAG in the depth-first order [40], in which we index a node in the DAG after all its children are indexed. So in the depth-first traversal procedure we start at the root of DAG(p), which is o , and node o will be indexed last. This number is also the size of nodes in DAG(p), say s . We also mark the sink nodes.

With considering an empty node, the algorithm then constructs a two-dimensional table RF with $(s + 1) \times (s + 1)$ entries. The dimensions of RF range from 0 to s . Each entry in RF is a triple. The first value in the triple, denoted $RF[i, j](1)$, records the index of the node in topic DAG that stores the value of i and j . Similarly, the second value in the triple $RF[i, j](2)$ and the third value $RF[i, j](3)$ store the index values of the node that record i or j and i not j . The empty node is indexed as 0. If there is no index node that records these operations in the topic DAG, the triple stores value -1 for the operations.

The simplification procedure initializes the triples in RF as follows. Entries $RF[0, i]$ and $RF[i, 0]$ are all set to $(0, i, 0)$. The diagonal entries in RF are initialized according to the above equalities and all other triples to -1 . Then, every $RF[p, p](1)$ and $RF[p, p](2)$ initiate as p ; $RF[p, p](3)$ always sets to 0; every $RF[p, 0](1)$ and $RF[0, p](1)$ set to 0; every $RF[p, 0](2)$ and $RF[0, p](2)$ set to p ; $RF[0, p](3)$ sets to 0 and $RF[p, 0](3) = p$.

After assigning the index values to each of the nodes in the topic DAG and initializing the RF table, the topic DAG is traversed again in the depth-first manner. In the second visit, sink nodes are ignored. Consider a non-sink node u with index entry u . Let u_l and u_r be the index entries of u 's children and let $u.op$ be the integer corresponding to the operation in u .

In the simplification procedure, $RF[u_l, u_l](1)$ and $RF[u_r, u_r](1)$ are first checked. If $RF[u_l, u_l](1)$ and $RF[u_r, u_r](1)$ are not equal to -1 , the results of u_l and u_r are computed and available in the corresponding nodes. Let $x = RF[u_l, u_l](1)$ and $y = RF[u_r, u_r](1)$, and then we need to check $RF[x, y](u.op)$. If $RF[x, y](u.op)$ is equal to -1 , we copy $RF[u, u](1)$ to $RF[x, y](u.op)$; otherwise we copy $RF[x, y](u.op)$ to $RF[u, u](1)$ and $RF[u, u](2)$. We also need to update $RF[y, x](u.op)$ to the same value when $u.op$ equals 1 or 2. When $u.op = 3$, the procedure updates the values in the same manner, except that $RF[x, y](3)$ is not copied into $RF[y, x](3)$.

In Figure III, we show the algorithm for updating RF entries in detail. Figure IV gives the procedure of simplification of a given topic DAG(p). We also give a concrete example to illustrate the effect of the simplification procedure. Figure V shows the topic DAG(p) and its simplified DAG(p), and the details of RF entries are in Figure VI. Each entry $[i, j]$ in RF is stored as a triple (op_1, op_2, op_3) . op_1 stores the value for i and j , op_2 for i or j , and op_3 for i not j . In Figure VI, we showed -1 value as a blank in the triple in RF . For example, in Figure VIb $RF[2, 4]$ reads as $(., 5)$ meaning $(-1, -1, 5)$. The entry $RF[7, 7]$ is $(5, 5, 0)$, which means that node 7 can reuse the result from node 5; therefore, node 8, which is waiting for node 7, can get the result from node 5 directly. This is shown in Figure Vc. The

simplification algorithm helps reduce redundant nodes in the topic DAG, which saves time in generating the membership list of the topic.

7. Experiments

7.1. Data Sets and System Setup

We considered two data sets in our experiments. The first data set contains all titles of papers published in the ACM SIGMOD and VLDB conferences from 1975 to 2007. There are 33 time points in the data set, one per year. The data set contains about 4,500 titles altogether with about 4,200 distinct keywords. More than half of these keywords appear in a single year or time point. Only about 200 keywords appear in at least 14 time points. Figure VIIa shows the distribution of keywords over time points. The X-axis in Figure VIIa plots the minimum number of time points in which a keyword appear, and the Y-axis plots the number of keywords.

