Using Incremental PLSI for Threshold-Resilient Online Event Analysis

Tzu-Chuan Chou and Meng Chang Chen

Abstract—The goal of online event analysis is to detect events and track their associated documents in real time from a continuous stream of documents generated by multiple information sources. Unlike traditional text categorization methods, event analysis approaches consider the temporal relations among documents. However, such methods suffer from the threshold-dependency problem, so they only perform well for a narrow range of thresholds. In addition, if the contents of a document stream change, the optimal threshold (that is, the threshold that yields the best performance) often changes as well. In this paper, we propose a threshold-resilient online algorithm, called the Incremental Probabilistic Latent Semantic Indexing (IPLSI) algorithm, which alleviates the threshold-dependency problem and simultaneously maintains the continuity of the latent semantics to better capture the story line development of events. The IPLSI algorithm is theoretically sound and empirically efficient and effective for event analysis. The results of the performance evaluation performed on the Topic Detection and Tracking (TDT)-4 corpus show that the algorithm reduces the cost of event analysis by as much as 15 percent $\sim$ 20 percent and increases the acceptable threshold range by 200 percent to 300 percent over the baseline.

Index Terms—Clustering, online event analysis, probabilistic algorithms, text mining.

1 Introduction

Publishing activities are now ubiquitous because of the rapid growth of the Internet. When an event occurs, numerous independent authors (called *information sources*) publish articles about the event during its life span. Such articles form a *document stream*, which is a collection of chronologically ordered documents reporting concurrent events. The goal of the DARPA-sponsored Topic Detection and Tracking (TDT) Project [1] is to promote *TDT* research, where a topic is defined as something associated with a specific action that occurs at some place and time. *Event analysis* is similar to TDT in that it automatically identifies events and associated documents in a document stream. In many ways, an event is the same as a topic in TDT, except that from our perspective, an event has a story line, and its focus may change during its life span. In addition, an event may last for a long time (for example, an international conflict) or for a short period (for example, a car accident). In terms of processing outcomes, event analysis is similar to traditional document clustering. Moreover, both techniques partition a set of documents into several coherent parts, each of which relates to a single event.

There are two types of event analysis: *online* and *retrospective*. In online event analysis, documents are generated continuously and ordered chronologically so that the analysis process needs to make decisions in real time with all or some of the published documents. When a new document arrives, the online analysis process either assigns it to a known event or creates a new event for it. In the latter case, it becomes the first document of the new event. In addition to accurate analysis, prompt responses and reasonable computational overheads are the major issues in online event analysis. In contrast, in *retrospective* event analysis, all the documents are known in advance, and the processing time may not be a concern, because the processing is usually performed offline.

In online event analysis, the incremental clustering algorithm [1] is employed to cluster documents on by one in chronological order. A document is deemed a member of a certain cluster if the similarity between its text and that of the other documents in the cluster is above a predefined threshold. If no cluster is similar enough to the document, a new cluster is created, and the document is treated as the first document of the newly created topic. The performance of the incremental clustering algorithm on the TDT task is excellent, as long as the contents of the topic maintain a high degree of similarity during the topic's life span. However, the textual content of a long-term event or a multitopic event may change over time to reflect theme changes or different aspects of the event. Allan et al. [2] noted that temporally approximate documents probably relate to the same event. By constantly raising the similarity threshold of the event detection method for each time increment, it is possible to prevent temporally remote documents being merged into the same cluster. Therefore, documents with similar contents that relate to different events can be correctly distinguished.

Traditional document classification methods employ the conventional vector space model (VSM) to represent documents as vectors. Each vector is a set of terms whose weights are usually determined by the TF-IDF scheme [17], whereas the similarity between two vectors is measured by the cosine similarity. Yang et al. [19] applied the VSM to the

• *The authors are with the Institute of Information Science, Academia Sinica Taiwan, 128 Sec. 2, Academia Rd., Nankang, Taipei 115 Taiwan. E-mail: {tzuchuan, mcc}@iis.sinica.edu.tw.*

Manuscript received 4 Jan. 2007; revised 10 July 2007; accepted 4 Oct. 2007; published online 12 Oct. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0006-0107.
Digital Object Identifier no. 10.1109/TKDE.2007.190702.

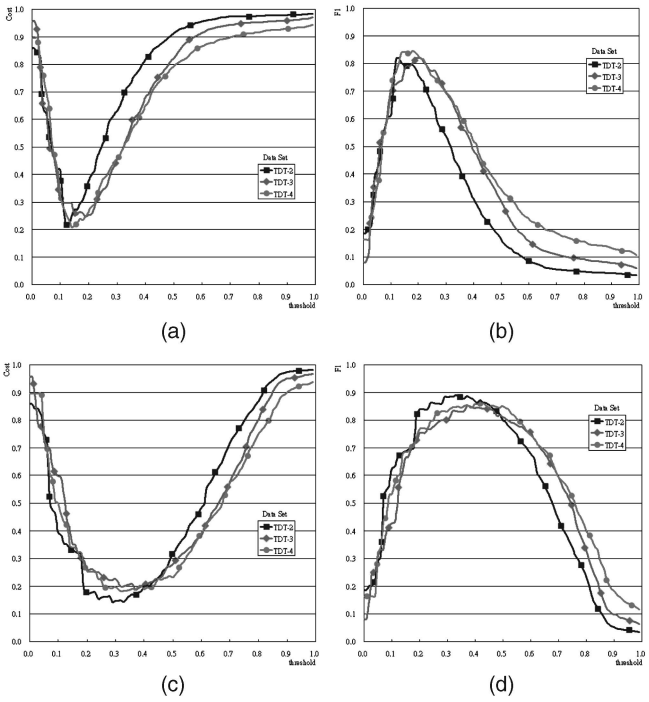

Fig. 1. The detection error trade-off costs of TW-DF and PLSI for TDT-2, TDT-3, and TDT-4. (a) Cost of the TW-DF method. (b) F1 score of the TW-DF method. (c) Cost of the PLSI algorithm. (d) F1 score of the PLSI algorithm.

task of news TDT and used a time window with a decaying function (*TW-DF*) to model the temporal relations between documents and events. In this method, the size of the time window dictates the number of previous documents to be considered for clustering, and the decaying function weights the influence of a document in the window according to the gap between it and the newest document. Like the time-based threshold approach, remote documents within the time window have less impact on clustering than documents that are chronologically close. In addition, analogous events that occur in different time periods are less likely to be clustered together. This method is one of the best online news detection and tracking algorithms available [20], [21].

The TF-IDF approach achieves a reasonably good classification and clustering performance if a proper threshold is selected. However, the performance degrades rapidly when the selected threshold differs from the optimum threshold (that is, the threshold that achieves the best performance) by even a small amount. We apply the TW-DF method to the official TDT corpora TDT-2, TDT-3, and TDT-4 [22]. The computational costs and F1 scores are shown in Figs. 1a and 1b, respectively. In the figures, we observe that the computational cost and F1 score degrade sharply from their respective optimal thresholds. We call this phenomenon the *threshold-dependency* problem. Interestingly, although this is an important issue, it is frequently neglected.

The Probabilistic Latent Semantic Indexing (PLSI) algorithm proposed by Hofmann [10], [11] uses a probabilistic model and the expectation-maximization (EM) method for text classification and other applications. We found that the PLSI algorithm can alleviate the *threshold-dependency* problem. Figs. 1c and 1d show that the computational cost and F1 score degrade gently from the optimal threshold, which suggests that a near-optimal threshold can achieve a reasonable performance.

