

# Constructing Time Decompositions for Analyzing Time Stamped Documents

Parvathi Chundi\*

Daniel J. Rosenkrantz†

## Abstract

Extraction of sequences of events from news and other documents based on the publication times of these documents has been shown to be extremely effective in tracking past events. This paper addresses the issue of constructing an *optimal* decomposition of the time period associated with a given document set, i.e., a decomposition with the smallest number of subintervals, subject to no or limited loss of information. We introduce the notion of the *compressed interval decomposition*, where each subinterval consists of consecutive time points having identical information content. We define optimality, and show that any optimal information preserving decomposition of the time period is a refinement of the compressed interval decomposition. We define several special classes of measure functions (functions that compute the significant information from document sets), based on their effect on the information computed as document sets are combined. These classes are used in developing algorithms for computing an optimal information preserving decomposition of the time period of a given document set. We also define the notion of information loss of a time decomposition of a given document set and give an efficient algorithm for computing an optimal lossy decomposition. We discuss the effectiveness of our algorithms on the Reuters-21578, Distribution 1.0 data set and a subset of Medline abstracts.

Keywords: text mining, time decomposition, optimal information preserving decomposition, optimal lossy decomposition, information loss

## 1 Introduction

**1.1 Background** There is an enormous growth in the information available in text documents, such as news documents, research articles, web documents etc. Application of data mining techniques to extract useful information from these text based resources holds enormous potential and has received a lot of interest recently in both academia and industry. Many of these text based documents have been archived, with time of their publication or creation giving an approximate time of

occurrence for the events/information contained in the document. The time stamps associated with text documents have been employed in discovering how the information in the text documents has evolved over time. In particular, time stamped documents have been effectively analyzed for identifying past trends from patent documents [1], emerging trends from research articles [2], detecting and tracking topics from news articles [4, 5] and for extracting timelines of events from articles [6, 7, 8].

The usual approach for extracting temporal information (such as trends) from text documents has been to construct a time series representing the evolution of the significant keywords/topics in the document set over time. A decomposition of the time period  $T$  spanning a document set is constructed by decomposing it into equal length subintervals. The document set is partitioned into subsets by assigning documents to each subinterval based on time stamps. Text mining functions are applied to each document subset to compute the information deemed significant for the corresponding subinterval. The information computed is a set of keywords/phrases/nouns/topics deemed significant for that subinterval. Then, the information from each subinterval is mapped into the temporal dimension by constructing a sequence of significance values. Trends are identified from these time series using languages such as *SDL* [1, 13], visually or by grouping the consecutive subintervals with related information [2, 6, 7, 8].

The temporal information constructed using the above method is crucially linked to the way the time period is divided into subintervals, since this affects the size of the document set in each subinterval, which in turn affects the significance values of the keywords/topics in each subinterval. Consider a document set in which documents were created daily. Then, how should we partition the time period associated with the document set – one day, one month, or coarser subintervals? Ideally, the temporal trends from a document set can be analyzed by dividing the time period into the shortest possible subintervals (in the above example, days) such that all of the temporal sequence information can be obtained. However, this shortest length decomposition, denoted by  $\Pi_S$ , in which each subinter-

\*Computer Science Department, University of Nebraska at Omaha, Omaha, NE 68106. Email: pchundi@mail.unomaha.edu

†University at Albany, SUNY, Albany, NY. Email: djr@cs.albany.edu. Research supported by NSF Grant CCR-0105536.

val is a single time point, may have too many intervals and prove to be difficult to analyze visually. This decomposition may also contain too much transient information and make it difficult to identify consecutive intervals with similar information. On the other hand, choosing month or year long subintervals may lead to a loss of temporal information that may be unacceptable. However, in some cases, when larger subintervals are considered, it is possible that there may be little or no information loss,

**1.2 Overview of This Paper** Despite the close relationship between the decomposition of the time period associated with a document set and the significant information computed for temporal analysis, the problem of identifying a suitable time decomposition for a given document set does not seem to have received adequate attention. The main objective of this paper is to show how a suitable time decomposition can be obtained for a given document set. Given a set  $D$  of time stamped documents and a measure function  $f_m$  used to identify the significant information from a document set, we give efficient algorithms to compute the *optimal* (i.e., with the smallest number of subintervals) information preserving time decomposition of the time period associated with  $D$  that preserves all of the information in the shortest length interval decomposition. We also give an efficient algorithm for computing an optimal (smallest number of subintervals) time decomposition when some loss of information is permissible, but is constrained to be within a specified bound.

We introduce the notion of the **compressed interval** decomposition which can be computed from the  $\Pi_S$  of the document set by coalescing consecutive time points of  $\Pi_S$  with the same significant information. The relationship between the significant information from a coalesced interval and that of the corresponding time points of  $\Pi_S$  that were coalesced depends on the measure function used. We show that for all measure functions, any optimal information preserving decomposition of the time period associated with the given document set is a refinement (not necessarily proper) of the compressed interval decomposition. Further, we define **stable** measure functions (based on ratio measure functions), and show that for such measure functions the optimal information preserving decomposition is unique and is the same as the compressed interval decomposition. We also define **monotone increasing** measure functions (based on count measures), and present an efficient greedy algorithm to construct an optimal information preserving decomposition from the compressed interval decomposition for such measure functions. We also provide a dynamic programming based algorithm

for computing an optimal information preserving decomposition of the time period for any measure function.

In cases where an optimal information preserving decomposition of the time period may contain too many subintervals, one may wish to construct a decomposition with fewer subintervals. However, coarsening an optimal decomposition might result in some information loss. Time period decompositions can be optimized for one or more parameters such as the number of subintervals in the decomposition, the amount of information loss, the ratio of lengths of the subintervals, etc. We characterize the notions of **information loss** and **variability** of a decomposition. We give an efficient dynamic programming based algorithm for constructing an **optimal lossy** decomposition of the time period associated with the given set of documents, where the number of subintervals is constrained by a user-specified number, and the information loss is minimized. We also discuss how to compute an optimal lossy decomposition with minimal information loss subject to constraints on the number of subintervals and variability.

We considered two document sets, a subset of the Reuters-21578, Distribution 1.0 data set<sup>1</sup> and a subset of Medline<sup>2</sup> abstracts. A ratio measure function and an  $\alpha$  value of 0.25 were chosen for computing the information content from document sets. We then constructed an optimal information preserving decomposition of the time period for each of the data sets. The number of subintervals in the optimal information preserving decomposition for the Reuters data set was 10% less than the size of its  $\Pi_S$ , whereas for the Medline data set, the size of the optimal information preserving decomposition was approximately the same as that of its  $\Pi_S$ . We then constructed a set of optimal lossy decompositions with various values constraining the number of subintervals, and studied the relationship between the number of subintervals and the information loss. Our experiments show how the information loss decreases as the number of subintervals in an optimal lossy decomposition increases.

The rest of the paper is organized as follows. Section 2 provides definitions of time decompositions, information content of time intervals, etc. Section 3 provides a classification of measure functions and defines monotone increasing and stable measure functions. Section 4 provides the definitions of information preserving decompositions and the compressed interval decomposition, and gives efficient algorithms for computing the optimal information preserving decomposition of the time

<sup>1</sup>Available at <http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>.

