Fast Mining and Forecasting of Complex Time-Stamped Events

Yasuko Matsubara
Kyoto University
y.matsubara@
db.soc.i.kyoto-u.ac.jp

Yasushi Sakurai
NTT Communication
Science Labs
yasushi.sakurai@acm.org

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

Tomoharu Iwata
NTT Communication
Science Labs
iwata.tomoharu@lab.ntt.co.jp

Masatoshi Yoshikawa
Kyoto University
yoshikawa@i.kyoto-u.ac.jp

ABSTRACT

Given huge collections of time-evolving events such as web-click logs, which consist of multiple attributes (e.g., URL, userID, timestamp), how do we find patterns and trends? How do we go about capturing daily patterns and forecasting future events? We need two properties: (a) effectiveness, that is, the patterns should help us understand the data, discover groups, and enable forecasting, and (b) scalability, that is, the method should be linear with the data size.

We introduce *TriMine*, which performs three-way mining for all three attributes, namely, URLs, users, and time. Specifically *TriMine* discovers hidden topics, groups of URLs, and groups of users, simultaneously. Thanks to its concise but effective summarization, it makes it possible to accomplish the most challenging and important task, namely, to forecast future events. Extensive experiments on real datasets demonstrate that *TriMine* discovers meaningful topics and makes long-range forecasts, which are notoriously difficult to achieve. In fact, *TriMine* consistently outperforms the best state-of-the-art existing methods in terms of accuracy and execution speed (up to *74x faster*).

Categories and Subject Descriptors: H.2.8 [**Database management**]: Database applications–*Data mining*

General Terms: Algorithms, Experimentation, Performance

Keywords: Time-stamped events, Tensor analysis, Topic model, Forecasting

1. INTRODUCTION

Our motivating application is to find patterns and trends in a large set of clicks, consisting of multiple attributes (URL, user ID, timestamp, access devices, http/document referrer, etc). Are there emerging topics? Can we group URLs (and users), accordingly? How many clicks should we expect from user 'Smith', tomorrow?

KDD'12, August 12–16, 2012, Beijing, China.

We shall refer to such settings as a sequence of *complex events*. Each event has a timestamp, and has multiple categorical attributes associated with it - in our example, a URL, which we shall refer to as an '*object*', and a userID as an '*actor*'. Such settings naturally appear in countless domains including click logs on web sites [1], social network services, location-based services [15], social tagging, and many more. Informally, if each event has a time stamp and two attributes, the problem we want to solve is that described below.

INFORMAL PROBLEM 1 (COMPLEX EVENT MINING). *Given a sequence of millions of triplets (object, actor, time stamp),*

- *find the major topics and their trends*
- *make traffic forecasts*

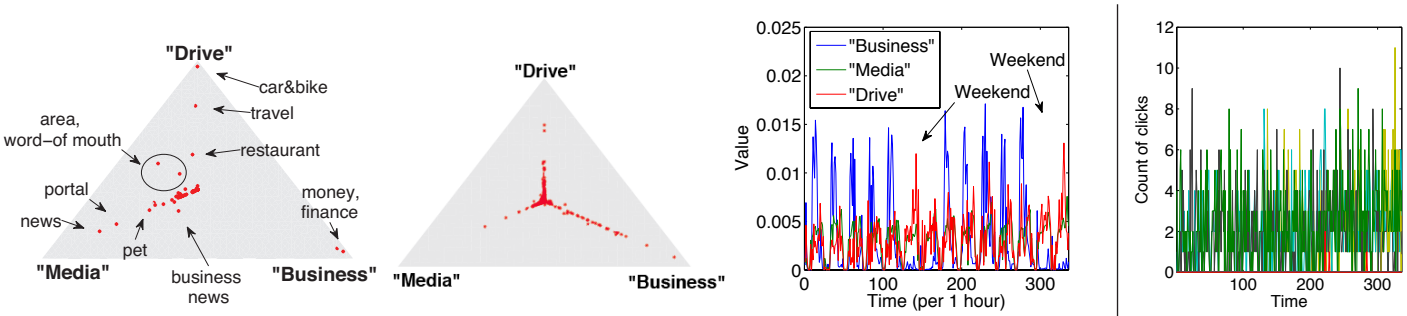
We can generalize for an arbitrary number of attributes (i.e., more than three attributes) and we describe this later. The ideal method should be (a) *effective* in finding patterns, (b) *accurate* in forecasting, and (c) *scalable*; it should help us understand what are the major trends, it should provide an easy-to-understand summary, it should help spot anomalies (with respect to *all three* aspects - URLs, users, time); and to visualize the results; and of course, the processing should be fast, and ideally, linear with the input size.

We present a novel method, *TriMine*, which automatically finds patterns in huge collections of complex events. Specifically, it finds three-way patterns (hence the prefix): It finds hidden topics (such as, 'sports', 'news'), and correlates each topic with all three aspects.

Preview of our results. Figure 1 shows the results we obtained with *TriMine* on real web-click logs, where each click is of the form $(URL, userID, timestamp)$. We will see this collection of plots so often that we have given it a name, "*TriMine*-plot". This plot consists of two *ternary* plots, and a time plot, as described later.

In general, each web site may cover one or more topic(s), and each user would be associated with a few topics. Thus, we could expect both finance news and stock market sites to attract roughly the same users, with similar temporal patterns, and so those sites should be grouped under the same topic, say, 'business'.

Figure 1 shows a *TriMine*-plot with some of our findings, namely it detects/reveals the hidden topics of web click events. More specifically, the figure shows the three major topics, and how they related to each of the three aspects, namely (a) to each web site (URL), expressed in the $\mathbf{O}$ matrix, (b) to each user (in the $\mathbf{A}$ matrix), and (c)



(a) URLs in topic space (object matrix $\mathbf{O}$)

(b) Users in topic space (actor matrix $\mathbf{A}$)

(c) Time evolution in topic space (time matrix $\mathbf{C}$)

(d) Original sequence of 100 random users

Figure 1: *TriMine* is effective in spotting three-way patterns. Results for web-click data (URL, userID, timestamp), with three hidden topics, manually labeled as shown. (a) Ternary plot for URLs - notice the three groups along the center-vertex spokes, (b) ditto for users - showing very clear groups along the spokes, (c) temporal evolution of the three topics: 'business' traffic (in blue) drops during weekends; 'drive/travel' traffic (in red) *spikes* during weekends; and 'media/news' traffic (in green) remains the same. All show daily periodicity. (d) original counts of "money" site, 100 randomly selected users, appears bursty and noisy.

to each time-tick ($\mathbf{C}$ matrix). The first two are represented as triangular plots (also known as 'ternary plots'), where each 3-d point is represented as a point in a triangle (since the sum of the 3 coordinates is less than 1).