The PLSI algorithm uses the training data to build a probabilistic model, which estimates the parameters of new documents in the test phase. This process is called the "fold-in" approach [10], [11]. Because of this characteristic, the PLSI algorithm is only suitable for offline applications and not for applications with a continuous incoming data stream such as online news event analysis. To apply the PLSI algorithm in online event analysis, the algorithm has to be rerun, and the latent semantic indices have to be reestimated for every time period. Thus, to analyze events that cover different time periods, one can establish the connection between the latent semantic indices of adjacent time periods by calculating their similarities. For instance, to construct event trails, Mei and Zhai utilized the Kullback-Leibler (KL) divergence to calculate the similarities between the latent semantic indices of different time periods [15]. However, under this approach, the latent semantics between time periods may be incompatible, because the PLSI algorithm may converge to different local optima. Thus, since this approach cannot maintain latent continuity, the smooth presentation of the resulting event trails may be disrupted.

In this paper, we propose a threshold-resilient online algorithm, called the Incremental Probabilistic Latent Semantic Index (IPLSI) algorithm. In contrast to PLSI, IPLSI processes incoming documents incrementally for each time period (or after collecting a certain number of documents), discards out-of-date documents and terms not used in recent documents already processed, and "folds in" new terms and documents for that time period. Therefore, the latent semantic indices are likely preserved from one time period to the next. (We call this property maintaining *latent continuity*.) Consequently, IPLSI can track the development of events better than PLSI. In addition, IPLSI alleviates the threshold-dependency problem and thereby extends the acceptable threshold range. The evaluation results of the IPLSI algorithm on the TDT-4 corpus show that it reduces the trade-off cost of errors in event detection by as much as 15 percent $\sim$ 20 percent and increases the acceptable threshold range by 200 percent $\sim$ 300 percent over the baseline.

The remainder of this paper is structured as follows: In Section 2, we introduce some related work. In Section 3, we describe the original PLSI algorithm, the naive incremental approach, and our proposed IPLSI algorithm. Section 4 considers our test corpora, the performance measures, and the baseline method and discusses the threshold-dependency problem. Section 5 details the experiment results. In Section 6, we discuss the issue of latent semantic continuity. Then, in Section 7, we summarize our work and present our conclusions.

2 Related Work

There are several noteworthy related work. Li et al. [13] adopted a mixture of unigram models to handle text information, a Gaussian Mixture Model to handle time information, and a generative model to combine text and time information for clustering and summarizing news topics. Although the method avoids the inflexible use of

time stamps employed in traditional event detection algorithms, it is designed for retrospective, not online, news event detection. For the online approach, Morinaga and Yamanishi [16] use a Gaussian Mixture Model to deal with text information and a time-stamp-based discounting learning algorithm that tracks topic structures adaptively by deleting out-of-date statistical data. For computational simplicity, the Gaussian Mixture Model employed in [16] assumes that the covariance matrices of all Gaussian distributions are diagonal.

For online event analysis, Surendran and Sra [18] proposed incrementally Built Aspect Models (BAMs) to dynamically discover new themes from document streams. BAMs are probabilistic models designed to accommodate new topics with the spectral algorithm and use a "fold-in" approach similar to that of the original PLSI. This approach retains all the conditional probabilities of the old words, given the old latent variables, and the spectral step is used to estimate the probabilities of new words and new documents. The new model becomes the starting point for discovering subsequent new themes so that the latent variables in the BAM model can be grown incrementally. Under this approach, the probabilities of old latent variables are retained, and the probabilities of new latent variables are detected as needed while the streaming data is being processed. This is an excellent means of applying incremental algorithms to online text analysis. Although this is a new theme (or latent) detection mechanism, it is not an online text clustering approach; therefore, its purpose differs from that of online event analysis.

Chakrabarti et al. proposed a framework of evolutionary clustering [6]. They argued that evolutionary clustering should simultaneously optimize the clustering accuracy of snapshots and the clustering consistency along a timeline. In their work, a user-defined *change parameter* is required to arrange trade-offs between the two objectives. The authors proposed several greedy approaches to modify traditional clustering methods, K-Means methods, and agglomerative hierarchical clustering algorithms to fulfill their requirements. These greedy approaches provide users with a smooth view of cluster changes when the input data drifts from the current clusters. However, in the TDT new event detection and tracking task, it is a requirement that every news document should be grouped with documents related to the same real-world event. Evolutionary clustering tends to maintain the consistency of clustering by sacrificing the clustering accuracy; hence, it is not suitable for event analysis tasks.

3 The Probabilistic Latent Semantic Indexing Algorithm and the Proposed Algorithm

3.1 Probabilistic Latent Semantic Indexing Algorithm

The PLSI model incorporates higher level latent concepts/semantics to smooth the weights of terms in documents [10], [11]. The latent semantic variables can be viewed as intermediate concepts or topics placed between documents and terms. Meanwhile, the associations between documents, concepts, and terms are represented as conditional probabilities and are estimated by the EM algorithm, an iterative technique that converges to a maximum likelihood estimator under incomplete data [9]. After the PLSI parameters have been estimated, the similarities between new documents (called *query documents* in [10], [11]) and existing documents can be calculated by using the smoothed term vectors. The PLSI algorithm, which can be used in text classification and information retrieval applications [4], [12], achieves better results than traditional VSM methods [10], [11]. We discovered another advantage of the PLSI model in that it can expand the acceptable threshold range. We discuss this aspect later in the paper.

The following notations are used in the PLSI algorithm: d denotes a document, w denotes a term in a document, z denotes a latent variable, D denotes the set of documents, W denotes the set of terms, Z denotes the set of latent variables, and q denotes a new (query) document. We also use these notations in the proposed IPLSI algorithm.

In the PLSI algorithm, a latent variable z is introduced between documents and terms so that their association can be represented as conditional probabilities $P(w|z)$ and $P(z|d)$ [4], [11]. The probabilities can also be represented by $P(z)$, $P(d|z)$, and $P(w|z)$, as reported in [3], [10]. The PLSI algorithm assumes that $P(w,d)$, that is, the distribution of a term w and a document d, is conditionally independent, given z; that is, $P(w,d|z) = P(w|z)P(d|z)$, $P(w|z,d) = P(w|z)$, and $P(d|z,w) = P(d|z)$. Using these definitions and assumptions, we can define a generative model for term/document co-occurrences as follows:

- select a document d with probability $P(d)$,
- pick a latent class z with probability $P(z|d)$, and
- generate a word w with probability $P(w|z)$.

This yields an observation pair (d,w), whereas the latent class variable z is discarded. The translation of the data generation process into a joint probability model is shown as follows:

$$\begin{aligned} P(w,d) &= \sum_{z\in Z} P(z)P(w|z)P(d|z) \\ &= P(d)\sum_{z\in Z} P(w|z)P(z|d) \\ &= P(w)\sum_{z\in Z} P(d|z)P(z|w). \end{aligned} \tag{1}$$

The parameters of the PLSI model are estimated by the iterative EM algorithm, which uses a training document set D to maximize the log-likelihood function L:

$$L = \sum_{d\in D}\sum_{w\in d} f(w,d)\log P(w,d), \tag{2}$$

where $f(w,d)$ is the frequency of a word w in a document d. The PLSI parameters $P(w|z)$ and $P(z|d)$ are initialized randomly and normalized, after which they are revised by applying the EM procedure iteratively until they converge. According to the Bayes rule, the conditional probability of $P(z|w,d)$ can be estimated by the following in the E (*Estimation*) step:

$$P(z|w,d) = \frac{P(w|z)P(z|d)}{\sum_{z'\in Z} P(w|z')P(z'|d)}. \tag{3}$$

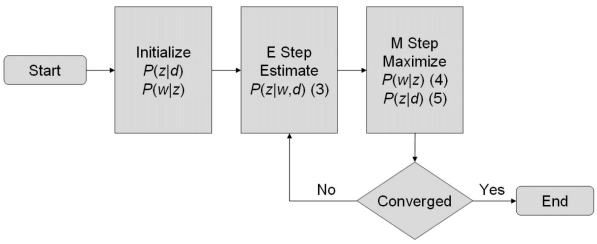

Fig. 2. The PLSI training process.