<sup>2</sup>Available at <http://www.pubmed.org>.

period. Section 5 provides definitions of information loss and optimal lossy decompositions, and gives a dynamic programming based algorithm for constructing an optimal lossy decomposition. Section 6 presents experimental results. Section 7 discusses some related work and Section 8 concludes the paper.

## 2 Preliminaries

### 2.1 Time Points, Intervals and Decompositions

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 could be represented by a single numerical value, specifying a given second, minute, day, etc. Alternatively, a time point could be represented by a calendar value consisting of a tuple of numerical values. For example, (05,12,13,26,04,1999) is a time point with calendar representation that denotes the time instance where seconds = 05, minutes = 12, hours = 13, day = 26, month = 04, and year = 1999. We assume that the time points are always defined over a totally ordered domain of base values, and thus can be compared. A *time interval*, denoted by  $[t_1, t_2)$ , where  $t_1 < t_2$ , is the finite set of base granularity time points  $t$  such that  $t_1 \leq t < t_2$ . The *length* of a time interval  $T$ , denoted by  $|T|$ , is the number of time points in  $T$ . Given two time intervals  $T$  and  $S$ ,  $T$  *overlaps*  $S$  if  $T$  and  $S$  have non-empty intersection of time points. We say that time interval  $S$  *is covered by* time interval  $T$ , denoted by  $S \subseteq T$ , if  $t_1 \leq s_1$  and  $s_2 \leq t_2$ . In that case, we also say that  $T$  *covers*  $S$ . Let  $R = [r_1, r_2)$  and  $S = [s_1, s_2)$  be two intervals.  $S$  *follows*  $R$  if  $r_2 \leq s_1$ . If  $r_2 = s_1$  then  $S$  *immediately follows*  $R$ . If  $S$  immediately follows  $R$ , the **combination** of  $R$  and  $S$  is the interval  $[r_1, s_2)$ , and is denoted by  $R * S$ . Note that the combination operator  $*$  is associative, and is applicable to a sequence of intervals, each of which is immediately followed by the next.

A **decomposition**  $\Pi$  of a time interval  $T$ , is a sequence of time intervals  $T_1, T_2, \dots, T_n$ , such that  $T_{i+1}$  immediately follows  $T_i$  for  $1 \leq i < n$ , and  $T = T_1 * T_2 * \dots * T_n$ . Each  $T_i$  is called a *subinterval*<sup>3</sup> of  $T$ . The *size* of decomposition  $\Pi$  of a time interval  $T$  is the number of subintervals in  $\Pi$ . The time interval associated with a decomposition  $\Pi$  is denoted as  $T(\Pi)$ .

The **longest interval decomposition**  $\Pi_L$  of a time interval  $T$  is the decomposition with a single subinterval, namely  $T$ . The **shortest interval decomposition**  $\Pi_S$  of a time interval  $T$  is the decomposition

<sup>3</sup>Note that because our time points have a base granularity, the time intervals are not dense and contain only a finite number of time points. Hence it is not possible to repeatedly decompose a time interval into multiple subintervals.

with  $|T|$  subintervals, one for each base granularity time point within  $T$ . Given two decompositions  $\Pi_1$  and  $\Pi_2$  of a time interval  $T$ , we say that  $\Pi_1$  is a **refinement** of  $\Pi_2$  if every subinterval of  $\Pi_1$  is covered by some interval in  $\Pi_2$ . Decomposition  $\Pi_1$  is a **proper refinement** of  $\Pi_2$  if  $\Pi_1$  is a refinement of  $\Pi_2$  and  $\Pi_1$  and  $\Pi_2$  are not identical.

We now describe the relationship between time stamped documents and the time points, intervals and decompositions.

**2.2 Document Data** Consider a finite set of documents  $D$  where each document has a *time stamp* denoting its time of creation or publication.

To map these documents to the time domain, we identify a time stamp in a document with a time point. This implies that time stamps in all documents in  $D$  have the same base granularity.

*Example.* Let  $D = d_1, d_2, d_3$ . The time stamps of the documents respectively, represented as calendar values, are  $\langle 24, 5, 99 \rangle$ ,  $\langle 26, 5, 99 \rangle$ ,  $\langle 19, 6, 99 \rangle$ . The base granularity of these time stamps is days.

Documents can be assigned to a given time interval in a straightforward manner. A document is assigned to a time interval if and only if the time stamp in the document is a time point in the interval. The documents assigned to an interval  $T$ ,  $Docs(T) = \{d_i | \text{time stamp of } d_i \in T, d_i \in D\}$ . If a time interval has no documents assigned, then it can be safely removed from the time decomposition with no effect on the temporal information gathered from the document set.

**PROPOSITION 2.1.** *If  $T_1 \subseteq T_2$  then  $Docs(T_1) \subseteq Docs(T_2)$ .*

Documents are assigned to a decomposition of  $T$  by simply assigning documents to the individual subintervals.

Note that the time stamps in  $D$  define a time interval  $T_D$  corresponding to the minimum time stamp  $t_{min}$  and the maximum time stamp  $t_{max}$  of documents in  $D$ ,  $T_D = [t_{min}, t_{max} + 1)$ .  $T_D$  has the property that  $Docs(T_D) = D$ .

For any time interval  $T \subseteq T_D$ ,  $Docs(T) \subseteq Docs(T_D)$ . Any decomposition of  $T_D$  produces a partition of  $D$ .

### 2.3 Mapping Document Information to Time by Text Mining

The information corresponding to each interval  $T_i$  is computed by applying text mining methods to  $Docs(T_i)$ . The text mining techniques typically identify a set of *significant* keywords or phrases

appearing in the input document set based on some criterion such as occurrence frequency being above a specified threshold, or top  $k$  keywords/phrases, etc., and return as output the set of significant keywords/phrases. We refer to the function used to compute the significance of keywords as the *measure function* ( $f_m$ ). Formally, given a keyword  $w_i$  and a document set  $D$ ,  $f_m(w_i, D) = v$  where  $v \in R^+$ . A function called the *textmine* function is defined below to capture the text mining techniques in a generic way. Without loss of generality, we assume that the output of a *textmine* function is a set of keywords.

DEFINITION 2.1. *textmine*( $D, f_m, \alpha$ ), where  $D$  is a set of documents,  $f_m$  is a measure function, and  $\alpha \in R^+$  is a user specified real valued threshold, is the set of keywords  $S$  appearing in  $D$  such that  $w_i \in S$  if and only if  $f_m(w_i, D) \geq \alpha$ .

Note that the output of the *textmine* function on a given document set  $D$  may be different for different measure functions and different values of  $\alpha$ .

DEFINITION 2.2. The **information content** of a document set  $D$  for a given measure function  $f_m$ , denoted by  $I_\alpha(D, f_m)$ , is *textmine*( $D, f_m, \alpha$ ).

The *information content* of a time interval is the information content of the document set assigned to it.

DEFINITION 2.3. Given a time interval  $T$ ,  $I_\alpha(T, f_m) = I_\alpha(\text{Docs}(T), f_m)$ .

DEFINITION 2.4. The **information content** of a decomposition  $\Pi = T_1 * \dots * T_n$ , denoted as  $I_\alpha(\Pi, f_m)$ , is  $\bigcup_{i=1}^n I_\alpha(T_i, f_m)$ .