TriMine successfully detects some of the hidden topics in the web click logs, and we have manually named them 'business', 'media' and 'drive'. The first ternary plot shows the URLs; notice that several of them focus almost exclusively on the topic 'drive', several others on 'media/news', a few focus exclusively on 'business'. The ternary plot of the users shows an even clearer separation: most dots (= users) are on the spokes from the center to the vertices, indicating a very clear clustering of users, along the three hidden topics. Neither of the two ternary plots exhibits any striking outliers.

The third plot (Figure 1 (c)), shows the evolution of each topic over time (red, green, blue, for 'drive', 'media', 'business' respectively). Notice a clear daily periodicity for all the topics (unsurprising), but also notice (1) the decline in 'blue' (='business') during weekends, and (2) the spiking of 'red' (='drive/traveling') during weekends. In retrospect, both make sense. Also notice that 'green' (= 'media/news') does *not* have a weekly periodicity, apparently because our users are constantly interested in news.

In contrast, Figure 1 (d) shows the original raw sequences of 100 randomly selected users (at a "money" site). Notice that the raw data exhibits no obvious patterns, neither periodicities nor clusters of users.

Our main point is that the two 'ternary plots' and a simple timeplot can give a clear, concise summary of both '*actors*' (users) and '*objects*' (URLs), as well as of the major temporal patterns. Thanks to these three building blocks, we can extend this approach to complex event forecasting, which is notoriously difficult to achieve.

Contrast with competitors. Table 1 compares *TriMine* and its forecasting method, *TriMine-F*, with existing methods. We illustrate their strengths and shortcomings: a check mark ($\surd$) indicates that the corresponding method (column) fulfills the corresponding requirement (row). Only our approach has checks against all entries, while,

- Wavelet transforms can identify multi-scale trends in a *"single"* sequence as regards objects or actors, but find it hard to determine the *"relationship"* between all these objects/actors.
- Complex events can be turned into a tensor. Higher-order SVD (HOSVD) [12] and alternating least squares (ALS) [23] capture the hidden components from tensors, but they cannot handle categorical/non-Gaussian data or perform forecasting.
- Topic models (e.g., latent Dirichlet allocation (LDA) [4]) are probabilistic models for sparse vectors of count data, and they can find hidden topics and perform clustering. However, they are not intended to capture cyclic time-evolving patterns, or undertake forecasting.
- Auto regression (AR) and *PLiF* [13] have the ability to forecast sequences, however, they cannot handle multi-scale sequences, and so cannot perform long-term forecasting, because they converge quickly to the mean (see Figures 7 and 8).

Table 1: Capabilities of approaches. Only our approach meets all specifications.

	DWT	HOSVD /ALS	LDA	*AR /PLiF*	*TriMine /TriMine-F*
Multi-scale	$\surd$				$\surd$
Tensor		$\surd$			$\surd$
Categorical			$\surd$		$\surd$
Short-term forecasting				$\surd$	$\surd$
Long-term forecasting					$\surd$

Contributions. The proposed method has the following advantages:

1. *Effectiveness: TriMine* operates on large collections of complex events, summarizes them succinctly, gives three-way patterns (i.e., with respect to *objects*, *actors*, and time stamps), and enables forecasting, clustering and anomaly detection.
2. *Accuracy:* as we will show, our approach is accurate in forecasting complex time-stamped events.
3. *Scalability: TriMine* is linear with the input size, and thus scales up very well.

The rest of the paper is organized in a typical way: Next we describe related work, followed by definitions, the proposed method, experiments and conclusions.

2. RELATED WORK

Recently significant progress has been made on understanding the theoretical issues surrounding learning mixture distributions

in theoretical computing and machine learning [9, 28, 22]. Latent Dirichlet allocation (LDA) [4] and probabilistic latent semantic analysis (PLSA) [5] are well-known latent variable models for analyzing large sets of categorical data, such as bags of words for text, and bags of features for images. A number of methods have been proposed for analyzing the time evolution of topics in document collections, such as the dynamic topic model (DTM) [3], topics over time (TOT) [24], and more [25, 2, 27, 7, 8, 6]. For example, DTM employs a fixed length of window size, and takes account of capturing only the current epoch distribution. Similarly, TOT handles a single window size, and it uses Beta distributions to capture time-evolving topics. Very recently, Hong et al. presented a new topic model for predicting the volume of terms in documents (i.e., aggregate count of each keyword). These models do not focus on finding multiscale and/or periodical trends, which means that it is difficult to employ them for the long-term forecasting of complex time-stamped events. On the other hand, as shown in the introduction section (see Table 1), our method finds cyclic patterns with different timescales, which allows it to predict future events effectively and efficiently.

For web-click analysis, Agarwal et al. [1] exploit the Gamma-Poisson model to estimate click-through rates (number of clicks per display) in the context of content recommendation, which does not focus on trend discovery of time-stamped events.

The work on tensors is also related. Kolda et al. [10] provide a powerful tool for tensor analysis on a web link structure. Rendle et al. [19] propose a method for tag recommendation based on tensor factorization. Unlike our method, these methods are not intended to predict future events.

Remotely related is the work on large-scale time-series mining. Similarity search and pattern discovery in time sequences have also attracted huge interest [20, 14]. Papadimitriou et al. [17] propose an algorithm for discovering multi-scale local patterns, which concisely describes the main trends in data streams. Skewed stream problems have been studied in the context of summarization and modeling [11]. Sakurai et al. [21] proposed BRAID, which efficiently detects lag correlations between data streams. AWSOM [16] is one of the first streaming methods for forecasting and is designed to discover arbitrary periodicities in single time sequences.

3. PROBLEM FORMULATION

In this section, we formally define related concepts and the task of event forecasting. Consider that we receive time-stamped event entries of the form *(object, actor, timestamp)*. We then have a collection of entries with u unique objects and v unique actors. Assume that we are given time intervals t $(= 1, 2, \ldots, n)$ of length l (e.g., one hour). It is convenient to treat our complex-event sequence as a 3rd-order tensor, i.e., $\mathcal{X} \in \mathbb{N}^{u \times v \times n}$, where n is the duration of events.

DEFINITION 1 (COMPLEX EVENT TENSOR). *Let $\mathcal{X}$ be a 3rd-order tensor of complex time-stamped events. The element $x_{i,j,t}$ of $\mathcal{X}$ shows the total number of event entries of the i-th object and the j-th actor at time interval t.*

After we decide on some time granularity (say, one hour) then we have (actor, object, time-stamp; count), for example, ('Smith', 'cnn.com', '3am June 1, 2003'; 23). Unless otherwise specified, our time granularity is one hour. Thus, the example tuple above means that 'smith' visited 'cnn.com' 23 times, between 3am-4am on June 1, 2003.