The second data set is a subset of the TDT-Pilot corpus [17]. The TDT-Pilot corpus includes approximately 16,000 stories from July 1, 1994 to June 30, 1995. About half of the stories collected are from Reuters Newswire, and half are from CNN broadcast news transcripts. This data set contains 365 time points. We separate the titles of news in CNN from Reuters. The Reuters set contains 7965 stories with 7594 valid titles, 371 blank titles and 8745 distinct keywords. The distribution of these keywords at different time points is shown in Figure VIIb. The CNN data set contains 7898 stories with 7898 valid titles and 9043 distinct keywords. The distribution of these keywords in different time points is shown in Figure VIIc. In our experiments, we only count the valid titles that are tagged as “title” in data files and the distinct keywords that are not English stop words.

We used a Red Hat Linux System with a dual core Intel Xeon 3.73GHz processor with 3GB of RAM to conduct the experiments. We implemented the algorithms described in Section 4 in Java. The hot spot extraction in Section 5 was implemented using the R Project (<http://www.r-project.org>).

7.2. Naive versus EHE

In Section 5 we designed the EHE algorithm to improve the naive one. Unlike the naive algorithm, the EHE algorithm skips the time intervals that cannot contribute to the highest discrepancy scores, and hence cannot be hot spots.

We define a simple notion called “percentage of efficiency” to compare these algorithms. The percentage of efficiency is the percentage ratio of the number of intervals explored by the EHE algorithm to the number of intervals explored by the naive algorithm. The smaller the percentage of efficiency is, the larger the improvement of performance is in the EHE algorithm compared to the naive one.

In Figure VIII(a–c), the x-axis plots the minimum time points a keyword appears, while the y-axis plots the average percentage of efficiency if the keyword appears at least a certain number of time points. For example, from Figure VIIIa, in the conference paper collection, the percentage of efficiency for keywords appearing in least 14 time points is about 40%, which means that for these keywords, the EHE algorithm scanned only 40% of intervals as the naive algorithm did.

For Reuters and CNN data, the smallest improvement in efficiency (about 30%) with the EHE algorithm occurs when keywords appear in at least 256 and 128 days respectively. The percentage of efficiency gains is maximum (about 0.06%) for the EHE algorithm when we

consider all the keywords in the data set, that is, keywords appearing in at least one time point.

It can be seen from Figure VIII that the performance of the EHE algorithm is indeed much better than the naive algorithm. However, the percentage of intervals explored by the EHE gradually increases as the minimum time points increases. The reason why the value of percentage of efficiency increases with time point increments is that the prediction process finds it harder to skip intervals if the keywords are distributed over many time periods. These initial results show that the performance of the EHE algorithm is quite encouraging for real-world data sets where it is common for keywords/topics to appear in few time points.

7.3. Study of Effectiveness of the EHE Algorithm

In order to study the effectiveness of our temporal scan statistic-based method for capturing hot spots, we conducted the following two studies: 1) we used a synthetic data set to study the accuracy of our algorithm in identifying the hot spots and 2) we computed hot spots of single keywords used in [7] and compared the outputs.

7.3.1. EHE Algorithm on a Synthetic Data Set—We suppose there are 100 consecutive time points; for every time point i , $b[i]$ is an independently Poisson random variable with the mean set as 100. In every test, we randomly select a time interval $[st, ed]$ as a preset hot spot; r is a vector, where $r[i] = R1$ if $i \in [st, ed]$, and $r[i] = R2 = 0.02$ if $i \notin [st, ed]$. Then, for each time point i , the presence measure $m[i]$ is an independently Poisson random variable with the mean at $b[i] * r[i]$. The accuracy is defined as a percentage ratio n/N , where N is the number of testing times, and n is the number of times that the outputs from the EHE have more than 90% overlapping with preset $[st, ed]$. A higher accuracy is better. All test results are shown in Figure IX, and from the results, it can be concluded that our algorithm is highly accurate.

First, we fixed $R2$ as 0.02, and changed $R1$ from 0.04 to 0.14. In Figure IX(a), the x-axis plots the setting of $R1$, such that the mean of the random variable $m[i]$ inside of the preset hot spot is $b[i] * R1$; the y-axis plots a corresponding accuracy value. As can be seen from the figure, the accuracy of the algorithm increases with the value of $R1$. That is, the larger contrast the presence measure of the topic in the hot spot is compared to the outside of it, the higher the accuracy of the EHE algorithm.

Secondly, we set $R1$ to 0.05 (and 0.1) and varied the size of the preset hot spot. In Figure IX(b), the x-axis plots the size of the preset hot spots; the y-axis plots the corresponding accuracy value. The figure shows that the accuracy is always higher when $R1$ is larger and confirms the conclusion of the first test. In the case when the value of $R1$ is small or when the size of the preset hot spot is small, the accuracy of the EHE algorithm falls as low as 50%. The low accuracy is due to our own stringent constraint that the hot spot detected by the EHE algorithm must be 90% overlapping with the preset hot spot. This constraint can be relaxed depending on user requirements. The 90% overlap requirement may be hard to satisfy for small hot spot size or low $R1$. For example, if a preset hot spot is $[34 - 37]$ and the EHE algorithm outputs $[34 - 36]$, the overlapping is 75%, which does not meet the 90% overlap constraint. When we relax the constraint to at least 50%, the accuracy jumps to over 90% in all cases.