Note that  $I_\alpha(\Pi, f_m)$  is not necessarily equal to  $I_\alpha(T(\Pi), f_m)$ . (Recall from Section 2.1 that  $T(\Pi)$  is the time interval associated with the decomposition  $\Pi$ .)

### 3 Properties of the Measure Function

In this section, we examine the information computed by *textmine* functions for a time interval and how this information is related to that of its decomposition. There are many possible types of measure functions that can be used to compute the significant keywords from a document set. The information content for a given document set computed by different measure functions may be different. And, these functions may behave differently as the document set sizes change. The typical measure functions used in the literature to compute significance can be partitioned into the following three types.

- **Count measure** ( $f_{mc}$ ): Includes all measure functions that base significance on the number of occurrences of a keyword. For example, a keyword is deemed to be significant if the number of its occurrences in a given document set is above some threshold  $\alpha$ . A related measure is to deem a keyword to be significant if the number of documents in which it appears is above threshold  $\alpha$ .
- **Ratio measure** ( $f_{mr}$ ): This category contains all measure functions that compute significance as a normalized value. All keywords with the corresponding normalized values above some threshold  $\alpha$  are termed significant. Examples of such functions are significance computed as the ratio of occurrence frequency of a keyword to the total number of occurrences of words in the document set or as a ratio of number of documents in which a keyword appears to the total number of documents in a given document set, etc.
- **top- $\alpha$  measure** ( $f_{mt}$ ): Represents the category of measure functions that compute the significance of a keyword using one of the above measures and return the top  $\alpha$  keywords as output. Examples include returning the top 100 significant keywords when ranked according to the frequency with which they occur in the document set. Note that an issue in defining the measure is if there is a tie for the top  $\alpha$  keywords, whether to include all the keywords involved in the tie, or just some of them. We assume that the output of top- $\alpha$  includes all the keywords involved in a tie.

DEFINITION 3.1. We say that a measure function  $f_m$  is **monotone increasing** if, given two document sets  $D_1$  and  $D_2$  such that  $D_1 \subseteq D_2$ ,  $I_\alpha(f_m, D_1) \subseteq I_\alpha(f_m, D_2)$ .

The following observation holds for any monotone increasing measure function.

PROPOSITION 3.1. Let  $f_m$  be any monotone increasing measure function.

1. Given two disjoint document sets  $D_1$  and  $D_2$ ,  $I_\alpha(D_1 \cup D_2, f_m) \supseteq I_\alpha(D_1, f_m) \cup I_\alpha(D_2, f_m)$ .
2. Let  $D$  be a document set, with time interval  $T_D$ , and let  $\Pi$  be a decomposition of  $T_D$ . Then,  $I_\alpha(T_D, f_m) \supseteq I_\alpha(\Pi, f_m)$ .

We note that count measure functions are monotone increasing.

PROPOSITION 3.2. Any count measure function  $f_m$  is monotone increasing.

The following lemmas hold for textmine functions with a ratio measure function  $f_{mr}$ , or top- $\alpha$  measure function  $f_{mt}$ .

LEMMA 3.1. *Let  $f_{mr}$  be any ratio measure function. Let  $D_1$  and  $D_2$  be two disjoint document sets. Then  $I_\alpha(D_1, f_{mr}) \cap I_\alpha(D_2, f_{mr}) \subseteq I_\alpha(D_1 \cup D_2, f_{mr}) \subseteq I_\alpha(D_1, f_{mr}) \cup I_\alpha(D_2, f_{mr})$ .*

$$1. I_\alpha(D_1, f_{mr}) \cap I_\alpha(D_2, f_{mr}) \subseteq I_\alpha(D_1 \cup D_2, f_{mr}).$$

*Proof.* : Let  $w_p \in I_\alpha(D_1, f_{mr}) \cap I_\alpha(D_2, f_{mr})$ . Let the number of keyword occurrences in  $D_1$  be  $n_1$  and the number of keyword occurrences in  $D_2$  be  $n_2$ . Then, the number of keyword occurrences in  $D_1 \cup D_2$  is  $n_1 + n_2$ . Let  $v_1$  ( $v_2$ ) be the number of occurrences of  $w_p$  in  $D_1$  ( $D_2$ ). Then,  $v_1 = \alpha * n_1 + \epsilon_1$  and  $v_2 = \alpha * n_2 + \epsilon_2$  where  $\epsilon_1, \epsilon_2 \geq 0$ . Then,  $f_{mr}(w, D_1 \cup D_2) = (v_1 + v_2)/(n_1 + n_2)$  which is  $(\alpha * n_1 + \epsilon_1 + \alpha * n_2 + \epsilon_2)/(n_1 + n_2) = \alpha + (\epsilon_1 + \epsilon_2)/(n_1 + n_2) \geq \alpha$ . Therefore,  $w_p \in I_\alpha(D_1 \cup D_2, f_{mr})$ .

$$2. I_\alpha(D_1 \cup D_2, f_{mr}) \subseteq I_\alpha(D_1, f_{mr}) \cup I_\alpha(D_2, f_{mr}).$$

*Proof.* : Consider  $w_p \notin I_\alpha(D_1, f_{mr}) \cup I_\alpha(D_2, f_{mr})$ . In this case,  $v_1/n_1 < \alpha$  and  $v_2/n_2 < \alpha$ . Hence,  $v_1 = \alpha * n_1 - \epsilon_1$  and  $v_2 = \alpha * n_2 - \epsilon_2$  where  $\epsilon_1, \epsilon_2 > 0$ . The ratio of  $w_p$  in the union of  $D_1$  and  $D_2$  is  $(v_1 + v_2)/(n_1 + n_2)$ . By substituting for  $v_1$  and  $v_2$  in this equation, we get  $(\alpha * n_1 - \epsilon_1 + \alpha * n_2 - \epsilon_2)/(n_1 + n_2) = \alpha - (\epsilon_1 + \epsilon_2)/(n_1 + n_2) < \alpha$ . Therefore,  $w_p \notin I_\alpha(D_1 \cup D_2, f_{mr})$ .

DEFINITION 3.2. *We say that a measure function  $f_m$  is **stable** if, given two disjoint document sets  $D_1$  and  $D_2$  such that  $I_\alpha(D_1, f_m) = I_\alpha(D_2, f_m)$ ,  $I_\alpha(D_1 \cup D_2, f_m) = I_\alpha(D_1, f_m)$ .*

LEMMA 3.2. *The ratio measure function  $f_{mr}$  is stable.*

*Proof.* : Follows from Lemma 3.1.