For a particular event collection, we assume that an event entry has a certain "*latent topic*". We model such hidden topics in terms of three aspects, namely, "*object*", "*actor*", and "time". In that case, the original tensor will be decomposed in three matrices $\mathbf{O}$, $\mathbf{A}$, $\mathbf{C}$ with the following definitions and properties:

Table 2: Symbols and definitions.

Symbol	Definition
u	Number of unique objects
v	Number of unique actors
n	Duration: number of time-ticks
$\mathcal{X}$	3rd-order tensor of complex time-stamped events ($\mathcal{X} \in \mathbb{N}^{u \times v \times n}$)
k	Number of hidden topics
$\mathbf{O}$	Object matrix, $u \times k$
$\mathbf{A}$	Actor matrix, $k \times v$
$\mathbf{C}$	Time matrix, $k \times n$

DEFINITION 2 (OBJECT MATRIX $\mathbf{O}$ $(u \times k)$). *Each entry $o_{i,j}$ shows the participation strength of object i for topic j.*

Our upcoming *TriMine* forces that the participation weights $o_{i,j}$ to be non-negative, and they sum up to 1, for each object ($\sum_j o_{i,j} = 1$). The definitions of $\mathbf{A}$ and $\mathbf{C}$ are analogous, and omitted for brevity. We shall refer to $\mathbf{O}$, $\mathbf{A}$ and $\mathbf{C}$ as *participation matrices*, exactly because they show how strongly each *actor*, *object*, time stamp participates in topics #1, #2, ..., #k, respectively. Figure 1 plots the visualization of the three matrices for a real dataset (we show three major topics).

We also consider a case where every event has more than three attributes (i.e., $M > 3$). We provide an Mth-order tensor, which is decomposed in M matrices, $\mathbf{O}$, $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(M-2)}$, and $\mathbf{C}$. We mainly focus on a 3rd-order tensor for simplicity throughout this paper, our method however can be applied to higher order tensors. Table 2 summarizes the notation correspondences for the time-stamped events.

3.1 Problem definition

Our goal is to extract the main trends and hidden patterns of an event tensor $\mathcal{X}$, and forecast future events. Specifically, the subproblem that we want to solve is as follows:

PROBLEM 1 (PATTERN DISCOVERY IN COMPLEX EVENTS). **Given** *a tensor $\mathcal{X}$ of complex events of (actor, object, time stamp) triplets,* **Find** *the hidden topics that best summarize the events in $\mathcal{X}$, and the corresponding actor- and object- groupings.*

Once we have the main trends from $\mathcal{X}$, we can proceed to solve the forecasting problem:

PROBLEM 2 (TRAFFIC FORECASTING). **Given** *a tensor $\mathcal{X}$ of complex events,* **Forecast** *future traffic.*

More specifically, we want to forecast, e.g., how many clicks user 'Smith' will generate tomorrow; or how many clicks 'www.cnn.com' will attract over the next seven days; or we want to generate a realistic-looking set of clicks, for, say, next Sunday.

As mentioned in the introduction, the ideal method should be *effective* (capturing trends that make sense), *accurate* in its forecasts, and *scalable*, to handle millions or billions of events.

3.2 Running examples

There are many applications for time-stamped events. Here, we briefly describe application domains and provide some illustrative, intuitive examples of the usefulness of our approach.

Access analysis. Web content such as blogs, on-line news, web mail and SNS has been attracting considerable interest for business

purposes as well as personal use. Consider a large number of clicks on web sites, where each click has a list of attributes, e.g., *access (url_id, user_id, time)*. Suppose that we can record all these attributes on an hourly basis for all users of the web sites. For this huge collection of web-click logs, we would like to find patterns of user behavior to allow us to perform a sociological and behavioral analysis. For example, web masters and web-site owners could find daily access patterns and forecast the subsequent week to aid the design of advertisements.

E-commerce. Let us assume an online service such as ondemand TV, which records the TV programs viewed by every user; each record is an online viewing event of form *view (channel_id, user_id, time)*. Discovering groups in such data and forecasting future viewing events would assist with tasks such as content targeting.

4. PROPOSED METHOD

In this section we describe our method, *TriMine*, for dealing with the pattern discovery problem (Problem 1). In the next section we show how to employ *TriMine* results for forecasting (Problem 2).

4.1 Intuition behind our method

There are two main ideas behind *TriMine*:

- M*-way analysis*: For a single time granularity (say, l_0=1 hour), we perform an M-way topic analysis, which discovers k topics and shows the relationship between these topics and the M matrices. As an example of a 3-way case, we show how the topics relate to the *objects* (*object* matrix $\mathbf{O}$, $u \times k$), to the *actors* (*actor* matrix $\mathbf{A}$, $k \times v$), and to time-ticks (time matrix $\mathbf{C}$, $k \times n$).
- *Multi-scale analysis*: To achieve better forecasting (see Figures 7 and 8), and to capture longer-period trends, we repeat the analysis, for several time granularities $\{\mathbf{C}^{(0)}, \mathbf{C}^{(1)}, \dots\}$ (say, hours, days, etc), insisting on *fixed object* and *actor* matrices $\mathbf{O}$, $\mathbf{A}$.

Figure 1(a)-(c) illustrates the $\mathbf{O}$, $\mathbf{A}$ and $\mathbf{C}$ matrices, respectively, for the *WebClick* dataset. *objects* and *actors* are categorical, and the $\mathbf{O}$ and $\mathbf{A}$ matrices are visualized as ternary plots, where each dot is an *object* / *actor*. The three columns of the time matrix are plotted in (c), illustrating the temporal evolution of the three discovered topics.

Single-level analysis - ***TriMine-single*****.** Figure 2 (dotted-line rectangle) gives a pictorial description of the 3-way analysis. The object matrix $\mathbf{O}$ takes account of all the objects over the entire time range, which produces an accurate summary of the object-topic relationship. The actor matrix $\mathbf{A}$ shows the frequency of entries over actors for the i-th topic ($i = 1, \dots, k$). The time matrix $\mathbf{C}$ describes the degree to which the time stamp is associated with the i-th topic.

Multiple-level analysis - ***TriMine*****.** Figure 2 (solid-boundary rectangle) also shows an example to explain how to perform the multi-level analysis. Instead of handling only a single window size, we introduce a multiple time window approach to capture the main trends of multiple scales. Starting with the window size l_0 on the first level $h = 0$, we compute the time matrix $\mathbf{C}^{(0)}$, then increase the window size l_h, and repeatedly obtain $\mathbf{C}^{(h)}$ for various sizes. This enables the forecasting method (discussed later in Section 5) to exploit the relationship between $\mathbf{C}^{(h)}$ across different scales.