7.3.2. Comparison with Previous Work—In this section, we compare the hot spots extracted from our method with the results in [7]. We first give simple examples to compare the output from both methods. We then compare the output on the conference data set computed by our method to that illustrated in [7].

We used the two-state automaton model defined on Page 14 of [7] with parameter γ set to value 1. Let us say that we have 6 time points with 20 documents each. Assume that the number of relevant documents for the given topic is 5 in time points 1 and 3. In the rest of the four time points, the number of relevant documents is 0. For this example data set, the two-state automaton model marks time points 1 and 3 as having the burst, whereas our model returns the time interval [1, 3] as the hot spot. Now, consider another example where the total number of documents in each time point is unchanged; however, the number of relevant documents is now set to 1 in time points 1, 3, and 5. In this case, the two-state automaton model reports that there are no bursts, whereas our model correctly reports [1, 5] as the hot spot. The two-state automaton model recognizes the burst if γ is set to 0.01.

We used the conference data set and the EHE algorithm to extract the hot spots for the set of topics mentioned in [7]. There were 28 topics in the set. Each topic in the set treated as a basic topic containing only one keyword. In Figure X, column 1 lists all of the 28 topic keywords. Column 2 lists the hot spots of keywords when the time span is set to 1975–2001 (same as that used in [7]), whereas Column 3 lists the same when the time span is set to 1975–2007. Columns 4 and 5 show the hot spots of the 28 keywords in individual VLDB and SIGMOD data sets, respectively, for the time span 1975–2007.

The hot spots extracted by the EHE algorithm were exactly the same as those reported in [7] for about 9 topics. For the rest of the topics, the EHE algorithm extracted coarser (or longer) hot spots than those reported in [7]. The EHE algorithm uses the ratio of the presence measure of the given topic, m , to the total number of documents, b , in a time interval/point to identify the interval with the highest discrepancy score. The value of m/b tends to be higher in a hot spot when compared to the rest of the time period. Upon examining the m/b values of a topic over all time points of in the time period, we concluded that the EHE algorithm performs better in data sets where m/b value of a topic over the time period has many spikes. In that case, the hot spots extracted by the EHE algorithm tend to include nearby time points with high m/b values, whereas the methods in [7] ignore these interesting time points. Figure XI illustrates two examples to highlight this point. In the figure, the x-axis plots the time points from 1975 to 2001; the y-axis plots the ratio of m/b of topic **database** and topic **relational** at each time point. For topic **relational**, both methods output the same time interval 1975–1989. For topic **database**, the automaton model outputs the interval from 1982 to 1987; our method is from 1977 to 1995 and includes other time points with relatively a high value of m/b .

We also extracted the hot spots of the same 28 topics with the time span expanded to 2007. The results are listed in Column 4. It is interesting to observe that, for popular topics such as *mining*, the hot spot in the time period 1975–2001 is covered by the hot spot of the same topic in the expanded time period.

7.4. Hot Spots of Topics

In this section, the experiments were designed to test how well our method extracts hot spots of multi-keyword topics, both basic and complex topics. For the conference data set, we constructed several topics and listed the results of four basic topics and two complex topics in this paper. Every basic topic contained two keywords which were evenly weighted; the complex topics contained two or more basic topics connected by operators. Figure XII lists these hot spots from VLDB and SIGMOD. It is easy to observe from the figure that most of the hot spots from VLDB overlap with those from SIGMOD. The only exception is the topic **conceptual schema** and **indexing model**, whose hot spot from VLDB is much earlier than that from SIGMOD.

All the basic topics, such as **conceptual schema** and **xml query**, have almost the same hot spots in both VLDB and SIGMOD. Based on these experiments, we observed that the hot spots from the VLDB data set and the SIGMOD data set were likely the same when the *and* operator is used to combine topics.