LEMMA 3.3. *The top- $\alpha$  measure function  $f_{mt}$  is stable.*

*Proof.* : Consider any top- $\alpha$  measure function  $f_{mt}$ . Let  $D_1$  and  $D_2$  be disjoint document sets such that  $I_\alpha(D_1, f_{mt}) = I_\alpha(D_2, f_{mt})$ . Consider any keyword  $w_p$  in  $I_\alpha(D_1, f_{mt})$  and any keyword  $w_q$  not in  $I_\alpha(D_1, f_{mt})$ . Since  $I_\alpha(D_1, f_{mt}) = I_\alpha(D_2, f_{mt})$ ,  $w_p$  belongs to the information contents of both  $D_1$  and  $D_2$ , and  $w_q$  belongs to neither information content. Consequently, the significance value of  $w_p$  is strictly greater than that of  $w_q$  in both document sets  $D_1$  and  $D_2$ . Therefore, the significance value of  $w_p$  in  $D_1 \cup D_2$  is also strictly greater than that of  $w_q$ . Since this is true for  $w_p$  and for all  $w_q$ ,

not in  $I_\alpha(D_1, f_{mt})$ ,  $w_p$  is in  $I_\alpha(D_1 \cup D_2, f_{mt})$ . Since it is true for  $w_q$  and for all  $w_{p'}$  in  $I_\alpha(D_1, f_{mt})$ ,  $w_q$  is not in  $I_\alpha(D_1 \cup D_2, f_{mt})$ . Consequently,  $I_\alpha(D_1 \cup D_2, f_{mt}) = I_\alpha(D_1, f_{mt})$ .

The following example shows that the count measure function is not stable.

*Example.* Let  $D_1$  be a document set with three documents  $\{d_{11}, d_{12}, d_{13}\}$ . Let  $D_2$  be another document set with  $\{d_{21}, d_{22}, d_{23}\}$ . Let the contents of the documents be as follows.

$$\begin{aligned} d_{11} &= \{w_a, w_b, w_c\} & d_{12} &= \{w_a, w_d\} & d_{13} &= \{w_b, w_e\}. \\ d_{21} &= \{w_a, w_b\} & d_{22} &= \{w_a, w_g\} & d_{23} &= \{w_b, w_e\}. \end{aligned}$$

Let  $\alpha = 2$ . Then,  $I_\alpha(D_1, f_{mc}) = I_\alpha(D_2, f_{mc}) = \{w_a, w_b\}$ . However,  $I_\alpha(D_1 \cup D_2, f_{mc}) = \{w_a, w_b, w_e\}$ .

One can easily define measure functions that are neither stable nor monotone increasing. As an example, let  $f_{m\beta}$  be a measure function that takes as input a keyword  $w_i$  and a document set  $D$  and first computes the document frequency for all keywords that appear in at least two documents. It then returns for each such keyword, the ratio of its document frequency to the total document frequency of all keywords appearing in at least two documents. The following example shows that  $f_{m\beta}$  is neither stable nor monotone increasing.

*Example.*  $D_1$  is a document set with keyword  $a$  appearing in two documents, and keywords  $b$  and  $c$  appearing in one document each.  $D_2$  is a document set also containing keyword  $a$  in two documents, and  $b$  and  $c$  in one document each. Document set  $D_3$  contains only one keyword  $a$  and its document frequency in  $D_3$  is 2.

$$\begin{aligned} f_{m\beta}(a, D_1) &= 1, f_{m\beta}(b, D_1) = 0, f_{m\beta}(c, D_1) = 0. \\ f_{m\beta}(a, D_2) &= 1, f_{m\beta}(b, D_2) = 0, f_{m\beta}(c, D_2) = 0. \\ f_{m\beta}(a, D_3) &= 1. \\ f_{m\beta}(a, D_1 \cup D_2) &= 0.5, f_{m\beta}(b, D_1 \cup D_2) = 0.25 \text{ and} \\ f_{m\beta}(c, D_1 \cup D_2) &= 0.25. \\ f_{m\beta}(a, D_2 \cup D_3) &= 1, f_{m\beta}(b, D_2 \cup D_3) = 0, \text{ and} \\ f_{m\beta}(c, D_2 \cup D_3) &= 0. \\ f_{m\beta}(a, D_1 \cup D_2 \cup D_3) &= 0.6, f_{m\beta}(b, D_1 \cup D_2 \cup D_3) = 0.2, \\ \text{and } f_{m\beta}(c, D_1 \cup D_2 \cup D_3) &= 0.2. \end{aligned}$$

Let  $\alpha = 0.6$ . Then  $I_\alpha(D_1, f_{m\beta}) = I_\alpha(D_2, f_{m\beta}) = \{a\}$ . However,  $I_\alpha(D_1 \cup D_2, f_{m\beta}) = \{\}$ , so  $f_{m\beta}$  is neither stable nor monotone increasing. However, looking at other intervals,  $I_\alpha(D_1, f_{m\beta}) = I_\alpha(D_2 \cup D_3, f_{m\beta}) = \{a\}$ . In this case,  $I_\alpha(D_1 \cup D_2 \cup D_3, f_{m\beta}) = \{a\}$ .

#### 4 Information Preserving Decompositions

Given a document set  $D$  and the time interval  $T_D$  containing the time stamps of all documents in  $D$ , a natural way to analyze the temporal evolution of

information in the document set is to decompose  $T_D$  into its shortest interval decomposition,  $\Pi_S$ . Using the  $\Pi_S$  decomposition explicates the time based evolution of concepts in the document set  $D$ . However, it is not clear whether such a fine-grained decomposition is necessary given that the number of intervals could be very large. On the other hand, using  $\Pi_L$  to capture all periodic activity corresponds to applying text mining techniques without explicitly constructing time sequences and their analysis. The two decompositions  $\Pi_S$  and  $\Pi_L$  define the two endpoints of assigning documents to time intervals, with  $\Pi_S$  being the most refined document assignment and  $\Pi_L$  being the most coarse assignment. Additional decompositions of  $T_D$  which have more time intervals than  $\Pi_L$  and fewer time intervals than  $\Pi_S$  are also possible.

We say that a time interval  $T_i$  is **information preserving** if it has the following property. Let  $\Pi_S(T_i) = ST_{i,1} * ST_{i,2} * \dots * ST_{i,j_i}$  for some  $j_i$ . Then,  $\forall p(1 \leq p \leq j_i) I_\alpha(f_m, ST_{i,p}) = I_\alpha(f_m, T_i)$ . Given a document set  $D$  with associated time interval  $T_D$ , a decomposition  $\Pi$  of  $T_D$  is **information preserving** if each of the subintervals in  $\Pi$  is information preserving. We note that the shortest interval decomposition  $\Pi_S$  of  $T_D$  is an information preserving decomposition of  $T_D$ . We define a decomposition  $\Pi_O$  of  $T_D$  for a given  $D$  to be **optimal information preserving** if it is information preserving and no other information preserving decomposition of  $T_D$  has fewer subintervals. There is not necessarily a unique optimal information preserving decomposition of  $T_D$ , but there is at least one optimal information preserving decomposition.

We first define the compressed interval decomposition of  $T_D$  and show how to construct it. Let  $w_p$  be any keyword from  $D$ . Let  $T_h$  be a time interval, where  $\Pi_S(T_h) = ST_{h,1} * ST_{h,2} * \dots * ST_{h,j_h}$  for some  $j_h$ . Time interval  $T_h$  is defined to be  **$p$ -uniform** for keyword  $w_p$  if either  $w_p \in I_\alpha(f_m, ST_{i,s})$  for all  $s(1 \leq s \leq j_h)$ , or  $w_p \notin I_\alpha(f_m, ST_{i,s})$  for all  $s(1 \leq s \leq j_h)$ . Time interval  $T_h$  is defined to be **maximal  $p$ -uniform** if there is no  $p$ -uniform  $T'_h$  that properly contains  $T_h$ . We define the **compressed interval decomposition** ( $\Pi_C^p$ ) for  $w_p$  as the decomposition of  $T_D$  where each subinterval  $T_h$  is maximal  $p$ -uniform.