4.2 Proposed solution - TriMine

Given an Mth-order complex event tensor $\mathcal{X}$, our goal is to find a meaningful summary with k hidden topics that best describes $\mathcal{X}$.

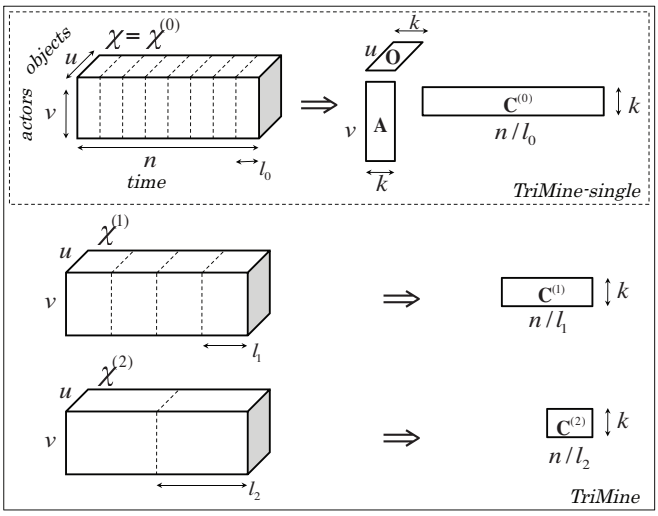


Figure 2: Illustration of ***TriMine*****.** ***TriMine-single*** **(dotted-line rectangle): We extract k topics from an event tensor $\mathcal{X}$ with respect to three aspects, i.e., objects $\mathbf{O}$, actors $\mathbf{A}$, and time $\mathbf{C}$.** ***TriMine*** **(full, solid-boundary rectangle): We perform a 3-way analysis for multiple window sizes l_0, l_1, l_2, ... to capture the multi-scale dynamics of $\mathcal{X}$. Note that we reuse the $\mathbf{O}$ and $\mathbf{A}$ matrices for all scale levels to provide an efficient solution.**

More specifically, we want to compute the "interpretable features" with respect to all M aspects. We propose using the concept of topic modeling to extract the major factors automatically. *TriMine* infers the M matrices, i.e., the *object* matrix $\mathbf{O}$, the *actor* matrices $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(M-2)}$, and the time matrix $\mathbf{C}$, and then identifies the relationship between the properties of these matrices. It can also discard many redundancies (e.g., noise) from the events, and focus on what really matters.

4.2.1 Single level inference

The task is to infer the M matrices over the hidden topics. We assume that each event entry has its own "latent topic", and thus we propose using collapsed Gibbs sampling [18] to assign a latent topic for each event entry. The generative process for a set of event entries is as follows:

1. For each topic $r = 1, \dots, k$:
   (a) For each tensor mode $m = 1, \dots, M-2$:
      i. Draw $\mathbf{A}_r^{(m)} \sim$ Dirichlet($\beta^{(m)}$).
   (b) Draw $\mathbf{C}_r \sim$ Dirichlet(γ).
2. For each object $i = 1, \dots, u$:
   (a) Draw $\mathbf{O}_i \sim$ Dirichlet(α).
   (b) For each entry $j = 1, \dots, N_i$:
      i. Draw a latent variable $z_{i,j} \sim$ Multinomial($\mathbf{O}_i$).
      ii. For each tensor mode $m = 1, \dots, M-2$:
         A. Draw an actor $e_{i,j}^{(m)} \sim$ Multinomial($\mathbf{A}_{z_{i,j}}^{(m)}$).
      iii. Draw a timestamp $t_{i,j} \sim$ Multinomial($\mathbf{C}_{z_{i,j}}$).

Here, α, $\beta^{(m)}$ and γ are the hyperparameters for $\mathbf{O}$, $\mathbf{A}^{(m)}$ and $\mathbf{C}$, respectively.

For simplicity, hereafter we focus on a 3rd-order tensor (i.e., $M = 3$). For each non-zero element $x_{i,j,t}$ in $\mathcal{X}$, we draw $x_{i,j,t}$ latent variables with probability p (Equation 1), and decide the latent

variable for every event entry. The latent variable $z_{i,j,t}$ is assigned for element $x_{i,j,t}$ in the following manner:

$$p(z_{i,j,t} = r|\mathcal{X}, \mathbf{O}', \mathbf{A}', \mathbf{C}', \alpha, \beta, \gamma) \propto \frac{o'_{i,r} + \alpha}{\sum_r o'_{i,r} + \alpha k} \cdot \frac{a'_{r,j} + \beta}{\sum_j a'_{r,j} + \beta v} \cdot \frac{c'_{r,t} + \gamma}{\sum_t c'_{r,t} + \gamma n} \quad (1)$$

where $o'_{i,r}$, $a'_{r,j}$, and $c'_{r,t}$ are the total counts that topic r is assigned to the i-th *object*, j-th *actor*, and time-tick t, respectively. Note that the prime (e.g., $o'_{i,r}$) indicates that the current datum has been excluded from the count summations, that is, it indicates the count with the entry of the i-th object and the j-th entry at time t removed. Here, α, β, and γ are the parameters of the Dirichlet priors for $\mathbf{O}$, $\mathbf{A}$, and $\mathbf{C}$, respectively. After the sampler has burned-in, we can produce estimates of $\tilde{\mathbf{O}}$, $\tilde{\mathbf{A}}$, and $\tilde{\mathbf{C}}$ as follows:

$$\tilde{o}_{i,r} \propto \frac{o_{i,r} + \alpha}{\sum_r o_{i,r} + \alpha k}, \quad \tilde{a}_{r,j} \propto \frac{a_{r,j} + \beta}{\sum_j a_{r,j} + \beta v}, \quad \tilde{c}_{r,t} \propto \frac{c_{r,t} + \gamma}{\sum_t c_{r,t} + \gamma n}. \quad (2)$$

The time complexity of each sampling iteration is $O(N)$, where N is the total number of event entries in $\mathcal{X}$ (i.e., $\sum_{i,j,t} x_{i,j,t}$).

4.2.2 Full TriMine: multiple levels

Until now, we have assumed that the window size (say, l) was given. In reality, it is up to us to choose it. What is the right value for it? minute? hour? day? This is a difficult choice - our approach is to *not* make a choice, and use all of the above. More specifically, we use several window sizes. Our approach makes the algorithm slightly more complicated, but it pays off significantly in terms of long-term forecasting accuracy, as our experiments show (see Figures 7 and 8).

In short, as a typical choice, we use a set of all such window sizes, i.e., an hour, a day, a week, a month. Another reasonable idea is that we use a geometric progression of window sizes, that is $l_h := l_0 \cdot L^h$ for $h = 0, 1, 2, ..., \lceil \log n \rceil$, where l_0 is the shortest window, and L is the growth factor, typically $L = 2$. Thus, in either case, the count of the window set $\mathcal{L}$ that we need to examine is greatly reduced, as compared, say, to an arithmetic progression.