We also compared the hot spots of complex topics with those of the basic topics to build them. As we can see in Figure XII, topic **semantic object** has an earlier hot spot than topic **xml query** in both data sets. The hot spot of topic **semantic object or xml query** includes the hot spots of both the topics in both data sets. This can be explained by the function used to compute the topic presence measure for *or* operator. On the other hand, the *and* operator works as a filter, and removes the information that only appears in one basic topic. For topic **conceptual schema and indexing model**, the hot spot in VLDB is 76–85, which overlaps with the hot spot of **conceptual schema**, but does not overlap with the hot spot of **indexing model**. This is because **conceptual schema** does not appear much during the hot spot of topic **indexing model** and contributed to the low topic presence measure of the complex topic **conceptual schema and indexing model** in this time period.

For the TDT-pilot corpus data set, we separated the titles of news into two parts according to their sources. We chose three basic topics and three complex topics, and the results are shown in Figure XIII. Every basic topic contained at least two distinct keywords and was evenly weighted.

As we mentioned in the previous section, the simplification procedure described in Section 6 would be activated if the query topic was a complex topic. In the experiments, to test the simplification procedure, we constructed a complicated complex topic: (**Carter Bosnia or Bosnia war**) or (((**negotiate and Carter Bosnia**) or (**Bosnia war or Carter Bosnia**)) not ((**Bosnia war or Carter Bosnia**) or (**Carter Bosnia and negotiate**))). This topic contained nine operators and three basic topics. The simplification procedure produced a non-redundant topic DAG which is simply **Carter Bosnia or Bosnia war** and contains only one operator, thereby saving a lot of effort in computing the presence measure of the topic.

We found interesting phenomena when we compared the hot spots of the basic topics and the complex topics containing those basic topics. For example, topic **Simpson** has hot spots 01/12/95-03/06/95 and 07/02/94-07/28/95 in Reuters and CNN, respectively. If we check the hot spots of the topic **Simpson trial** in Figure XIII, the hot spot of **Simpson** always contained the hot spot of **Simpson trial** in both news sources, although the two sources have completely different hot spots. This is because the set of documents related to **Simpson trial** must be a subset of documents related to **Simpson**.

We also carefully studied the effect of operators on hot spots of topics in Reuters and CNN data, and found that the effects are similar to those in the conference data.

7.5. Comparison with Vector Space Model

Our experiments also explored the relationship between the similarity search and hot spots. The similarity score is computed commonly by using a standard vector model. If the hot spot of a topic contains an unusual presence of the topic, it seems reasonable to expect that the documents in the hot spot contain a significant amount of relevant information on the topic.

We used Apache Lucene [18] to index the document titles and performed queries to get the top ranked documents using topics as inputs. Then we examined the time stamps of the results to determine how many of them were in the intervals corresponding to the hot spots.

For the conference data set, we treated the titles as documents from the SIGMOD and VLDB data separately. Seven basic topics including three single-keyword and four multi-keyword topics were sent to the search algorithm. We then examined the years in which the top five and the top ten ranked documents were published. Figure XIV illustrates the results. Columns 2 and 3 list the numbers of documents within hot spots and have the top five and the top ten ranked similarity scores in VLDB, and Columns 4 and 5 listed the numbers in SIGMOD. For topic **xml** in VLDB, topics **indexing** and **xml query** in SIGMOD, the top five and the top ten documents were all in their hot spots. On the average, more than half the documents from the top five or top ten are in the hot spots of their topics in both conference sources. This shows that the hot spots indeed contain more relevant information in the conference paper collection. This fact confirmed our conjecture about the relationship between hot spots and relevant documents. The exception was topic **indexing model**. The hot spot of **indexing model** does not contain any of the top ranked documents. However, it does contain relevant documents, with a non-zero similarity score in Lucene.

We performed the same tests on Reuters' and CNN's data. The results were shown in a similar format in Figure XV. Comparing the length of the hot spots with the length of the whole time line, we still can conclude that the hot spots contain a significant amount of relevant documents.

It is worth mentioning that all of the documents containing non-zero presence measures to the topic within its hot spot also have non-zero similarity scores in Lucene. These documents may not rank high in the search list due to the ranking model of Lucene. For complex topics containing multiple operators, we did not compare the results of hot spots and those from Lucene. This is due to the difference in the usage of these operators in our system and Lucene.

7.6. Highly Related Keywords

In this section, we use highly related keywords to check the context of the hot spots of a given topic. In this way, we can understand the context within its hot spot.

To collect the highly related keywords of the topic in its hot spot, we use the following simple procedure. First, we collect all documents with non-zero presence measures for the given topic within the hot spot. Then, we find the frequencies of occurrence of all non-topic keywords from these documents. We choose the top five keywords in terms of the occurrence frequency as "highly related keywords" to the given topic in its hot spot.