Let  $\Gamma = \{w_1, \dots, w_k\}$  be a set of keywords from the document set  $D$ . We can extend the above definitions to  $\Gamma$  as follows. An interval  $T_h$  is  **$\Gamma$ -uniform** if it is  $p$ -uniform for each  $w_p \in \Gamma$ . Interval  $T_h$  is **maximal  $\Gamma$ -uniform** if there is no  $\Gamma$ -uniform  $T'_h$  that properly contains  $T_h$ . We define the **compressed interval decomposition**  $\Pi_C$  for a set of keywords  $\Gamma$  as the decomposition of  $T_D$  where each subinterval  $T_h$  of  $\Pi_C$  is maximal  $\Gamma$ -uniform.

The compressed interval decomposition of  $T_D$  for a keyword  $w_p$ ,  $\Pi_C^p$ , can be constructed from  $\Pi_S$  by simply combining into a single interval each maximal length sequence of consecutive intervals  $ST_k$  in  $\Pi_S$  such that  $w_p \in I_\alpha(ST_k, f_m)$ . Each maximal length sequence of consecutive intervals of  $\Pi_S$  such that  $w_p$  is not in the information content are similarly combined into a single interval. Each  $\Pi_C^p(1 \leq p \leq k)$  is a decomposition of  $T_D$ . The compressed interval decomposition  $\Pi_C$  of all the keywords is the *product*<sup>4</sup> of the  $k$  decompositions  $\Pi_C^p, 1 \leq p \leq k$ . Let  $n$  be the total number of subintervals in all the given  $k$  decompositions. The product decomposition can be computed in time  $O(n \lg k)$ -time [11] in a straight-forward manner as follows. We can represent a decomposition over the given time interval as a sorted list of the start times of the subintervals in the decomposition. Given  $k$  sorted lists, with a total of  $n$  elements, these lists can be merged into a single sorted list, and duplicate values can be eliminated. The final merged list of start times, with duplicates eliminated, can be converted to the standard representation of a decomposition as a list of subintervals. (The key to the merging algorithm is to use a heap.)

We now consider the relationship between the compressed interval decomposition of  $T_D$  and its optimal information preserving decomposition.

**LEMMA 4.1.** *Let  $T_{c1}$  and  $T_{c2}$  be two subintervals of  $\Pi_C$  such that  $T_{c2}$  immediately follows  $T_{c1}$ . Let  $T$  be an interval that overlaps both  $T_{c1}$  and  $T_{c2}$ , i.e.,  $T$  contains both the last time point in  $T_{c1}$  and the first time point in  $T_{c2}$ . Then,  $T$  is not information preserving.*

*Proof.* : Based on the construction of  $\Pi_C$ , there exists at least one keyword  $w_p$  such that  $w_p \in I_\alpha(T_{c1}, f_m)$  and  $w_p \notin I_\alpha(T_{c2}, f_m)$  or vice versa. Therefore, the information content of the subintervals of  $\Pi_S(T)$  do not all have the same information content. Hence,  $T$  is not information preserving,

**THEOREM 4.1.** *Every information preserving decomposition (and consequently every optimal information preserving decomposition) is a refinement of the  $\Pi_C$  for the set of keywords in  $D$ .*

*Proof.* : Consider a decomposition  $\Pi_i$  that is not a refinement of  $\Pi_C$ . Then, there exists a time interval  $T_x$  of  $\Pi_i$  that overlaps  $k$  ( $k > 1$ ) consecutive intervals in  $\Pi_C$ . From Lemma 4.1,  $T_x$  is not information preserving, so  $\Pi_i$  is not information preserving.

<sup>4</sup>The  $\Pi_C^p$  of each word  $w_p$  is a partition of  $T_D$  into subintervals. The product of a set of partitions is the coarsest partition that is a refinement of each of the individual partitions, and is unique [12].

For stable measure functions, the following theorem shows that the optimal information preserving decomposition of  $T_D$  is unique and is the same as the  $\Pi_C$  for the set of keywords.

**THEOREM 4.2.** *Given a document set  $D$ , let  $T_D$  be the time interval covering the time stamps of the documents in  $D$ . If a measure function  $f_m$  is stable, then the compressed interval decomposition  $\Pi_C$  of  $T_D$  is an optimal information preserving decomposition, and is the only optimal information preserving decomposition.*

*Proof.* : Consider any interval  $T_h$  from  $\Pi_C$ . By the construction of  $\Pi_C$ , all subintervals in  $\Pi_S(T_h)$  have the same information content. Let  $\mathcal{W}$  denote this information content. Since  $f_m$  is stable,  $I_\alpha(T_h, f_m) = \mathcal{W}$ . Therefore, interval  $T_h$  is information preserving. Since this holds for every interval in  $\Pi_C$ , decomposition  $\Pi_C$  is information preserving.

Let  $w_1, \dots, w_m$  be the set of keywords from  $D$ . From Theorem 4.1, any optimal information preserving decomposition of  $T_D$  must be a refinement of  $\Pi_C$ . Therefore,  $\Pi_C$  is an optimal information preserving decomposition, and is the only optimal information preserving decomposition.

From Section 3, we know that the information content computed by monotone increasing measure functions from a union of (disjoint) sets of documents, all with the same information content, may be a superset of the information content of each of the individual sets of documents. Therefore, the compressed interval decomposition  $\Pi_C$  for a monotone increasing measure function might not be information preserving. From Theorem 4.1, an optimal information preserving decomposition of  $T_D$  in this case can be computed as a refinement of  $\Pi_C$ , where each of the refined subintervals is information preserving. A greedy algorithm is outlined below in Figure 1 to construct an optimal information preserving decomposition of  $T_D$  for monotone increasing measure functions. The algorithm starts with  $\Pi_C$ . Given  $\Pi_C$ , the number of steps in the remainder of the algorithm is linear in the size of  $T_D$ . The algorithm refines each subinterval of  $\Pi_C$  into a minimum number of subintervals, each of which is information preserving. Note that since there are many different ways in which a subinterval can be made information preserving, the optimal information preserving decomposition of  $T_D$  for monotone increasing measure functions is not unique. We can show that the optimal information preserving decomposition of  $T_D$  computed by the algorithm contains the minimum number of subintervals over all possible optimal information preserving decompositions.

The greedy approach to refining each subinterval of the compressed interval decomposition for constructing

- Let  $\Pi_C$  be the compressed interval decomposition for the set of keywords from the document set  $D$ .
- Repeat for each subinterval  $T_i$  of  $\Pi_C$ 
  - Let  $\Pi_S(T_i) = ST_{i1} * \dots * ST_{ih}$ .
  - Let  $Temp = ST_{i1}$ .
  - for  $l = 2$  to  $h$ 
    - if  $I_\alpha(ST_{il}, f_m) == I_\alpha(Temp * ST_{il}, f_m)$
    - then  $Temp = Temp * ST_{il}$ .
    - else
      - Output  $Temp$  as a new information-preserving interval of  $\Pi_{OPT}$ .
      - Start a new subinterval with  $Temp = ST_{il}$ .
  - Output  $Temp$  as a new information-preserving interval of  $\Pi_{OPT}$ .