In the M (*Maximization*) step, the probabilities $P(w|z)$ and $P(z|d)$ can be estimated, respectively, by the following:

$$P(w|z) = \frac{\sum_{d \in D} f(w,d) P(z|w,d)}{\sum_{d \in D} \sum_{w' \in d} f(w',d) P(z|w',d)}, \quad (4)$$

$$P(z|d) = \frac{\sum_{w \in d} f(w,d) P(z|w,d)}{\sum_{w \in d} f(w,d)}. \quad (5)$$

The equations used in the M step are derived by the Lagrange Multiplier Method. Note that the parameters $P(z|w,d)$, $P(w|z)$, and $P(z|d)$ are iteratively refined until they converge. The above process is called the *training process* of the PLSI model (see Fig. 2).

After the completion of the above training process, the estimated $P(w|z)$ parameters are used to estimate new parameters $P(z|q)$ and $P(z|w,q)$ for a new document q. This is called the *folding-in process* [4], [10]. In this process, the probability $P(z|q)$ is first initialized randomly and normalized and, then, it is revised using (6) and (7) for the E and M *steps*, respectively. Note that in the EM procedure, all $P(w|z)$ remain fixed. Consequently, the folding-in process can usually be accomplished in just a few iterations:

$$P(z|w,q) = \frac{P(w|z) P(z|q)}{\sum_{z' \in Z} P(w|z') P(z'|q)}, \quad (6)$$

$$P(z|q) = \frac{\sum_{w \in q} f(w,q) P(z|w,q)}{\sum_{w \in q} f(w,q)}. \quad (7)$$

When the PLSI algorithm is used in text classification applications, a document d is represented by a smoothed version of the term vector $(P(w_1|d), P(w_2|d), P(w_3|d), \ldots)$, where

$$P(w|d) = \sum_{z \in Z} P(w|z) P(z|d). \quad (8)$$

The new document q is represented by

$$(P(w_1|q), P(w_2|q), P(w_3|q), \ldots),$$

where

$$P(w|q) = \sum_{z \in Z} P(w|z) P(z|q). \quad (9)$$

Then, after weighting by the IDF, the similarity between any two documents can be calculated by the following cosine function:

$$sim(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1 \bullet \vec{d}_2}{|\vec{d}_1| \times |\vec{d}_2|}, \quad (10)$$

where

$$\vec{d} = (P(w_1|d) \times IDF(w_1), P(w_2|d) \times IDF(w_2), \ldots), \quad (11)$$

$$IDF(w) = \log\left(\frac{N}{DF(w)}\right). \quad (12)$$

In (12), N is the total number of documents (that is, the existing documents d plus new documents q), and $DF(w)$ is the number of documents that contain the term w. In the original work on PLSI [10], [11], it was shown by experiment that the algorithm improves text classification performance.

The model of the PLSI algorithm, which is learned from training data, is constant during the whole processing period; that is, data that arrives after the training phase does not affect the original model. When a new document q is input to the system, the parameters of the PLSI algorithm, that is, $P(z|w,d)$, $P(w|z)$, and $P(z|d)$, remain fixed; thus, only the parameters related to the new document q, that is, $P(z|q)$ and $P(z|w,q)$, are estimated in the folding-in process. For example, all the $P(w|z)$ remain fixed during the folding-in process, even if the term w occurs in the new documents. In addition, new terms w_{new}, which only occur in new documents, are ignored in the folding-in process. In an online application, new documents arrive continuously, so the parameters of the PLSI algorithm must be updated regularly to accommodate changes, because the concepts of the latent variables may change. Since the original PLSI algorithm cannot handle this process effectively, we designed the IPLSI algorithm.

3.2 A Naive Incremental Probabilistic Latent Semantic Indexing Approach

In online event analysis, the system contains an initial set of documents, and new documents arrive continuously. The system compares an incoming document with existing documents to decide which event the new document belongs to, or it generates a new event if the document does not relate to an existing event. Because of the "aging" nature of events [7], an old inactive event less likely attracts new documents than a recently active event. Therefore, an event analysis system has to consider the temporal relations of documents. Incorporating a lookup window is a popular way of limiting the time frame that an incoming document can relate to. An example of a window-based document scope system is shown in Fig. 3. With each advance of the window, which can be measured in time units or by a certain number of documents, the system discards old documents and folds in new ones.

A naive way of performing event analysis in a window-based system using the PLSI technique is to run the PLSI algorithm for each advance of the window. This means that for every advance of the window, the EM algorithm uses a

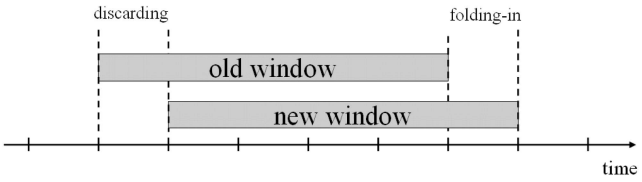

Fig. 3. The discarding and folding-in processes.

new random initial setting to reestimate all the parameters of the PLSI algorithm. Although the approach is straightforward, some problems may arise. For example, the EM algorithm can only guarantee obtaining a local optimum, so different initial settings of the PLSI parameters may result in different local optima. Even if we choose the same initial setting every time, the EM algorithm may still derive different local optima if the processed documents sets are different. Thus, the latent semantic variables may not be continuous for each window advance. In Section 6, we present some evaluation results that exemplify this discontinuous scenario. Another problem is that the long execution time of the naive approach for each window advance makes it unsuitable for online processing. We also address this point in Section 5.

3.3 The Proposed Incremental Probabilistic Latent Semantic Indexing Algorithm

In event analysis, an incoming document q is classified into an existing category or is assigned as the first story of a new category. In the original PLSI algorithm, only the probabilities $P(z|q)$ and $P(z|w,q)$ are estimated when a new document q is added to the system. In other words, the other system parameters are not adjusted, and new terms in the new documents are completely ignored. However, in the case of online analysis, the story line of an event may evolve, so a new document may actually indicate a turning point in the story line of the event. Hence, a new document d_{new} must be folded into the system with all other PLSI parameters, that is, $P(z|w_{old},d_{old})$, $P(z|w_{old},d_{new})$, $P(z|w_{new},d_{new})$, $P(w_{old}|z)$, $P(w_{new}|z)$, $P(z|d_{old})$, and $P(z|d_{new})$, where d_{old} and w_{old} are old documents and old terms, respectively. Since the original PLSI algorithm is not suitable for online news analysis, we propose the *IPLSI* algorithm to resolve the problems in online event analysis.