In Figure XVII(a–b), we display the highly related keywords for all single-keyword topics for the TDT corpus used in the previous section. The highly related keywords are information-rich. For the topic *fire* in Figure XVIIa, the highly related keywords were *kills*, *eve*, *year*, *head* and *children*. From these keywords we guessed that the documents in the hot spot were published in the holiday season around Christmas and New Year, and were about fire-related accidents that killed children. The hot spot of *fire* in Reuters was 12-26-94 to 01-01-09, which can confirm the conjecture about the context of the hot spot. The highly related keyword helped us understand the both hot spot and the topic at the same time.

It is interesting to compare highly related keywords across different news sources. For any given topic, if it has similar hot spots in both news sources, it is likely that it has similar highly related keywords. For example, the hot spot of the topic *Haiti* from Reuters is very similar to one from CNN as can be seen in Figures XIII. It is not surprising to find that four of the top five highly related keywords are the same for both sources. An opposite case is for the topic *japan*. In CNN, its hot spot was 01-16-95 to 01-21-95; the highly related keywords highlighted the context of *quake*, *kobe*, *earthquake*, *osaka* and *efforts*, as we know 01-16-95

is the exact day when the Kobe earthquake happened; however, the hot spot of *japan* in Reuters spanned from 05-15-95 to 06-28-95 and was associated with *car*, *trade*, *row*, *aid* and *talks*, which were totally different from CNN and may reveal more about the car trade at that time.

We selected a few topics from the conference data set and listed them in Figure XVI. Almost all of the topics analyzed contained *data* as a highly related keyword due to the fact that the documents are from conferences related to databases.

8. Conclusion and Future Work

In this paper, we described a method to extract the hot spot of a topic from a time-stamped document set spanning a time period T . We considered basic topics containing lists of weighted keywords as well as complex topics constructed from basic topics and operators. The presence measure of a given topic in a document set is computed as the sum of the degree of memberships of all documents to the topic. We defined the notion of a discrepancy score for each time interval based on the temporal scan statistic. The discrepancy score uses the presence measure of a given topic to detect the high presence of a topic in a time interval compared to its presence outside the interval. The interval of T with the highest discrepancy score is defined as the hot spot of the topic. Efficient algorithms were described to compute the presence measure of a topic in each interval of T and to compute the discrepancy score of all $O(n^2)$ intervals. The EHE algorithm is designed to ignore the time points and intervals with zero/low presence measure value and thereby achieve efficiency in both space and time. Topic DAGs were applied to capture the logical structure of complex topics. A simplification procedure was introduced to remove redundancy within the structure and save computational cost.

Two different data sets – SIGMOD and VLDB conference titles collected from DBLP and the Reuters and CNN news titles collected from the TDT-Pilot Corpus [17] were used to study the effect of different data resources. Different topics were constructed, and their hot spots were extracted from the data sets. A comparison between the naive algorithm and the EHE algorithm showed encouraging improvement in the performance. The simplification procedure efficiently removed duplicated subexpressions within complex topics.

We are currently examining the documents in the hot spot of a topic to study their relevance to the topic keywords. We are also designing methods to identify hot spots of a topic as a document set evolves.

References

1. Chen, W.; Chundi, P. An approach for discovering hot spots of topics from time stamped documents; 2008 SIAM Conference on Data Mining Text Mining Workshop;
2. Chen, W.; Chundi, P. Extracting hot spots of basic and complex topics from time stamped documents; 2009 IEEE Symposium on Computational Intelligence and Data Mining; 2009. p. 125-132.
3. Chen, W.; Chundi, P. Trends analysis of topics based on temporal segmentation; Proceedings of 11th International Conference on Data Warehousing and Knowledge Discovery; 2009. p. 402-414.
4. Chundi, P.; Zhang, R.; Rosenkrantz, DJ. Efficient algorithms for constructing time decompositions of time stamped documents; 16th International Conference on Database and Expert Systems Applications; 2005. p. 514-523.
5. Chundi, P.; Rosenkrantz, DJ. On lossy time decompositions of time stamped documents; Proceedings of the thirteenth ACM international conference on Information and knowledge management; New York, NY, USA: ACM; 2004. p. 437-445.