Figure 1: An Algorithm for Computing an Optimal Information Preserving Decomposition  $\Pi_{OPT}$  of  $T_D$  for monotone increasing measure functions.

an optimal information preserving decomposition may not work for measure functions such as  $f_{m\beta}$  that are neither stable nor monotone increasing. In general, it is possible that the information content computed from disjoint sets of documents may not have a predictable relationship to the information content of the union of the document sets. In such a case, it may not be enough to examine the intervals just once as is done in Figure 1. We may need to compute all the different ways of refining a given time interval and choose the best one for that interval.

We present a dynamic programming based approach, much like the well known matrix chain multiplication algorithm [11], to computing an optimal information preserving decomposition for a given time interval  $T_D$  and document set  $D$ . The algorithm starts with the the compressed interval decomposition  $\Pi_C$ . Given a subinterval  $T_i$  of  $\Pi_C$ , let  $\Pi_S(T_i) = ST_{i1} * \dots * ST_{ij}$ . Let  $T_i(x, y)$  ( $1 \leq x \leq y \leq j$ ) denote the subinterval of  $T_i$  consisting of  $ST_{ix} * \dots * ST_{iy}$ . The dynamic programming approach determines, for every  $T_i(x, y)$ , the minimum number of subintervals that can occur in an information preserving decomposition of  $T_i(x, y)$ . For each subinterval  $T_i$  of  $\Pi_C$ , the algorithm constructs a

table  $R$  with  $j$  rows and  $j$  column, where  $j$  is the size of  $T_i$ . Each entry  $R[x, y]$  in the table  $R$  for  $T_i$  stores the count of the minimum number of subintervals in an information preserving decomposition of interval  $T_i(x, y)$ . Table  $R$  is used in deciding how to construct an information preserving decomposition of  $T_i$ . More specifically, there is an entry  $R[x, y]$  for each pair of values  $x$  and  $y$  such that  $1 \leq x \leq y \leq j$ . The value of each of these entries is computed as follows.

$$R[x, x] = 1.$$

if  $x < y$ , then

if  $T_i(x, y)$  is information preserving

then  $R[x, y] = 1$

else  $R[x, y] = \min_{x \leq z < y} (R[x, z] + R[z + 1, y])$

The minimum number of subintervals in any information preserving decomposition of  $T_i$  is  $R[1, j]$ . We can use a separate table  $M$  to remember the value of  $z$  that gives the least number of subintervals for each  $R[i, j]$ . Figure 2 gives an outline of our dynamic programming approach to computing an optimal information preserving decomposition of  $T_D$  for any arbitrary measure function. Note, from the above equation for  $R[x, y]$  when  $x < y$ , that the value of  $R[x, y]$  can be computed once the values of the entries in  $R$  for all shorter intervals that are covered by  $T_i(x, y)$  have been computed. Each iteration of the  $l$  loop in Figure 2 computes the value of  $R$  for all intervals of size  $l$ . Since these iterations are in increasing order of  $l$ , when an entry in  $R[x, y]$  is to be computed, the entries for all shorter intervals have already been computed. The algorithm runs in  $O(mj^2)$  time where  $m$  is the number of subintervals  $T_i$  in  $\Pi_C$  and  $j$  is the maximum number of subintervals in the  $\Pi_S(T_i)$ .

## 5 Information Lossy Decompositions

When the time interval  $T_D$  associated with a document set  $D$  is large, an information preserving decomposition may consist of too many subintervals, and this may make it hard to understand the temporal changes in the information. A coarser decomposition may be used, but this may lead to loss of some temporal information. In this section, we consider a notion of information loss and give a method for computing decompositions subject to constraints or optimization goals pertaining to the amount of information loss and the number of intervals in the decomposition.

Informally, the notion of information loss captures the amount of keywords missing (if any) in the information content of a time interval  $T_i$  as compared to the

- Let  $\Pi_C$  be the compressed interval decomposition for the set of keywords from the document set  $D$ .
- Repeat for each subinterval  $T_i$  of  $\Pi_C$ 
  - Let  $\Pi_S(T_i) = ST_{i1} * \dots * ST_{ij}$ .
  - for  $x = 1$  to  $j$ 

$$R[x, x] = 1$$
  - for  $l = 2$  to  $j$ 
    - for  $x = 1$  to  $j - l + 1$ 

$$y = x + l - 1$$
    - if  $I_\alpha(T_i(x, y), f_m) == I_\alpha(ST_{ix}, f_m)$
    - then  $R[x, y] = 1$
    - else  $R[x, y] = \min_{x \leq z < y} (R[x, z] + R[z + 1, y])$

Figure 2: A dynamic programming algorithm for finding an optimal information preserving decomposition of  $T_D$ .

keywords in the information content of its shortest interval decomposition. Given a subinterval  $ST_{ig}$  of  $\Pi_S(T_i)$ , we consider the total number of keywords  $w_j$  such that  $w_j \in I_\alpha(ST_{ig}, f_m)$  and  $w_j \notin I_\alpha(T_i, f_m)$ . We denote this number for  $ST_{ig}$  by  $\mu_g$ . Let  $\Pi_S(T_i) = ST_{i1} * \dots * ST_{ih}$ . Then, the **information loss**, denoted by  $\mu(T_i)$ , of time interval  $T_i$  is defined to be  $\sum_{g=1}^h \mu_g$ . The **information loss** of a decomposition  $\Pi$  is the sum of the information loss of each of its subintervals, and is denoted by  $\mu(\Pi)$ . Information preserving decompositions of a time interval have zero information loss. We call a decomposition with a non-zero information loss a **lossy decomposition**.

We can optimize a lossy decomposition for various parameters, under various constraints. One may be interested in constructing a decomposition with the smallest information loss, subject to a constraint that the number of subintervals cannot exceed a specified value. Similarly, one may be interested in constructing a decomposition with the smallest number of subintervals, subject to a constraint that the information loss cannot exceed a specified value. Computing an optimal lossy decomposition involving these two parameters is an optimization problem where one of the above parameters is specified and the other parameter is to be minimized. For example, we define the **optimal lossy decomposition problem** to be the problem of computing a decomposition with minimum information loss,

given a set of documents  $D$  published during time period  $T_D$ , a measure function  $f_m$ , and a constraint  $s$  on the number of subintervals. Similarly, we can consider the problem of computing a decomposition with a minimum number of subintervals, subject to a given constraint on information loss.