In the IPLSI algorithm, the PLSI algorithm is executed once on the initial documents. Then, for each window advance, the IPLSI algorithm performs four steps to update the model (note that steps 2, 3, and 4 employ the EM algorithm):

1. **Discard old documents and terms.** As the time window advances, the IPLSI algorithm removes out-of-date documents d_{out} and old terms w_{out} not used in recent documents. During this process, the PLSI parameters $P(w_{out}|z)$, $P(d_{out}|z)$, $P(z|w_{out})$, and $P(z|d_{out})$ are also removed. In order to observe the basic principle of probability that the total probability will be equal to 1, the remaining $P(w|z)$ and $P(d|z)$ must be renormalized proportionally as follows (note that $P_0(w|z)$ and $P_0(d|z)$ are the probabilities of the remaining terms and documents, respectively, whereas W_0 and D_0 are the respective sets of the remaining terms and documents):

$$P(w|z)=\frac{P_0(w|z)}{\sum_{w'\in W_0}P_0(w'|z)},\ P(d|z)=\frac{P_0(d|z)}{\sum_{d'\in D_0}P_0(d'|z)}. \quad (13)$$

2. **Fold in new documents.** In this step, the new documents d_{new} are folded in, and all $P(z|d_{new})$ are initialized randomly. $P(w|z)$ are fixed during this step and are used to estimate $P(z|d_{new})$. The EM algorithm is used to estimate $P(z|w,d_{new})$ and $P(z|d_{new})$, where the following are the E and M *steps*, respectively (note that the folding-in method, which is the same as that used in [4]and [10], simply replaces a query document q with a new document d_{new}):

$$P(z|w,d_{new})=\frac{P(w|z)P(z|d_{new})}{\sum_{z'\in Z}P(w|z')P(z'|d_{new})}, \quad (14)$$

$$P(z|d_{new})=\frac{\sum_{w\in d_{new}}f(w,d_{new})P(z|w,d_{new})}{\sum_{z'\in Z}\sum_{w\in d_{new}}f(w,d_{new})P(z'|w,d_{new})}. \quad (15)$$

3. **Fold in new terms.** New terms w_{new} found in the new documents are folded in. To estimate $P(z|w_{new})$, the $P(d_{new}|z)$ must be calculated as follows, since $P(z|d_{new})$ have been estimated in the previous step (note that D_{new} is a set of new documents):

$$P(z|w,d_{new})=\frac{P(w|z)P(z|d_{new})}{\sum_{z'\in Z}P(w|z')P(z'|d_{new})}, \quad (16)$$

$$P(d_{new}|z)=\frac{\sum_{w\in d_{new}}f(w,d_{new})P(z|w,d_{new})}{\sum_{d\in D_{new}}\sum_{w\in d}f(w,d)P(z|w,d)}. \quad (17)$$

After all $P(d_{new}|z)$ have been calculated, $P(z|w_{new})$ are initialized randomly and are normalized. Meanwhile, the $P(d_{new}|z)$ parameters are fixed and used to estimate $P(z|w_{new})$ for new terms. The EM algorithm is applied in this folding-in process, and the probabilities $P(z|w_{new},d_{new})$ and $P(z|w_{new})$ are estimated as follows in the E and M *steps*, respectively:

$$P(z|w_{new},d_{new})=\frac{P(d_{new}|z)P(z|w_{new})}{\sum_{z'\in Z}P(d_{new}|z')P(z'|w_{new})}, \quad (18)$$

$$P(z|w_{new})=\frac{\sum_{d\in D_{new}}f(w_{new},d)P(z|w_{new},d)}{\sum_{d'\in D_{new}}f(w_{new},d')}. \quad (19)$$

4. **Update the PLSI parameters.** Before revising the PLSI parameters, we need to calculate $P(w_{new}|z)$ and adjust $P(w_{old}|z)$, because the values of $P(w_{new}|z)$ did

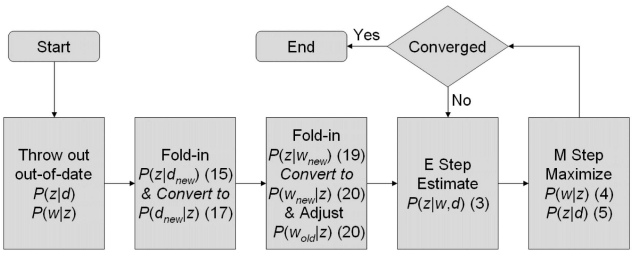

Fig. 4. The flowchart of the IPLSI algorithm.

not exist in the previous window, and w_{out} may occur in the new documents d_{new}. For the new terms w_{new}, we use (18) to calculate $P(z|w,d)$, and for the old terms w_{old}, we use (4) to calculate $P(z|w,d)$. Next, to observe the basic principle that the total probability will be equal to 1, all $P(w|z)$ are normalized using the following:

$$P(w|z) = \frac{\sum_{d \in D \cup D_{new}} f(w,d) P(z|w,d)}{\sum_{d' \in D \cup D_{new}} \sum_{w' \in d'} f(w',d') P(z|w',d')}. \quad (20)$$

Then, the EM algorithm used by the PLSI algorithm, as described in (3), (4), and (5), is executed to revise all the PLSI parameters, because new terms w_{new} and new documents d_{new} have been introduced.

The IPLSI process for one window advance is summarized in the flowchart shown in Fig. 4. As the window advances, the four-step design of the IPLSI algorithm preserves the probability and continuity of the latent parameters during each revision of the model.

Our proposed approach folds in new terms and new documents in separate steps. In contrast, a naive approach folds in such terms and documents simultaneously. The problem with this simple approach is that after each advance of the window, the sum of all probabilities of terms in W_0 under z is 1, that is,

$$\sum_{w \in W_0} P(w|z) = 1. \quad (21)$$

After new documents arrive, all $P(w_{new}|z)$ are initialized randomly, which means that the sum of the probabilities of terms under z will be larger than 1. Hence, the probabilities must be normalized such that

$$\sum_{w \in W} P(w|z) = 1, \quad (22)$$

where $W = W_0 + W_{new}$, and W_{new} is the set of new terms. However, as $P(w_{new}|z)$ are random values between 0 and 1, it is meaningless to normalize the probabilities $P(w|z)$ for all w in W.

To avoid the above problem, we fold in new documents in the second step. In the third step, we derive $P(d_{new}|z)$ from $P(z|d_{new})$ by (16) and (17), and we fix $P(d_{new}|z)$ to estimate $P(z|w_{new})$ by (18) and (19). This way, we can systematically ensure that the total probability is equal to 1, and we add the parameters of the new terms w_{new} and new documents d_{new} smoothly at different times, unlike in simple methods.

TABLE 1
The Statistical Data of the Evaluation Corpora

	TDT-2	TDT-3	TDT-4
Time interval (months)	6	3	4
Source File #	2,531	1,387	1,426
Event #	96	115	70
Tagged Document #	7,959	5,593	1,845
Min Stemmed Word #	4	5	10
Max Stemmed Word #	798	918	1042
Avg Stemmed Word #	109	122	158

The advantages of the IPLSI algorithm are twofold. First, the convergence time of the EM algorithm is reduced substantially, because most parameters remain the same, or they are only modified slightly. The second advantage is that the continuity of the latent semantic variables is maintained. Because of these advantages, the proposed IPLSI algorithm is more effective and efficient than the naive IPLSI approach, which reestimates all the parameters by random initialization of the PLSI algorithm for each window advance.

4 Corpora and Settings for Evaluation

4.1 Corpora

In the evaluation, we used the standard corpora TDT-2, TDT-3, and TDT-4 from the NIST TDT corpora [22]. Only English documents tagged as definitely news topics (that is, tagged YES) were chosen for evaluation. The statistical data of the corpora is shown in Table 1.

4.2 Performance Metrics

We follow the performance measurements defined in [19]. An event analysis system may generate any number of clusters, but only the clusters that best match the labeled topics are used for evaluation.

Table 2 illustrates a 2×2 contingency table for a cluster-topic pair, where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ represent the numbers of documents in the four cases. Four singleton evaluation measures, ***Recall***, ***Precision***, ***Miss***, and ***False Alarm***, and two primary evaluation measures, $F1$ and normalized **Cost** (also called the *Normalized Detection Error Tradeoff Cost* [5], [14]), are defined as follows:

- ***Recall*** $= \mathbf{a}/(\mathbf{a}+\mathbf{c})$ if $(\mathbf{a}+\mathbf{c}) > 0$; otherwise, it is undefined.
- ***Precision*** $= \mathbf{a}/(\mathbf{a}+\mathbf{b})$ if $(\mathbf{a}+\mathbf{b}) > 0$; otherwise, it is undefined.
- ***Miss*** $= \mathbf{c}/(\mathbf{a}+\mathbf{c})$ if $(\mathbf{a}+\mathbf{c}) > 0$; otherwise, it is undefined.
- ***False Alarm*** $= \mathbf{b}/(\mathbf{b}+\mathbf{d})$ if $(\mathbf{b}+\mathbf{d}) > 0$; otherwise, it is undefined.