6. Chundi, P.; Rosenkrantz, DJ. Constructing time decompositions for analyzing time stamped documents; 2004 SIAM International Conference on Data Mining; 2004.
7. Kleinberg J. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*. 2003; 7(4):373–397.
8. Wang, X.; Zhai, C.; Hu, X.; Sproat, R. Mining correlated bursty topic patterns from coordinated text streams; Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining; New York, NY, USA: ACM; 2007. p. 784–793.
9. Dey, L.; Abulaish, M.; Jahiruddin; Sharma, G. Text mining through entity-relationship based information extraction; Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IATW '07; Washington, DC, USA: IEEE Computer Society; 2007. p. 177–180.
10. Wong T-L, Lam W. An unsupervised method for joint information extraction and feature mining across different web sites. *Data Knowl. Eng.* 68; 2009:107–125.
11. Gao W, Tian Y, Huang T, Yang Q. Vlogging: A survey of videoblogging technology on the web. *ACM Comput. Surv.* 2010; 42:15:1–15:57.
12. Peng W-C, Liao Z-X. Mining sequential patterns across multiple sequence databases. *Data Knowl. Eng.* 2009; 68:1014–1033.
13. Baeza-Yates, RA.; Ribeiro-Neto, BA. *Modern Information Retrieval*. ACM Press / Addison-Wesley; 1999.
14. Kulldorff M. Prospective time-periodic geographical disease surveillance using a scan statistic. *Journal of the Royal Statistical Society*. 2001; A164:61–72.
15. Akers SB. Binary decision diagrams. *IEEE Transactions on Computers*. 1978; C-27(6) 509C516.
16. Bryant RE. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*. 1986; 35:677–691.
17. TDT-Pilot. Tdt pilot study. <http://projects.ldc.upenn.edu/TDT-Pilot/>
18. The apache software foundation, lucene. <http://lucene.apache.org/>
19. Fung, GPC.; Yu, JX.; Yu, PS.; Lu, H. Parameter free bursty events detection in text streams; VLDB '05: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment; 2005. p. 181–192.
20. He, Q.; Chang, K.; Lim, E-P.; Zhang, J. Bursty feature representation for clustering text streams; Proceedings of the Seventh SIAM International Conference on Data Mining 2007; 2007.
21. Mei, Q.; Zhai, C. Discovering evolutionary theme patterns from text: an exploration of temporal text mining; Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining; ACM Press; 2005. p. 198–207.
22. Subasic I, Berendt B. From bursty patterns to bursty facts: The effectiveness of temporal text mining for news. *ECAI*. 2010:517–522.
23. Chen W, Chen C, Zhang L, Wang C, Bu J. Online detection of bursty events and their evolution in news streams. *Journal of Zhejiang University - Science C*. 2010; 11(5):340–355.
24. Kleinberg J. Temporal dynamics of on-line information streams. *Data Stream Management: Processing High-Speed Data Streams*. 2005
25. Kumar R, Novak J, Raghavan P, Tomkins A. On the bursty evolution of blogspace. *WWW*. 2003:568–576.
26. Naus JI. Some probabilities, expectations and variances for the size of largest clusters and smallest intervals. *Journal of the American Statistical Association*. 1966; 61(316):1191–1199.
27. Kulldorff M. A spatial scan statistic. *Communications in Statistics: Theory and Methods*. 1997; 26 1481C1496.
28. Agarwal, D.; McGregor, A.; Phillips, JM.; Venkatasubramanian, S.; Zhu, Z. Spatial scan statistics: approximations and performance study; Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining; New York, NY, USA: ACM; 2006. p. 24–33.
29. Neill, DB.; Moore, AW. *Advances in Neural Information Processing Systems* 16. MIT Press; 2003. A fast multi-resolution method for detection of significant spatial disease clusters; p. 651–658.

30. Neill, DB.; Moore, AW.; Pereira, F.; Mitchell, T. Advances in Neural Information Processing Systems 17. MIT Press; 2005. Detecting significant multidimensional spatial clusters; p. 969-976.
31. Naus J, Wallenstein S. Temporal surveillance using scan statistics. *Statistics in Medicine*. 2006; 25:311–324. [PubMed: 16345054]
32. Burkom HS. Biosurveillance applying scan statistics with multiple, disparate data sources. *Journal of Urban Health: Bulletin of the New York Academy of Medicine*. 2003; 80(2):57–65.
33. Woodall WH, Marshall JB, Joner MD, Fraker SE, Abdel-Salam A-SG. On the use and evaluation of prospective scan methods for health-related surveillance. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*. 2008; 171(1) 223C237.
34. Shi, L.; Janeja, VP. Anomalous window discovery through scan statistics for linear intersecting paths (sslip); *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*; New York, NY, USA: ACM; 2009. p. 767-776.
35. Zadeh L. Fuzzy sets. *Information Control*. 1965; 8:338–353.
36. Witte, R.; Bergler, S. Fuzzy clustering for topic analysis and summarization of document collections; *Proceedings of the 20th Canadian Conference on Artificial Intelligence, LNAI 4509*; Springer; 2007. p. 476-488.
37. Dedek, J.; Vojtas, P. Fuzzy classification of web reports with linguistic text mining; *WI-IAT '09: Proceedings of the Joint Conference on Web Intelligence and Intelligent Agent Technology*; Washington, DC, USA: IEEE Computer Society; 2009. p. 167-170.
38. Chen C-L, Tseng FS, Liang T. An integration of wordnet and fuzzy association rule mining for multi-label document clustering. *Data & Knowledge Engineering*. 2010; 69(11):1208–1226. special issue on contribution of ontologies in designing advanced information systems.
39. Lewis, PBA.; Kifer, M. *Database And Transaction Process*. Addison-Wesley; 2002.
40. Cormen, TH.; Leiserson, CE.; Rivest, RL.; Stein, C. *Introduction to Algorithms*. Second Edition. The MIT Press; 2001.