We give a  $O(n^4)$  dynamic programming algorithm in Figure 3 for the optimal lossy decomposition problem where  $n$  is the size of the  $\Pi_S$ . This algorithm computes a decomposition minimizing the information loss subject to a constraint that the number of subintervals in the decomposition does not exceed a given value  $s$ . The dynamic programming algorithm considers every possible subinterval of  $T_D$ . The algorithm maintains a three dimensional table  $R$ . Let  $\Pi_S(T_D) = ST_1 * \dots * ST_n$ . Let  $T(x, y)$  denote the time interval obtained by combining into a single interval  $ST_x, ST_{x+1}, \dots, ST_y$  of  $\Pi_S(T_D)$ . The size of table  $R$  is  $n \times n \times s$ , where  $n$  is the the number of subintervals in  $\Pi_S(T_D)$  and  $s$  is the user specified constraint on the number of subintervals. (Typically,  $s$  would be much less than  $n$ .) Entry  $R[x, y, k]$  denotes the minimum amount of information loss possible in a  $k$  subinterval decomposition of the time interval  $T(x, y)$ . Note that  $k$  can be at most the size of  $T(x, y)$ , namely  $y - x + 1$ . There is an entry  $R[x, y, k]$  for each  $x, y, k$  such that  $1 \leq x \leq y \leq n$  and  $1 \leq k \leq \min(s, y - x + 1)$ . The value of each of these entries is computed as follows.

$$R[x, x, 1] = 0.$$

$$R[x, y, 1] = \mu(T(x, y)) \text{ if } x < y.$$

$$R[x, y, k] = \min_{x+k-2 \leq z < y} (R[x, z, k-1] + R[z+1, y, 1]) \\ \text{if } x < y \text{ and } 2 \leq k \leq \min(s, y - x + 1).$$

The minimum value among all entries of the form  $R[1, n, k]$  gives the amount of information loss for the optimal lossy decomposition. A separate table  $T$  with the same dimensions as  $R$  can be used to remember the  $z$  value corresponding to each  $R[x, y, k]$ . This table can be used in reconstructing the optimal lossy decomposition.

If it is desired to construct a decomposition with the smallest number of intervals, subject to a constraint on the information loss, a variant of the dynamic programming algorithm in Figure 5 can be used. For this optimization problem, there is no constraint on the number of intervals, so  $R[x, y, k]$  is computed for all  $k$  such that  $1 \leq k \leq y - x + 1$ . The minimum value of  $k$  such that the value of  $R[1, n, k]$  does not exceed the given constraint on information loss gives the number of intervals in the optimal decomposition.

In some cases, it may be important to compute a decomposition with subintervals having more or less the

- Let  $\Pi_S$  be the shortest interval decomposition for time period  $T_D$  associated with document set  $D$ . Let  $n$  be the number of subintervals in  $\Pi_S$ . Let  $s$  be the user-specified constraint on the number of intervals.

- for  $x = 1$  to  $n$   
 $R[x, x, 1] = 0;$
- for  $l = 2$  to  $n$   
for  $x = 1$  to  $n - l + 1$   
 $y = x + l - 1$   
 $R[x, y, 1] = \mu(T(x, y))$   
for  $k = 2$  to  $\min(s, y - x + 1)$   
 $R[x, y, k] = \min_{x+k-2 \leq z < y} (R[x, z, k-1] + R[z+1, y, 1])$

Figure 3: A dynamic programming algorithm for finding an optimal lossy decomposition of  $T_D$ .

same length. We define the **variability** of a decomposition  $\Pi$  as the ratio of the maximum length to the minimum length of its subintervals. The higher the variability value of a decomposition, the more non-uniform the length of its subintervals. A given decomposition can be characterized by three parameters: information loss, number of subintervals, and variability. One can consider the problem of constructing a decomposition optimizing any one of these parameters, subject to constraints on the other two. Any such problem can be solved in polynomial time via dynamic programming, building on the algorithm in Figure 5. For instance, consider the problem of minimizing information loss, subject to constraints on the number of subintervals and the variability. For each possible value,  $length_{max}$ , for the length of the longest interval in the decomposition of  $T_D$ , the given variability constraint imposes a minimum value  $length_{min}$  on the length of the shortest interval in the decomposition. In this case, the only intervals  $T(x, y)$  that need be considered are those whose length,  $y - x + 1$ , satisfies  $length_{min} \leq y - x + 1 \leq length_{max}$ . A value  $R[x, y, k]$  is computed for each  $x, y, k$  such that  $1 \leq x, x + length_{min} - 1 \leq y \leq \min(x + length_{max} - 1, n)$ , and  $1 \leq k \leq \min(s, y - x + 1)$ . The value of each of these entries is computed as follows.

$$R[x, y, 1] = \mu(T(x, y)) \text{ if } 1 \leq x \text{ and } x + length_{min} - 1 \leq y \leq \min(x + length_{max} - 1, n).$$

$$R[x, y, k] = \min_{x+k-2 \leq z < y} (R[x, z, k-1] + R[z+1, y, 1])$$

if  $1 \leq x$ ,  $x + length_{min} - 1 \leq y \leq \min(x + length_{max} - 1, n)$ , and  $2 \leq k \leq \min(s, y - x + 1)$ .

## 6 Experiments

In this section, we describe the results from some preliminary experiments we conducted using two separate document sets. The first document set contained all news articles from Reuters-21578, Distribution 1.0 (from hereon referred to as the Reuters data set or the Reuters set) collection published during March 1<sup>st</sup> - 4<sup>th</sup>, 1987. The second set included all the Medline abstracts (hereon referred to as the Medline data set or Medline set) related to Diabetes published in the first quarter (Jan 1<sup>st</sup> - Apr 30<sup>th</sup>) of 2002. There are 1279 news articles in the Reuters data set and 150 abstracts in the Medline data set. Each document in the Reuters set is time stamped with the granularity of a second. The abstracts in the Medline set have a time granularity of a day. The base granularity for the Reuters set was chosen to be an hour (not a second) to avoid having a sparse distribution of the data. For the Medline set, we retained the base granularity of a day. Based on these base granularities, the Reuters set had 70 subintervals and the Medline set had 65 subintervals in their respective shortest interval decompositions. Note that, any time points for which no documents assigned were removed.

The main goal of the experiment was to understand how the optimal lossless and optimal lossy decompositions of the time period of the given data set help in understanding the underlying temporal trends. Each data set was cleaned using a standard procedure that involved removing XML/SGML tags and stop words. Each article/abstract was then broken into a sequence of stemmed keywords. The shortest interval decomposition  $\Pi_S$  was constructed by partitioning the data set into subsets by assigning the articles/abstracts to exactly one of the base granularity subintervals using the time stamps. A ratio measure function was used to compute the information content from each subset of the data, where the significance of a keyword in the subset was computed as the ratio of the documents with the keyword to the total number of documents in that subset. The threshold  $\alpha$  was chosen to be 0.25. The information content for each subinterval in the shortest interval decomposition was computed. The Reuters set had 2066 keywords that were significant in at least one subinterval of the  $\Pi_S$ . The Medline set had 1508 such keywords. Let  $W_r$  ( $W_m$ ) denote the set of keywords from the Reuters (Medline) set.

We first computed the compressed interval decomposition  $\Pi_C^p$  for each keyword  $w_p$ . The decomposition  $\Pi_C^p$  for the individual keywords achieves significant com-

pression. The maximum size of the compressed interval decomposition of any keyword from both data sets is about half the size of its  $\Pi_S$  and the average size was about a quarter of the size of the  $\Pi_S$  of the corresponding data set.

Then the compressed interval decomposition of the time period  $T_D$  of the data set was computed as the product of the  $\Pi_C^p$  for each  $w_p$ . Since the measure function used in the experiments is stable, the optimal decomposition of  $T_D$  is the same as the  $\Pi_C$ . The optimal information preserving decomposition for the Reuters data set had 62 subintervals and hence, had around a 10% reduction in the number of subintervals over the size of  $\Pi_S$ . However, for the Medline data set, the optimal information preserving decomposition had a very small reduction (about 2%) in the number of subintervals as compared to its  $\Pi_S$ .