Here, we explain how to perform the necessary computations for multi-scale trend discovery. The straightforward solution is that we consider a set of tensors $\{\mathcal{X}^{(0)}, \mathcal{X}^{(1)} \dots\}$ for all window sizes, and compute *TriMine-single*, for each level. We refer to this approach as *TriMine* (naive). Still, this is computationally expensive because we need an inference for each window.

Efficient solution. We propose reusing the inference results of the first level to approximate the inference for the other levels. We show how our method performs the computations in Figure 2. For level $h = 0$, we compute the matrices, $\mathbf{O}$, $\mathbf{A}$, and $\mathbf{C}^{(0)}$ with the tensor $\mathcal{X}^{(0)}$ (= $\mathcal{X}$). For the other levels ($h \geq 1$), we reuse the results of the sampling in the first level, and compute the current matrices, i.e., (a) we share $\mathbf{O}$ and $\mathbf{A}$ for all levels, and (b) compute the time matrix $\mathbf{C}^{(h)}$ by using the set of sampling results for the first level, as follows:

$$c^{(h)}_{r,t} \propto \sum_{i=1}^{l_h} c^{(0)}_{r,t-l_h+i} \quad (3)$$

where l_h denotes the window size at level h. The justification is that the assignment for each event entry is probabilistically equivalent for all levels, which means that the latent topics of every level are computed by the same procedure. Compared with *TriMine* (naive), which needs $O(N \log n)$ time to update the parameters for all levels of the structure, the efficient version of *TriMine* is linear with the input size, i.e, $O(N)$.

The overall procedure of *TriMine* is given by Algorithm 1. For each entry in $\mathcal{X}^{(0)}$ at the first level, we assign a hidden variable z according to Equation (1). After the sampling, we compute the triplet matrices $\mathbf{O}$, $\mathbf{A}$, and $\mathbf{C}^{(0)}$ given a set of hidden variables. For each window size level, we approximate the matrices from their first level. Our method maintains the multi-scale window structure, which allows us to provide the output of an arbitrary window scale.

Algorithm 1 TriMine($\mathcal{X}^{(0)}$)

```
/* compute the triplet matrices at level h = 0 */
for each iteration do
  for each non-zero element x in X^(0) do
    for each entry for x do
      Draw hidden variable z by Equation (1)
    end for
  end for
end for
Compute O, A, C^(0) by Equation (2)
/* compute the multi-scale matrices */
for h = 1 to ⌈log n⌉ do
  Compute C^(h) by Equation (3)
end for
return O, A, {C^(0), ..., C^(h)}
```

5. TRIMINE-F: FORECASTING

We have already presented some of our meaningful features (see Figure 1, ternary plots of $\mathbf{O}, \mathbf{A}$, time-topic sequences $\mathbf{C}$), both visually and numerically, extracted from complex time-stamped events. While clustering, visualization and anomaly detection are provided straightforwardly by our output, we focus on the most challenging problem of *TriMine*, namely, complex event forecasting (Problem 2). We refer to this extension of *TriMine* for forecasting as *TriMine-F*.

Individual-sequence forecasting. Once we decide a time-window length l, we can turn the problem into $u \times v$ forecasting problems, one for each (*actor*, *object*) time sequence. Then we can use any forecasting method. This approach requires at least $O(u\, v)$ space and $O(u\ v\ n)$ time, to predict the future event counts of all sequences. Here, one subtle, but important issue is that most of the (*object*, *actor*) pairs have very sparse sequences, which derails all typical forecasting methods because they look like noise (e.g., $\{0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 1, \dots\}$), and thus are hard to predict. So how can we avoid this issue? We propose using the "*hidden topics*" to achieve much better forecasting. Our proposed *TriMine-F* method can avoid the sparsity problem, because "topics" have non-trivial activities, even if some of the actors (or objects) of these topics have sparse activity.

5.1 Forecasting topic activity

Given the results of *TriMine*, we can forecast the dynamics of activities $\mathbf{C}$ for each topic r ($r = 1, \dots k$), and then translate it to URL (*object*), or user (*actor*) activity, using the participation matrices $\mathbf{O}$ and $\mathbf{A}$, respectively.

Preliminary: single-level equation. If we had a single time granularity (say, window width l_0), then we would do auto regression (AR), and we could forecast the activity $c_{r,t}$ for topic r at time t as a function of the w previous activity levels $c_{r,t-1}, \dots c_{r,t-w}$, plus, (filtered) noise ϵ_t:

$$c_{r,t} = \lambda_1 c_{r,t-1} + \dots + \lambda_w c_{r,t-w} + \epsilon_t, \quad (4)$$

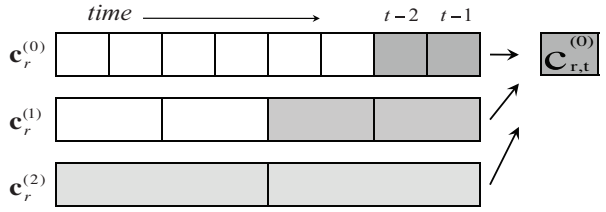


Figure 3: Illustration of multi-scale forecasting (here, $l_0 = 1, l = 2$). The gray cells indicate the variables we use to forecast $c_{r,t}$. Note that we use w variables ($w = 2$ in this example) from each level.

where λ is a regression coefficient.

Multi-scale dynamics. How should we operate multi-scale proportions for event forecasting? Real time-stamped events typically include bursty patterns, noise, spikes, and dips, as well as the event trends of multi-scale periods, all of which are unknown in advance. Long-term patterns are included in the time matrix at the top level while short and sharp fluctuations appear at the first level. Thus, instead of using a single level of window size, we fit a model to the time matrices of multiple levels, (i.e., $\mathbf{C}^{(0)}, \mathbf{C}^{(1)}, \dots$). In a multi-level case, our forecasting method uses $\lceil \log n \rceil$ additional levels (see Figure 3). Thus, we try to fit models of the following form:

$$c_{r,t}^{(0)} = \sum_{h=0}^{\lceil \log n \rceil} \sum_{i=1}^{w} \lambda_{i,r}^{(h)} c_{r,t-i}^{(h)} + \epsilon_t. \quad (5)$$

5.2 Complex event forecasting

Here we answer two questions:

- **Count estimate:** How many clicks $x_{i,j,t}$ should we expect to see from user/*actor* j, to URL/*object* i, at the next time-tick t (e.g., the next hour)?
- **Event-set forecast:** How can we generate a realistic set of event entries of form $\{object, actor\}$ for time-tick t.

Clearly, the second question is more general. if we can answer that, we can easily give the count estimate for the first question (and also the variance etc, since we can generate many event samples).