TABLE 2
A Cluster-Topic Contingency Table

	In topic	Not in topic
In cluster	**a**	**b**
Not in cluster	**c**	**d**

- $\boldsymbol{F1} = 2 * \boldsymbol{Recall} * \boldsymbol{Precision}/(\boldsymbol{Recall} + \boldsymbol{Precision})$.
- $\boldsymbol{Cost}_{Det} = \mathrm{C}_{\mathrm{miss}} * \mathrm{P}_{\mathrm{target}} * \boldsymbol{Miss} + \mathrm{C}_{\mathrm{FA}} * (1 - \mathrm{P}_{\mathrm{target}}) * \boldsymbol{False\ Alarm}$.
- $(\boldsymbol{Cost}_{Det})_{Norm} = \boldsymbol{Cost}_{Det}/\min(\mathrm{C}_{\mathrm{miss}} * \mathrm{P}_{\mathrm{target}}, \mathrm{C}_{\mathrm{FA}} * (1 - \mathrm{P}_{\mathrm{target}}))$.

In the definition of ***Cost***, $\mathrm{C}_{\mathrm{miss}}$, and C_{FA} are the costs of missed detection and false alarms, respectively, and $\mathrm{P}_{\mathrm{target}}$ is the probability of finding a relevant story. According to the standard TDT cost function used for all evaluations in TDT, $\mathrm{C}_{\mathrm{miss}} = 1$, $\mathrm{C}_{\mathrm{FA}} = 0.1$, and $\mathrm{P}_{\mathrm{target}} = 0.02$ [14]. In the following, we use ***Cost*** to denote the ***Normalized Detection Error Tradeoff Cost*** $(\boldsymbol{Cost}_{Det})_{Norm}$.

In our evaluations, we apply the microaverage method to the global performance measurement. The microaverage is obtained by merging the contingency tables of the topics (by summing the corresponding cells) and then using the merged table to derive global performance measurements.

4.3 Evaluation Baseline and the Threshold-Dependency Problem

We use the TW-DF method [19] as the baseline, since it is one of the most efficient online news event detection and tracking algorithms available [20], [21]. For comparison purposes, we also incorporate a time window and a time decay function into the proposed IPLSI algorithm. As the online process does not have any knowledge of incoming documents, the statistics such as the IDF must be modified to process new vocabulary from such documents. For instance, the IDF is modified by substituting (12) as follows:

$$IDF(w,t) = \log \frac{N_t}{DF(w,t)}, \tag{23}$$

where t is the current time, N_t is the number of documents in the window at time t, and $DF(w,t)$ is the number of documents containing the term w at time t. The similarity between two documents in the same time window is modified by substituting (10) as follows, where $T(d)$ is the time stamp of document d:

$$sim(d_1, d_2) = \frac{|T(d_1) - T(d_2)|}{window\ size} \times \frac{\vec{d_1} \bullet \vec{d_2}}{|\vec{d_1}| \times |\vec{d_2}|}. \tag{24}$$

The similarity between a new document and an event is defined as the maximum similarity between the new document and documents previously clustered into the event. A document is deemed the first story of a new event if its similarity with all the events in the current window is below a predetermined threshold; otherwise, it is assigned to the event that is the most similar.

Following the conventions of the TDT contest, the window size of TW-DF is set to 500 source files, which covers a period of 30 to 40 days. The obtained detection error trade-off costs and F1 scores are shown in Figs. 1a and 1b, respectively. Based on the figures, we observe that the computational cost and the F1 score degrade sharply from their respective optimal thresholds. In practice, it is not possible to determine the true optimal threshold, because there is no knowledge about incoming and future news documents. Thus, to preserve the high performance for nonoptimal thresholds, an event analysis algorithm has to be resilient. We call the range of thresholds whose performance is within an error bound of the best performance the *Optimal Threshold-Resilient Range* (***OTRR***). For instance, the ***OTRR*** of the TW-DF method for TDT-4 for $\boldsymbol{F1} > 0.800$ is 0.09, and for $\boldsymbol{Cost} < 0.3$, it is 0.11. In the next section, we show that the IPLSI can provide a wider ***OTRR*** to alleviate the threshold-dependency problem.

5 Performance Evaluation

To evaluate the efficacy of the proposed IPLSI algorithm, we compare its performance with that of two traditional methods and three variants of the PLSI algorithm. The traditional methods are 1) the TW-DF algorithm [19], which was mentioned in Sections 1 and 4, and 2) the Evolutionary Clustering algorithm [6], which modifies traditional clustering methods so that they can support the evolutionary clustering task. For the EM algorithm, we use the modified K-Means method and pick the best performance from different change parameters cp for comparison (see [6] for further details).

The three PLSI variants are the original PLSI algorithm, the Naive-IPLSI algorithm, and the IPLSI algorithm that ignores all new words. As mentioned previously, the original PLSI algorithm is not an online algorithm, so all the documents for processing are required at the start time. Therefore, the original PLSI algorithm is expected to outperform all the online algorithms, because they do not have complete knowledge of all the documents for processing at the start time; that is, the online algorithms use the knowledge of documents chronologically. The second method is the Naive-IPLSI algorithm discussed in Section 3.2. The third method is the IPLSI algorithm that ignores all new words as the window advances. The window size for each of the three variants and IPLSI is set to 500 source files, which is identical to the setting of the TW-DF algorithm. The deferral periods are all set to 10 source files. In this experiment, the number of latent variables is set to 32. Experiments with different numbers of latent variables are discussed later in this section. We evaluated all the methods by using 99 threshold values (from 0.01 to 0.99). Then, for each method, we picked the threshold that yielded the best performance.

The results of the six methods evaluated on TDT-4 are listed in Table 3. The performance of the Evolutionary Clustering (K-Means) algorithm is the least accurate. This is because the objective of this algorithm is to maintain the consistency of clustering along a timeline, which may not be suitable for applications like event analysis, where an event's life cycle consists of the phases of creation, growth, and decay.

The performance of the original PLSI algorithm is second to that of the proposed IPLSI algorithm. It performs better than the remaining methods, because it ignores the chronological order of documents and uses the complete knowledge of all documents, including any available information about documents yet to be published. Even so, the IPLSI algorithm still slightly outperforms the original PLSI algorithm. The proposed IPLSI algorithm and the Naive-IPLSI algorithm are both variants of the PLSI algorithm, and they both use the EM algorithm to estimate

TABLE 3
Performance of the Six Evaluated Methods

Method	The Best Cost(threshold) Improvement over the baseline	The Best F1(threshold) Improvement over the baseline
TW-DF (baseline)	**0.207(0.14)**	**0.845(0.18)**
Evolutionary Clustering (k-means)	0.423	0.613
	-104.3%	-27.5%
Original PLSI (32)	0.181(0.32)	0.860(0.43)
	+12.6%	+1.8%
Naïve-IPLSI (32)	0.216(0.29)	0.851(0.33)
	-4.3%	+0.7%
IPLSI ignoring new words (32)	0.188(0.24)	0.852(0.27)
	+9.1%	+0.8%
IPLSI (32)	**0.178(0.24)**	**0.868(0.32)**
	+14.5%	**+2.7%**

latent variables. The latent variables generated by the Naive-IPLSI algorithm are discontinuous, whereas the latent variables generated by the IPLSI algorithm are continuous. This is indirect evidence that latent continuity can improve the performance.