Biographies

Parvathi Chundi is an associate professor in the computer science department at the University of Nebraska at Omaha. She obtained her Ph. D from the University at Albany, SUNY and specialized in distributed and replicated database systems. Her current research interests include text mining, temporal mining, and bioinformatics.

Wei Chen received the BS degree from Nanchang University, China, in 2001, and the MS degree in the Department of Computer Science from Tsinghua University, China, in 2005. Currently, she is a Ph.D. candidate at the college of Information Science and Technology, University of Nebraska at Omaha, USA. Her research interests include data mining and text mining.

begin

/ n is the number of time
points in T. */*

dmax = -∞;

for *i = 1 to n do*

for *j = i to n do*

m_{ij} = b_{ij} = 0;

for *k = i to j do*

m_{ij} += m_k;

b_{ij} += b_k;

Compute d_{ij} for (i, j);

if (*d_{ij} ≥ dmax*) **then**

dmax = d_{ij};

hotspot = (i, j);

end

Figure I.

A Naive Implementation for Extracting a Hot Spot for a Given Topic *p*.

```

begin
/ * The  $M$  is total presence measure of topic  $P$  in  $T^*$  /
   $dmax = -\infty$ ;
   $tmj = -1$ ;
  for  $i = 1$  to  $n$  do
    for  $j = n$  downto  $i$  do
       $m_{ij} = cm_j - cm_{i-1}$ ;
       $b_{ij} = cb_j - cb_{i-1}$ ;
      if ( $j > tmj$ ) and ( $(m_{ij}/b_{ij} > M/B)$ ) then
         $dpredict = (M - m_{ij})\log(\frac{M-m_{ij}}{B-m_{ij}})$ ;
        if ( $dpredict < dmax$ ) then
           $tmj = j$ ;
          break;
        else
          compute  $d_{ij}$ ;
          if ( $d_{ij} \geq dmax$ )
             $dmax = d_{ij}$ ;
             $hotspot = (i, j)$ 
      end
    end
  end

```

Figure II.
EHE Algorithm to Compute the Hot Spot of a Given topic

Begin

/ *Given a node u , its children are u_l and u_r * /

$$x = RF[u_l, u_l](1);$$

$$y = RF[u_r, u_r](1);$$

If $(RF[x, y](u.op) = -1)$

Then $RF[x, y](u.op) = u;$

If $u.op$ is 1 or 2

Then $RF[y, x](u.op) = u;$

Else $RF[u, u](1) = RF[x, y](u.op)$

$RF[u, u](2) = RF[x, y](u.op)$

End

Figure III.
Updating RF for Node u

Begin

Index every node in $\text{DAG}(p)$ with the root o ;
Build matrix $RF = \text{int } [0:m, 0:m](1:3)$ and initiate it;
Visit every node u in the $\text{DAG}(p)$ in DFS

If u is not a sink node and all its children are
visited

Then update u in RF

End

Figure IV.
The Algorithm of Simplification

Figure V.
 (a) An Example of a Topic DAG (b) Before Simplification (c) After Simplification

Figure VI.
(a) *RF* of DAG(*p*) Before Simplification (b) *RF* of DAG(*p*) After Simplification

Figure VII.
 (a) Word Distribution of Conference Data (b) Word Distribution of Reuters Data (c) Word Distribution of CNN Data

(a)

(b)

(c)

Figure VIII.

(a) Percentage of Efficiency of Conference Data (b) Percentage of Efficiency of Reuters Data (c) Percentage of Efficiency of CNN Data

(a)

(b)

Figure IX.