A few clarifications, first: (a) without loss of generality, we assume that we have agreed on the aggregation length l, and each time-tick is l seconds long. (b) We can easily extend our solutions to long-term forecasting such as *estimate the number of clicks from user 'Smith' to URL 'CNN.com', for the next 30 days*, i.e., $x_{i,j,t}, (t = 1, 2, ..., 30)$. (c) We can easily extend our answers to marginals (or "don't cares"), for example, *estimate the number of clicks from user 'Smith', to* any *URL, for tomorrow*, i.e., estimate $x_{*,j,t}$ where we follow the Unix convention and '*' means "don't care".

Count estimation. The 'count' question can be answered easily by using following steps: (a) forecasting the strength $c_{r,t}$ for every topic $r = 1, \dots, k$ at time-tick t, and (b) using the *participation* matrices $\mathbf{O}$ and $\mathbf{A}$ to translate from topic activity to URL-and-user activity (= clicks). We can then estimate the count of an element $(x_{i,j,t})$ of the i-th *object* and the j-th *actor* at time t:

$$\hat{x}_{i,j,t} = n\bar{x}_i \sum_{r=1}^{k} o_{i,r} \cdot a_{r,j} \cdot \hat{c}_{r,t}, \quad (6)$$

where n is the duration of forecasted events, and $\bar{x}_i$ is the average number of events per time-tick for the i-th *object*.

Complex event generation. For the event-generation problem, the answer is more elaborate, and is illustrated by Algorithm 2. The idea is again to forecast the future activity $\hat{c}_{t,r}$ and then draw multinomial samples using the participation matrices $\mathbf{O}$, $\mathbf{A}$, and $\hat{\mathbf{C}}$. To generate a set of event entries, our sampling-based approach obtains an appropriate number of independent samples of the statistic directly from the underlying generative model. The samples can then be organized into a set of entries of form $\{object, actor, time\}$, simply by gathering all the event entries together.

Algorithm 2 EventGeneration ($\bar{x_1}, \dots, \bar{x_u}, n, \mathbf{O}, \mathbf{A}, \hat{\mathbf{C}}$)

```
/* Ê is a set of generated entries of form {object, actor, time} */
Ê ← ∅
for each object i = 1, ..., u do
  for each entry j = 1, ..., n x̄_i do
    Draw a hidden variable z_{i,j} ~ Multinomial(O_i)
    Draw an actor e ~ Multinomial(A_{z_{i,j}})
    Draw a timestamp t ~ Multinomial(Ĉ_{z_{i,j}})
    Ê = Ê_t ∪ {i, e, t}
  end for
end for
Return Ê
```

6. EXPERIMENTS

To evaluate the effectiveness of *TriMine*, we carried out experiments on real datasets. Our experiments were conducted on an Intel Core 2 Duo 1.86GHz with 4GB of memory, running Linux. The experiments were designed to answer the following questions:

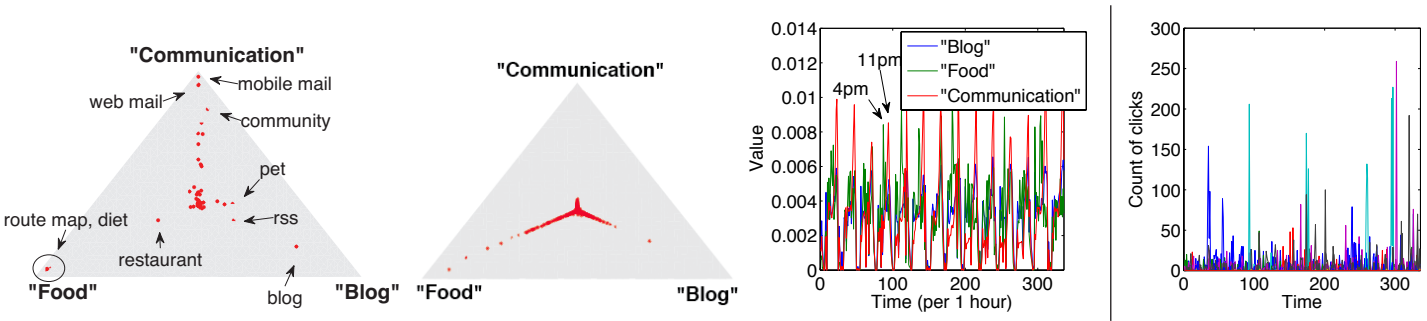
1. Effectiveness: How successful is *TriMine* in capturing time-stamped events, and spotting meaningful patterns?
2. Forecasting accuracy: How well does our method forecast future event entries?
3. Scalability: How does our method scale with the dataset size in terms of computational time?

Datasets. We performed experiments on two real datasets:

- *WebClick*: This dataset consists of web-click records, obtained over one month, (from 1st to 30th Apr. 2007). It contains one billion records, each of which has three attributes: URL ID (1,797 URLs), user ID (10,000 heavy users), and the time stamp of the click. There are various types of URLs, such as "blog", "news", "money", and "diet".
- *Ondemand TV*: This consists of 13,231 TV programs that 100,000 users viewed in a 6-month time frame (from 14th May to 15th Nov. 2007), and there are many genres of TV programs including "cartoons", "sports", "movies", and "music". This dataset contains a list of attributes (e.g., channel ID (*object*), user/viewer ID (*actor*), the date the user watched the program). We selected 100 TV programs from the dataset.

6.1 Effectiveness - patterns and visualization

WebClick **dataset.** For a few users (or URLs), human could eyeball them, and derive meaningful patterns. But, how can we accomplish this automatically for thousands of users? Some of the results obtained with for our method for the *WebClick* dataset have already been presented in Section 1 (see Figure 1), which shows that *TriMine* effectively and efficiently discovers the three-way patterns (i.e., *TriMine*-plot). Figure 4 also shows a *TriMine*-plot on *WebClick* in relation to three different topics, 'communication', 'blog', and 'food'.



(a) URLs in topic space (**O**) (b) Users in topic space (**A**) (c) Time evolution of topics (**C**) (d) Original sequence

Figure 4: ***TriMine*** **finds patterns (see (a-c)), when raw data seem noisy (part (d)).** ***WebClick*** **dataset and its** ***TriMine*****-plot (a-c); number of clicks for 100 random users (d).** ***TriMine*****-plot shows clear groups ('food', 'communication' etc), and clear temporal trends (daily periodicity etc, see (c)). Raw data (d), appears bursty and noisy.**

- *Membership Clusters*: Most *objects* (URLs) fall along the center-to-vertex spokes. The route map and restaurant sites are related as a food topic (see Figure 4 (a)), and the diet site also falls into the same topic. From the figure, we can recognize that the users carefully check out restaurants, the route map in their area, and also the calories of their meals.
- *Temporal Trends*: In Figure 4 (c), we observed that the food-related URLs are visited in the early evening before the users go out. We consider the communication sites (web mail, SNS, etc.) to be used heavily in the late evening for private purposes. Figure 4 (d) shows the original sequence of the "blog" site for randomly selected users. Unlike Figure 4 (c), Figure 4 (d) does not exhibit any daily periodicity or any relationship between users.