The IPLSI algorithm clearly outperforms the baseline and the other *online* methods. The improvements in terms of the best $F1$ and *Cost* over the baseline are 2.7 percent and 14.5 percent, respectively.

The $F1$ performance of the Naive-IPLSI algorithm is slightly better than that of the baseline algorithm, but the *Cost* performance is not as good. In the Naive-IPLSI approach, the PLSI algorithm is applied for every advance of the window; hence, it has a long execution time and large memory space overhead compared to IPLSI. Thus, the Naive-IPLSI approach is unsuitable for online processing. The experiment was performed on a PC with an AMD Athlon 3200+ CPU, 2 Gbytes of memory, and the Windows XP Professional SP2 platform. Table 4 details the average execution times and the number of iterations required to achieve convergence. The computation time of the Naive-IPLSI approach is 10 times longer than that of the proposed IPLSI algorithm. For example, the total time of the Naive-IPLSI approach is 5,731 seconds (1:35:31), with the number of latent number K set at 16. However, for the same setting, the proposed IPLSI algorithm takes only 468 seconds (7:48), which is the sum of the PLSI time (4:13) and the folding-in time (3:35). IPLSI also converges much faster than the Naive-IPLSI approach.

To understand why the Naive-IPLSI approach does not perform substantially better than the baseline method, we investigate the continuity of the latent variables. We believe that the IPLSI algorithm's good convergence time is due to the fact that it successfully maintains the continuity of latent semantics, as it discards out-of-date documents and old terms and folds in new ones. We consider the issue of latent continuity in detail in Section 6.

TABLE 4
Execution Times of Naive IPLSI and IPLSI

K	Naïve-IPLSI		The Proposed IPLSI			
	Average Iterations	Total Time	Average Iterations	PLSI Time	Folding-in Time	Total Time
16	29.04	1:35:31	2.40	4:13	3:35	7:48
24	24.84	1:38:37	2.27	4:32	4:23	8:56
32	23.83	2:35:06	2.24	5.22	5.16	10:38
40	22.62	2:42:07	2.22	7:02	6:55	13:57
48	21.46	3:05:35	2.20	7:08	7:01	14:09

TABLE 5
The Results of the Proposed IPLSI Algorithm (Minimum Cost)

Method	K (# of latent variables)	Min Cost	Miss	False Alarm (%)	Optimal Threshold
TW-DF-35	N/A	0.207	0.200	0.151	0.17
IPLSI-16	16	0.162	0.154	0.149	0.24
IPLSI-24	24	0.171	0.162	0.178	0.22
IPLSI-32	32	0.178	0.170	0.157	0.24
IPLSI-40	40	0.177	0.170	0.154	0.25
IPLSI-48	48	0.175	0.166	0.193	0.22
IPLSI-*1	24, 32, 40	0.177	0.171	0.125	0.27
IPLSI-*2	16, 24, 32, 40, 48	0.168	0.160	0.169	0.25

To assess the influence of the number of latent variables used in the IPLSI algorithm, we set the number of variables at 16, 24, 32, 40, and 48. The results of the evaluations, as shown in Tables 5 and 6, demonstrate that the proposed IPLSI algorithm outperforms the TW-DF method by 1.85 percent and 14.73 percent for $F1$ and *Cost*, respectively, as mentioned previously.

We further examined the impact of different numbers of latent variables by averaging the conditional probabilities $P(w|d)$ of the models with different numbers of latent variables. In the experiment, we averaged the conditional probabilities of 24, 32, and 40 latent variables (denoted as IPLSI-*1 in Tables 5 and 6) and 16, 24, 32, 40, and 48 latent variables (denoted as IPLSI-*2 in Tables 5 and 6), as follows (both IPLSI-*1 and IPLSI-*2 outperformed the baseline approach in most cases, and in this evaluation, IPLSI is not sensitive to the number of latent variables):

$$P(w^*|d) = \frac{1}{3}[P_{24}(w^*|d) + P_{32}(w^*|d) + P_{40}(w^*|d)], \tag{25}$$

$$P(w^*|d) = \frac{1}{5}[P_{16}(w^*|d) + P_{24}(w^*|d) + P_{32}(w^*|d) + P_{40}(w^*|d) + P_{48}(w^*|d)]. \tag{26}$$

The F1 and Cost performances with different thresholds for the proposed IPLSI algorithm are shown in Fig. 5. We observe that the ranges of the OTRRs of the IPLSI algorithm are much larger than those of the baseline method. For example, Fig. 5a shows that for $\mathrm{Cost} = 0.3$, the OTRR of the TW-DF method is $0.22 - 0.11 = 0.11$, whereas the OTRR of IPLSI-32 is $0.42 - 0.17 = 0.25$. Similarly, Fig. 5b shows that for $\mathrm{F1} = 0.8$, the OTRR of the TW-DF

TABLE 6
The Results of the Proposed IPLSI Algorithm (Maximum F1)

Method	K (# of latent variables)	Max F1	Recall	Precision	Optimal Threshold
TW-DF-35	N/A	0.845	0.767	0.941	0.18
IPLSI-16	16	0.868	0.846	0.891	0.24
IPLSI-24	24	0.865	0.830	0.903	0.26
IPLSI-32	32	0.868	0.807	0.939	0.32
IPLSI-40	40	0.869	0.833	0.909	0.29
IPLSI-48	48	0.856	0.825	0.889	0.28
IPLSI-*1	24, 32, 40	0.869	0.825	0.918	0.28
IPLSI-*2	16, 24, 32, 40, 48	0.874	0.834	0.918	0.28

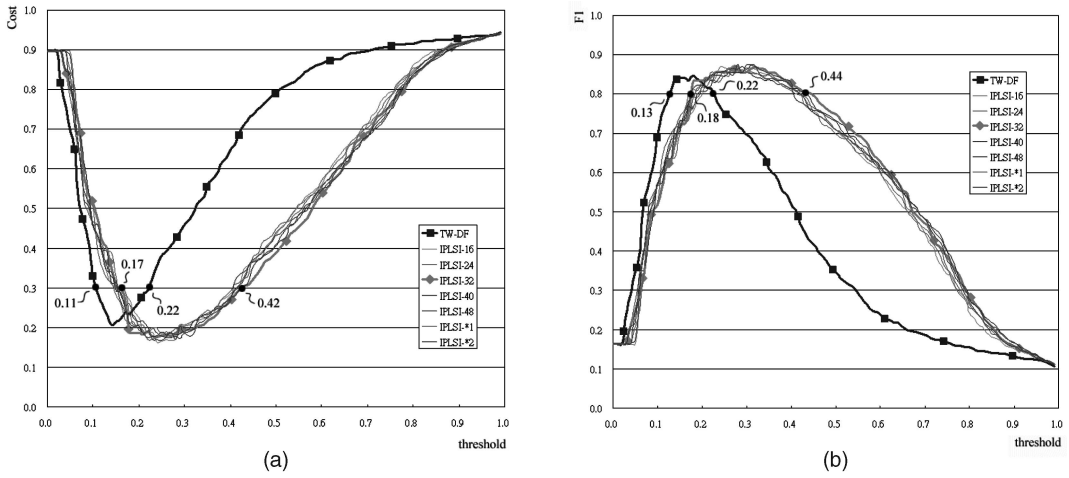

Fig. 5. (a) $Cost$ and (b) $F1$ of TW-DF and IPLSI.