(a) Accuracy vs R1 (b) Accuracy vs Size of the Preset Hot Spots

Topics	75-01	75-07	VLDB	SIGMOD
design	75-88	75-88	75-88	75-88
relational	75-89	75-89	75-89	75-89
model	75-92	75-92	75-92	75-92
parallel	89-98	89-98	89-98	89-98
web	97-01	97-07	97-07	97-07
xml	99-01	99-07	99-07	99-07
data	75-81	75-81	75-81	75-77
base	75-81	75-81	75-81	75-80
application	75-81	75-81	75-88	76-78
bases	75-93	75-82	75-95	75-80
schema	75-80	76-80	75-79	76-78
database	77-95	77-95	77-96	77-94
statistical	82-84	81-84	81-84	82-84
nested	87-89	87-89	84-91	87-89
objects	86-97	84-00	84-99	91-91
transaction	79-97	79-97	81-97	85-95
deductive	85-94	86-94	85-94	86-94
object	85-00	85-97	85-00	86-97
similarity	95-01	95-01	95-07	97-07
approximate	98-01	98-01	97-07	98-07
server	91-00	91-00	92-00	91-06
sql	95-01	95-07	96-05	94-05
indexing	88-01	94-07	98-07	94-07
large	75-77	75-77	75-77	88-07
theory	77-84	78-83	78-81	79-84
distributed	77-88	77-88	75-88	78-93
mining	94-01	94-07	97-07	95-05
warehouse	94-01	94-07	95-07	94-01

Figure X.
The Hot Spots of Single-keyword Topics in Conference Data Set

Figure XI.

The Examples in Comparison Between Results of Kleinberg's Model and EHE

Topics	VLDB	SIGMOD
conceptual schema	75-80	76-80
indexing model	90-92	75-91
xml query	01-06	00-07
semantic object	89-96	86-97
conceptual schema <i>and</i> indexing model	76-85	07-07
semantic object <i>or</i> xml query	85-07	82-07

Figure XII.
The Hot Spots of Selected Topics from Conference Data

Topics	Reuters	CNN
Simpson trial	01/14/95--03/06/95	07/02/94--07/11/94
Bill Clinton	02/05/95--04/10/95	08/11/94--08/26/94
Haiti invasion	08/31/94--10/17/94	09/12/94--10/02/94
Bill Clinton <i>and</i> Haiti invasion	09/17/94--09/18/94	09/14/94--09/19/94
Bill Clinton <i>not</i> Haiti invasion	05/22/95--06/07/95	08/11/94--08/26/94
(Carter Bosnia <i>or</i> Bosnia war) <i>or</i> (((negotiate <i>and</i> Carter Bosnia) <i>or</i> (Bosnia war <i>or</i> Carter Bosina)) <i>not</i> ((Bosina war <i>or</i> Carter Bosina) <i>or</i> (Carter Bosnia <i>and</i> negotiate)))	05/25/95--06/09/95	05/26/95--06/07/95

Figure XIII.

The Hot Spots of Selected Topics from the TDT Data Set

Topics	VLDB		SIGMOD	
	Top 5	Top 10	Top 5	Top 10
schema	4	4	2	3
xml	5	10	5	7
indexing	1	6	5	10
conceptual schema	4	7	3	4
indexing model	2	2	0	0
xml query	4	9	5	10
semantic object	2	5	2	5

Figure XIV.
Number of Top Ranked Documents in Hot Spots, Conference Data Set

Topics	Reuters		CNN	
	Top 10	Top 20	Top 10	Top 20
Simpson	1	5	4	13
Simpson trial	7	9	3	3
Bill Clinton	5	7	10	15
Haiti Invasion	7	9	6	12

Figure XV.
Number of Top Ranked Documents in Hot Spots, TDT Data Set

Topics	Related Keyword in Top 5
distributed	database, systems, data, management, base
statistical	database, large, management, system, data
parallel	data, database, systems, processing, shared
mining	data, association, rules, large, system
sql	server, microsoft, database, data, query
warehouse	data, maintenance, system, high, line

Figure XVI.
Top Related Keywords for Conference Data Set

Topics	Related Keyword in Top 5
clinton	bosnia, temporary, leader, talks, japanese
simpson	witness, trial, judge, case, defense
haiti	invasion, police, force, clinton, troops
bosnia	clinton, nato, troops, perry, force
japan	car, trade, row, aid, talks
fire	kills, eve, year, head, children

(a)

Topics	Related Keyword in Top 5
clinton	speech, plan, mexico, strike, baseball
simpson	case, evidence, defense, hearing, judge
haiti	invasion, clinton, troops, forces, military
bosnia	clinton, police, role, troops, nato
japan	quake, kobe, earthquake, osaka, efforts
fire	chelan, washington, school, high, continues

(b)

Figure XVII.

(a) Top Related Keywords in Reuters (b) Top Related Keywords in CNN