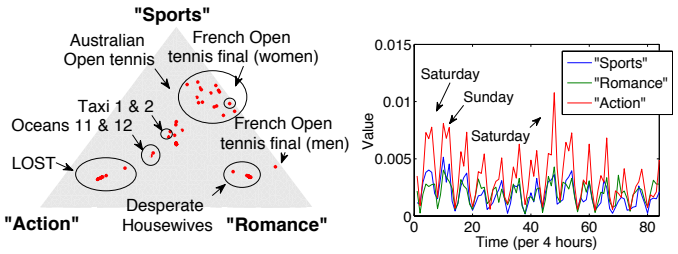
Ondemand TV **dataset.** Figure 5 shows the major topics, 'sports', 'action', and 'romance' (namely, 'soap operas').

- *Membership Outliers*: The ternary plot of the URLs shows clear clustering of the URLs with only one exception, the 2007 French Open men's tennis final. Soap operas and romances (like the series 'Desperate Housewives') probably attract a female audience, who do not seem to be interested in sports, except, apparently, for the men's tennis final of the French Open. We think perhaps that they are interested in one or both of the players in that match (i.e., Rafael Nadal and/or Roger Federer).
- *Temporal Trends*: The time-evolution pattern of the sports topic agrees with our intuition – daily periodicity for all three topics; 'action' and 'sports' show high peaks on weekends, but there is no weekly periodicity for the 'soap opera' topic.

6.2 Accuracy - forecasting

We demonstrate how our forecasting method, *TriMine-F*, works well for time-stamped events, specifically, we evaluate our method using a real dataset, *WebClick*, in terms of long-term forecasting, which is a challenging task. We should note that generating more than, say, 10 steps ahead is very rare: most reported methods [26] generate one step ahead, obtain the correct value of time-tick t, and only then try to generate $t+1$. Nevertheless, our goal is to capture long-term behavior, and we will show that our method achieves this, unlike the alternative methods.

Experimental setup. In our setting, we first trained our models by using the click entries of the first two weeks, and then forecasted the following weeks. We selected the window size $l_0 = 2$, that is,



(a) URL ternary plot (*object*-topic) (b) time evolution of topics

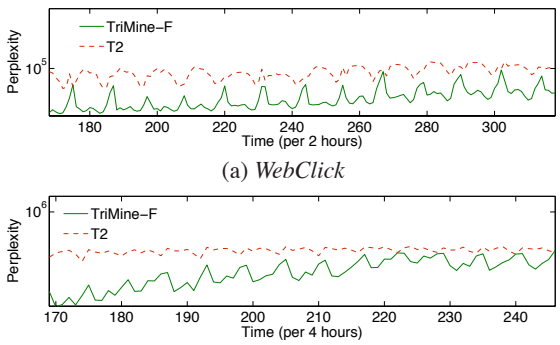
Figure 5: Effectiveness of ***TriMine*** **on the** ***Ondemand TV*** **dataset. Rather clear separation of URLs into 'action', 'sports' and 'romance' (see (a)); clear daily periodicity for all topics (see (b)), with weekly periodicity for 'sports' and 'action', but *not* for 'romance'.**

the length of the training set $\mathcal{X}^{(0)}$ was $n = 168$. We chose the total forecasting coefficients as 40, and the hidden topics $k = 30$.

We compared our algorithm with state-of-the-art methods with respect to forecasting the number of "future" clicks for all possible URL and user combinations. (a) **AR**: This is a straightforward solution to the forecasting problem. We turned the event tensor $\mathcal{X}$ into a set of $u \times v$ sequences for each URL and user. After obtaining the sequence set, we applied standard *AR* modeling for each sequence individually. For a fair comparison, we used 40 regression coefficients. (b) **PLiF**: We compared our algorithm with *PLiF* [13], which is based on Linear Dynamical Systems (also known as Kalman filters). It captures the correlations between multiple sequences, and has the ability to forecast sequences. The original signals are bursty, thus we take their logarithm according to [13]. (c) **T2**: Very recently, Hong et al. present a new topic model for tracking trends [6]. We refer to it as *T2* in this paper. We used the model parameters of *T2*, which can capture the evolution of topics in data collections. In our setting, we inferred the model parameters using entries of duration n. At time-tick n, we stopped the inference, and the latest model parameters were used to predict all of the future events.

6.2.1 Forecast accuracy

Temporal perplexity. We first evaluate the forecasting accuracy of our method in terms of perplexity. Here, we compare *TriMine-F* with *T2*. A lower perplexity indicates a higher predictive accuracy. The perplexities per time-tick are shown in Figure 6. *TriMine-F* captures the general periodic trend, with a desirable slight confu-



(a) *WebClick*

(b) *Ondemand TV*

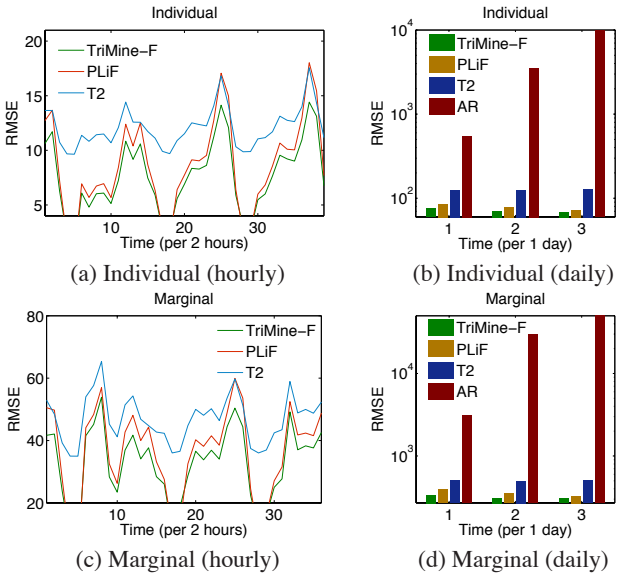
Figure 6: Perplexity at each time-tick. ***TriMine*** **achieves the lower perplexity; we can appropriately forecast the dynamics of hidden patterns.**

sion about the period in both datasets (i.e., *WebClick* and *Ondemand TV*). *T2*, on the other hand, is not intended to capture long-term trends. The figure shows that *T2* fails to predict the future events.