For each of the data sets, various optimal lossy decompositions were then computed for different thresholds on the size of the decomposition by using the dynamic programming algorithm in Figure 3 of Section 5. Given a constraint on the number of subintervals in a lossy decomposition, an optimal lossy decomposition

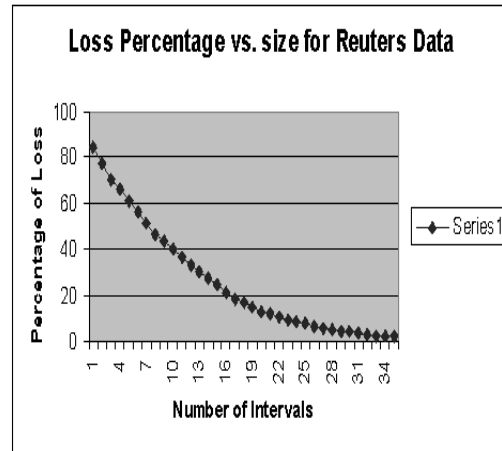


Figure 4: Percentage of information loss versus size of optimal lossy decomposition for the Reuters data.

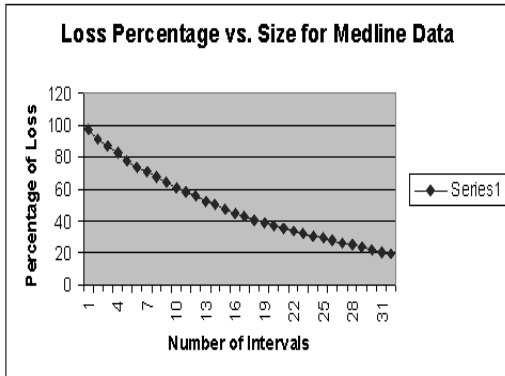


Figure 5: Percentage of information loss versus size of optimal lossy decomposition for the Medline data.

was constructed with minimal amount of information loss. A set of optimal lossy decompositions were constructed by varying the size of the optimal lossy decomposition. The results are plotted for both data sets in Figures 4 and 5. The  $x$ -axis is the size of the optimal lossy decomposition and the  $y$ -axis is the amount of information loss for each decomposition size as a percentage of the total information content of all subintervals in the  $\Pi_S$  corresponding to each data set. Both plots clearly illustrate that the information loss decreases as the number of subintervals allowed in the optimal lossy decomposition increases.

## 7 Related Work

Segmenting a document set based on the time stamps for identifying trends and tracking interesting topics is an active area of research. The work on topic detection and tracking in [4, 5] extracts significant topics/events from news articles by grouping the articles published on the same day together. Papers on extracting trends from time stamped text documents also use time decompositions, where subintervals are of length a year [1, 2] or a day [6, 7, 8]. These references focus more on iden-

tifying trends from the document set than on the issue of whether the decomposition chosen is appropriate for capturing all the trends.

Reference [9] uses time decompositions to discretize time series data for schema discovery and does not focus on text documents. Reference [14] discusses various methods for discretizing continuous features for machine learning. In [3], the authors describe how to identify bursts from document streams such as news articles by modeling the streams as infinite automaton, whereas our work is more applicable to finite document sets. Reference [10] uses dynamic programming to identify trends from time series data, but does not focus on text documents.

## 8 Conclusions

In this paper, we formalized the problem of identifying an optimal decomposition of a time period based on a given set of documents published in that time period, and designed efficient algorithms for computing it. We first formalized the notions of measure and textmine function to capture the behavior of commonly used text mining functions for computing significant events from a given document set. Then, we defined the information content of a time interval and studied the effect of different measure functions on the information content computed. We then defined information preserving, compressed interval and optimal information preserving decompositions for the given time period. We showed how these concepts relate to one another, based on whether the measure function used for computing the information content for each time interval is monotone increasing, stable, or neither. We then designed efficient algorithms for computing the optimal information preserving decomposition for the given time period. Since optimal information preserving decompositions may still have a large number of subintervals, we formalized the concept of an optimal *lossy* decomposition. We then developed an efficient dynamic programming based algorithm for constructing an optimal lossy decomposition. This decomposition has minimal information loss, subject to a constraint on the number of subintervals in the decomposition. We also discussed how to solve the problem of minimizing the number of subintervals, subject to a constraint on information loss, as well as problems involving information loss, the number of subintervals, and variability of subinterval length. We demonstrated the effectiveness of the approaches on two document sets – Reuters-21578 Distribution 1.0 collection and Medline abstracts.

We would like to extend our work to other text mining functions such as correlations. We would also like to study decompositions as data objects for querying and

comparison purposes.

## References

- [1] B. Lent, R. Agrawal, and R. Srikant, “Discovering Trends in Text Databases”, *Proc. 3rd Intl. Conf. Knowledge Discovery and Data Mining (KDD)*, 1997, pp 227-230.
- [2] S. Roy, D. Gevry, W. M. Pottenger, “Methodologies for Trend Detection in Textual Data Mining”, *Proc. Textmine '02 Workshop*, SIAM Intl. Conf. on Data Mining, 2002.
- [3] J. Kleinberg, “Bursty and Hierarchical Structure in Streams”, *Proc. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2002, pp 91-101.
- [4] J. Allan, V. Lavrenko, D. Malin, and R. Swan, “Detections, Bounds, and Timelines: UMass and TDT-3”, *Proc. 3rd Topic Detection and Tracking Workshop*, 2000.
- [5] J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang, “Topic Detection and Tracking Pilot Study: Final Report”, *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998.
- [6] R. Swan and J. Allan, “Automatic Generation of Overview Timelines”, *Proc. 23rd Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2000, pp.49-56.
- [7] R. Swan and J. Allan, “Extracting Significant Time Varying Features from Text”, *Proc. 8th Intl. Conf. on Information and Knowledge Management*, 1999, pp. 38-45.
- [8] R. Swan and D. Jensen, “TimeMines: Constructing Timelines with Statistical Models of Word Usage”, *Proc. KDD 2000 Workshop on Text Mining*, 2000.
- [9] M. Motoyoshi, T. Miura, and K. Watanabe, “Mining Temporal Classes from Time Series Data”, *Proc. 11th Intl. Conf. on Information and Knowledge Management*, 2002, pp. 493-498.
- [10] D. J. Berndt and J. Clifford, “Finding Patterns in Time Series: A Dynamic Programming Approach”, *Advances in Knowledge Discovery and Data Mining*, Editor U. M. Fayyad et al. AAAI Press, 1996.
- [11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “Introduction to Algorithms, 2nd Edition”, MIT Press, 2001.
- [12] K. Rosen, “Discrete Mathematics and Its Applications”, McGraw Hill College Division, 5th edition, 2003.
- [13] R. Agrawal, G. Psaila, E. Wimmers, and M. Zait, “Querying Shapes of Histories”, *Proc. of 21st International Conference on Very Large Databases*, 1995.
- [14] J. Dougherty, R. Kohavi, and M. Sahami, “Supervised and unsupervised discretization of continuous features”, *Proc. of the 12th International Conference on Machine Learning*, 1995, pp. 194–202.