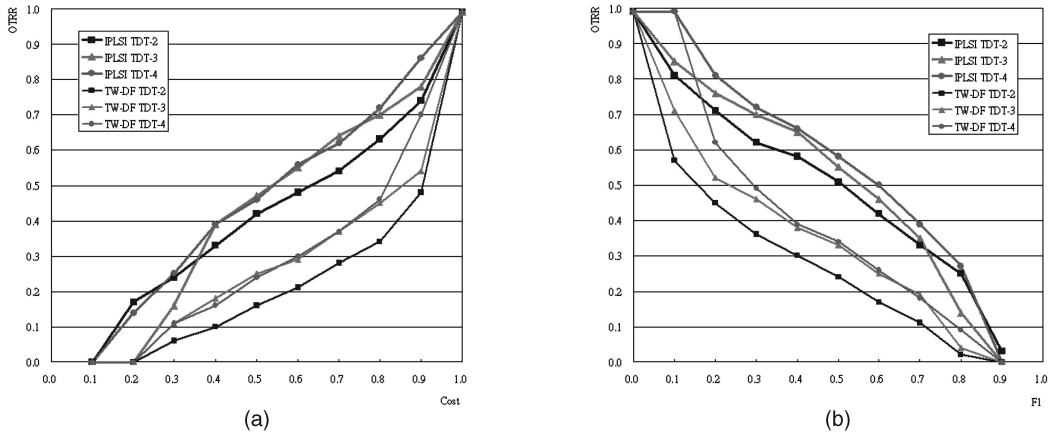

Fig. 6. OTRR of (a) $Cost$ and (b) $F1$ for TW-DF and IPLSI.

method is $0.22 - 0.13 = 0.09$, whereas the OTRR of IPLSI-32 is $0.44 - 0.18 = 0.26$.

To examine the changes in OTRR over a spectrum of values of the evaluation metrics (for example, ***F*1** and ***Cost***), we apply the TW-DF and IPLSI algorithms to the TDT-2, TDT-3, and TDT-4 corpora and record the ***OTRR*** values, as shown in Fig. 6. This figure shows that the ***OTRR*** of the proposed IPLSI algorithm for any value of ***Cost*** and ***F*1** is wider than that of the TW-DF method. In other words, the IPLSI algorithm is less dependent on the selected threshold to achieve an acceptable performance. Note that the OTRR properties of the other PLSI variants are similar. This positive characteristic is evidently inherited from the PLSI algorithm and is useful for real-world applications. In practice, the optimal threshold is usually unknown; thus, the proposed ILPSI algorithm can help alleviate the threshold-dependency problem and achieve a reasonable performance.

6 Discussion

With regard to the issue of the continuity of the latent variables, Figs. 7 and 8 show part of the evolution of the variables in the proposed IPLSI algorithm and the Naive-IPLSI approach, respectively. In these figures, the numbers in the large blocks are event IDs, and the numbers in the small blocks are latent IDs. For instance, in the upper left hand corner of Fig. 7, 40004 is the event ID, and 4 is the latent ID. To determine the event to which a latent variable belongs, we use the KL divergence rate [8] to measure the distance between the latent variables and events as follows:

$$KL(e\|z) = \sum_{w \in e} p(w|e) \log \frac{p(w|e)}{p(w|z)}, \tag{27}$$

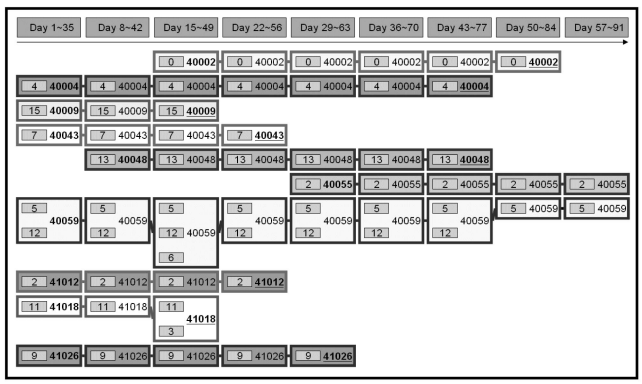

Fig. 7. Part of the evolution of the latent semantics in the proposed IPLSI algorithm.

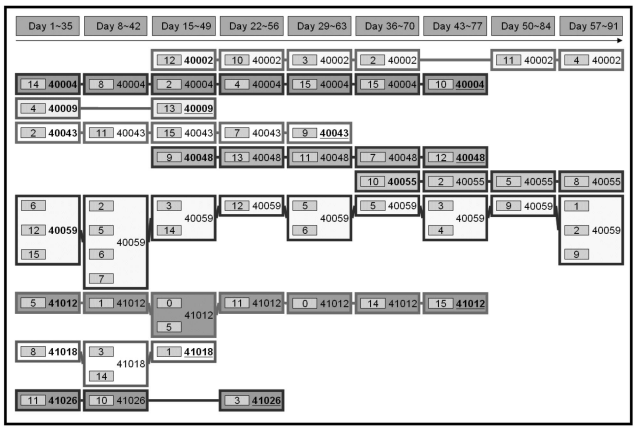

Fig. 8. Part of the evolution of the latent semantics in the Naive-IPLSI approach.

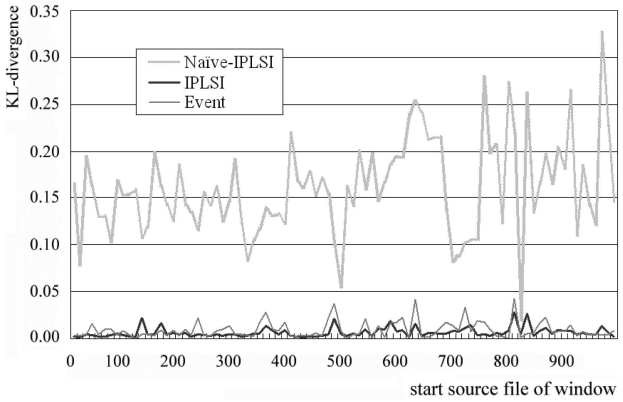

Fig. 9. Average KL divergence of events and latent variables for the same event.

where e is an event, and $p(w|e)$ is the normalized frequency of a word w that occurs in the documents of the event e. For a latent variable z, the event e with the smallest $KL(e\|z)$ below a certain threshold ξ is deemed the event to which z belongs. In other words, if all $KL(e\|z)$ are greater than ξ, then $E(z)$ does not exist; otherwise, $E(z) = \arg\min_e KL(e\|z)$. Thus, an event may be associated with more than one latent variable.

As shown in Fig. 7, the IPLSI algorithm successfully maintains the continuity of the latent semantics as the window advances. In contrast, the evolution of the latent semantics in the Naive-IPLSI approach is discontinuous, as shown in Fig. 8. This is because the Naive-PLSI approach recalculates the EM algorithm for each iteration; thus, it requires a large number of iterations to converge to a different local optimum after the window advances. However, in the proposed IPLSI algorithm, randomized initiation is only needed for the first window, and the latent semantics are adjusted by the discarding and folding-in processes as the window advances. Hence, the algorithm maintains the content of latent variables for each event. This explains why the Naive-IPLSI approach takes much longer to converge to the local optima than the proposed IPLSI algorithm.

Based on Figs. 7 and 8, we observe that the latent variables of the same event in the IPLSI algorithm are more continuous than the latent variables in the Naive-IPLSI approach. To measure the continuity, we calculate 1) the average KL divergence rate of the same real event in two adjacent time windows, as shown in (28), and 2) the two closest latent variables in two adjacent time windows of the same event for both the Naive-IPLSI approach and the IPLSI algorithm, as shown in (29).

$$KL_{AVG\_Event}(t) = \frac{1}{|C_t|} \sum_{e_t \in C_t, E(e_{t-1})=E(e_t)} \frac{KL(e_{t-1}\|e_t) + KL(e_t\|e_{t-1})}{2} \quad (28)$$

$$KL_{AVG\_Latent}(t) = \frac{1}{|C_t|} \sum_{e_t \in C_t} \min_{\substack{for\ all\ z_{t-1}, z_t \\ where\ E(z_{t-1})=E(z_t)=E(e_t)}} \frac{KL(z_{t-1}\|z_t) + KL(z_t\|z_{t-1})}{2}. \quad (29)$$

In the above equations, C_t is a set of events that occur in time window t and time window $t-1$ simultaneously, $|C_t|$ is the number of sets, z_t and z_{t-1} are latent variables produced in time window t and time window $t-1$, respectively, e_t and e_{t-1} are events in time window t and time window $t-1$, respectively, and $E(e_t)$ denotes the event ID of e_t. Note that comparing the differences among all time windows via an asymmetric distance measure, that is, the KL divergence, is not feasible; therefore, we average $KL(e_{t-1}\|e_t)$ and $KL(e_t\|et-1)$ in (28), and $KL(z_{t-1}\|z_t)$ and $KL(z_t\|z_{t-1})$ in (29). Based on Fig. 9, we observe that for the proposed IPLSI algorithm, the average KL divergence rates between latent variables in adjacent windows of the same event are much lower than those for the *Naive-IPLSI* approach. They are also much closer to the average KL divergence rates of events between adjacent windows than those in the *Naive-IPLSI* approach. Furthermore, the continuity of the content in the latent variables generated by the *Naive-IPLSI* approach is clearly very unstable, whereas the proposed IPLSI algorithm maintains good continuity in the content of latent variables and in the content of real events.

7 Conclusions

Event analysis is a challenging research topic that has many applications such as "hot" news stories provided by Internet portals, Internet event detection, e-mail event detection [16], and discussion board topic detection. The challenge of online event analysis is to detect unknown events and track their story line development from a continuous document stream generated by uncoordinated information sources. However, conventional text classification methods do not perform the tasks well, because the temporal relationships among documents are difficult to handle. The proposed IPLSI algorithm not only improves event detection and reduces the computation time but also alleviates the *threshold-dependency* problem, which traditional event analysis methods do not consider. The performance of such methods depends on optimal thresholds that are usually unknown in practice. Even though the thresholds are obtained by using training data sets, which is a popular text classification method, the story lines of events develop over time; hence, the training data sets become unrepresentative. The proposed IPLSI algorithm

alleviates the threshold-dependency problem in online event analysis tasks. Furthermore, the algorithm successfully maintains the continuity of the latent semantics along the time line and thus ensures the quality of event detection.

ACKNOWLEDGMENTS

Meng Chang Chen is the corresponding author. The authors wish to thank the anonymous reviewers for their valuable and constructive comments, which have helped improve the quality of this paper. This work was supported in part by the National Science Council of Taiwan under Grants 94-2524-S-001-001 and 95-2524-S-001-001 and by the National Digital Archives Program, Taiwan.

REFERENCES

[1] J. Allan, J. 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.
[2] J. Allan, R. Papka, and V. Lavrenko, "Online New Event Detection and Tracking," *Proc. ACM SIGIR '98,* 1998.
[3] D.M. Blei and P.J. Moreno, "Topic Segmentation with an Aspect Hidden Markov Model," *Proc. ACM SIGIR '01,* 2001.
[4] T. Brants, F. Chen, and I. Tsochantaridis, "Topic-Based Document Segmentation with Probabilistic Latent Semantic Analysis," *Proc. 11th ACM Int'l Conf. Information and Knowledge Management (CIKM '02),* 2002.
[5] T. Brants and F. Chen, "A System for New Event Detection," *Proc. ACM SIGIR '03,* 2003.
[6] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary Clustering," *Proc. ACM SIGKDD '06,* 2006.
[7] C.C. Chen, Y.T Chen, and M.C. Chen, "An Aging Theory for Event Life Cycle Modeling," *IEEE Trans. Systems, Man, and Cybernetics Part A,* vol. 37, no. 2, pp. 237-248, Mar. 2007.
[8] *Language Modeling and Information Retrieval,* WB Croft and J. Lafferty, eds. Kluwer Academic Publishers, 2003.
[9] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. B,* vol. 39, pp. 1-38, 1977.
[10] T. Hofmann, "Probabilistic Latent Semantic Indexing," *Proc. ACM SIGIR '99,* 1999.
[11] T. Hofmann, "Unsupervised Learning by Probabilistic Latent Semantic Analysis," *Machine Learning,* vol. 42, pp. 177-196, 2001.
[12] X. Jin, Y. Zhou, and B. Mobasher, "Web Usage Mining Based on Probabilistic Latent Semantic Analysis," *Proc. ACM SIGKDD '04,* 2004.
[13] Z.W. Li, B. Wang, M.J. Li, and W.Y. Ma, "A Probabilistic Model for Retrospective News Event Detection," *Proc. ACM SIGIR '05,* 2005.
[14] R. Manmatha, A. Feng, and J. Allan, "A Critical Examination of TDT's Cost Function," *Proc. ACM SIGIR '02,* 2002.
[15] Q. Mei and C.X. Zhai, "Discovering Evolutionary Theme Patterns from Text: An Exploration of Temporal Text Mining," *Proc. ACM SIGKDD '05,* 2005.
[16] S. Morinaga and K. Yamanishi, "Tracking Dynamics of Topic Trends Using a Finite Mixture Model," *Proc. ACM SIGKDD '04,* 2004.
[17] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, 1989.
[18] A. Surendran and S. Sra, "Incremental Aspect Models for Mining Document Streams," *Proc. 17th European Conf. Machine Learning (ECML '06),* 2006.
[19] Y. Yang, T. Pierce, and J. Carbonell, "A Study on Retrospective and Online Event Detection," *Proc. ACM SIGIR '98,* 1998.
[20] J. Zhang, Y. Yang, and J. Carbonell, "New Event Detection with Nearest Neighbor, Support Vector Machines and Kernel Regression," Technical Report CMU-CS-04-118 (CMU-LTI-04-180), Carnegie Mellon Univ., 2007.
[21] J. Zhang, Z. Ghahramani, and Y. Yang, "A Probabilistic Model for Online Document Clustering with Application to Novelty Detection," *Proc. Conf. Neural Information Processing Systems (NIPS '04),* 2004.
[22] NIST Topic Detection and Tracking Corpus, http://www.nist.gov/speech/tests/tdt/tdt98/index.htm, 1998.

Tzu-Chuan Chou received the MS and PhD degrees in computer science and information engineering from Tamkang University, Taipei, in 1998 and 2004, respectively. He is currently a postdoctoral fellow in the Institute of Information Science, Academia Sinica, Taiwan. His research interests include clustering algorithms, information retrieval, image compression, and prediction market.

Meng Chang Chen received the BS and MS degrees in computer science from the National Chiao-Tung University, Taiwan, in 1979 and 1981, respectively, and the PhD degree in computer science from the University of California, Los Angeles, in 1989. He joined AT&T Bell Laboratories in 1989 as a member of technical staff and led several R&D projects in the area of data quality of distributed databases for mission critical systems. From 1992 to 1993, he was an associate professor at the National Sun Yat-Sen University, Taiwan. Since then, he has been with the Institute of Information Science, Academia Sinica, Taiwan, where he is currently a research fellow. He was the deputy director from August 1999 to July 2002. For three years (from 2001), he was the chair of the Standards and Technology Transfer Group, National Science and Technology Program for Telecommunications Office (NTPO), Taiwan. His current research interests include information retrieval, knowledge management and engineering, wireless network, QoS networking, and operating systems.