Accuracy of event forecasting. We also compared our algorithm with the standard *AR* method and the state-of-the-art methods, *PLiF* and *T2*, with respect to forecasting a number of "future" events. Figure 7 (a)-(b) shows the root mean square error ($RMSE$) between the original and the forecasted event sets of each URL and each user (we call it "individual forecasts: $x_{i,j,t}$"). Similarly, Figure 7 (c)-(d) describes the results of "marginal" forecasts: $x_{*,j,t}$, that is we forecast the aggregated counts of each user j at every time-tick t. A lower value indicates a better forecasting accuracy. Note that cyclic dips occur in the middle of the night, because we have only a few click entries at this time interval. *T2* was partially successful in generating the future clicks, however, it frequently failed to forecast future event entries. The alternative forecasting methods, *AR* and *PLiF* also failed to forecast the entries, because each sequence was too sparse to capture the cyclic patterns. Unlike the alternative methods, our method achieves high forecasting accuracy for every time-tick. *TriMine-F* outperforms the state-of-the-art methods in terms of forecasting accuracy, and similar trends were observed with the other dataset. We omit the results due to space limitations.

6.2.2 *Benefit of multi-scale approach*

In this subsection, we discuss how *TriMine-F* captures the pattern dynamics. To evaluate the effort from the properties of multiple time scales in our model, we implement a special version of *TriMine-F* by removing this property. Specifically, we use only a single regression coefficient set, (i.e., Equation (4)). We refer to it as *TriMine-F* (single). We used the same coefficients ($= 40$) for a fair comparison. Figure 8 shows the temporal evolution of two major topics for the *WebClick* dataset. We trained our models with click events over a period of two weeks (dotted lines in the figure), and then forecasted the following two weeks. The top, middle and bottom rows of this figure show the output of *TriMine*, *TriMine-F* (single), and *TriMine-F*, respectively. In this figure, *TriMine* does not undertake forecasting, it simply shows the current topic weights at each time. *TriMine-F* is our full solution for event forecasting, which includes multi-scale analysis. The figure shows that our forecasting method, *TriMine-F*, successfully forecasts the subsequent weeks. Specifically, *TriMine-F* (single) failed to capture the dynamics and moved towards convergence. On the other hand, our full solution successfully captured cyclic patterns and periodicity. Consequently, using multi-scale analysis has a significant advantage, which leads to high forecasting accuracy.



(a) Individual (hourly)

(b) Individual (daily)

(c) Marginal (hourly)

(d) Marginal (daily)

Figure 7: Forecasting accuracy for individual $x_{i,j,t}$ (top) and marginal $x_{*,j,t}$ (bottom): ***TriMine-F*** **consistently outperforms the state-of-the-art methods with respect to accuracy (root mean square error ($RMSE$) between real values and forecasts).** *WebClick* **dataset, long-term forecasts starting at t=168 (i.e., after two weeks) (also see Figure 8). Lower is better. We omit the** *AR* **results for hourly error (a) and (c), due to high error values. Notice that** *TriMine-F* **gives the lowest error. Also see the text for more observations.**

6.3 Scalability

We now evaluate the efficiency of *TriMine-F* for event forecasting. Figure 9 compares our method with the AR method in terms of wall clock time when duration n is varied. We use the *WebClick* dataset for this experiment, where the numbers of URLs and users are $u = 1,000$, and $v = 10,000$. The wall clock time is the processing time needed to compute statistics/coefficients and provide the output (i.e., forecasts). Note that *T2* and *PLiF* are based on the Kalman filter and are not scalable for large datasets. These methods need more than 10^6 seconds to compute even at $n = 100$, thus we omit the results. We present our efficient approach for multi-scale analysis in Section 4.2.2. To evaluate the efficiency of this approach, we also show the basic approach, which we call *TriMine-F* (naive). The empirical results in these figures fully substantiate the superiority of *TriMine-F* (i.e., our full solution). We observed that *TriMine-F* achieved a dramatic reduction in computation time that can be up to 7 times faster than *TriMine-F* (naive), and 74 times faster than AR.

7. CONCLUSIONS

We addressed the problem of finding patterns and trends, for "complex" events of the form ($URL, userID, timestamp$), and, in general (*object*, *actor*, time).

We presented *TriMine*, which has all of the following properties:

- Effective: *TriMine* finds meaningful patterns in several real datasets. It thus enables visualization (*TriMine*-plots), anomaly detection, summarization and 'sense-making' (see Figures 1, 4, 5).

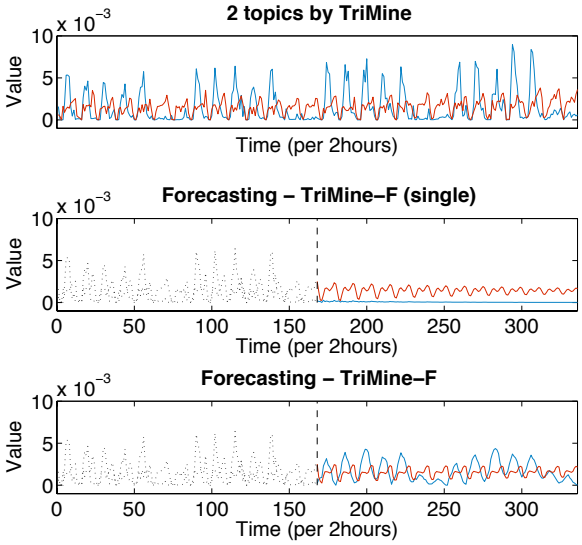


Figure 8: Benefit of multiple time scales. Top: the two main trends of the *WebClick* dataset, namely, 'business' (blue) and 'media' (red). Middle and bottom: *TriMine-F* clearly outperforms *TriMine-F* (single). Both methods start long-term forecasting at t=168 (i.e., after two weeks). Our multi-scale *TriMine-F* reflects reality better, while *TriMine-F* (single) quickly converges to the mean (red), and even worse, predicts near-zero for (blue), blinded by the low activity during weekends.

- Accurate *TriMine* enables forecasting, and specifically, our multi-scale *TriMine-F* is significantly faster and consistently more accurate than state-of-the-art methods.
- Scalable: *TriMine* scales very well, being linear on the database size, and several times faster than its competitors (7x, 74x).

We also demonstrated the practicality of *TriMine*, by applying it to several real datasets, both for pattern discovery and forecasting.

8. REFERENCES

[1] D. Agarwal, B.-C. Chen, and P. Elango. Spatio-temporal models for estimating click-through rate. In *WWW Conference*, pages 21–30, Madrid, Spain, April 2009.

[2] L. AlSumait, D. Barbará, and C. Domeniconi. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*, pages 3–12, 2008.

[3] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *ICML*, pages 113–120, 2006.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[6] L. Hong, D. Yin, J. Guo, and B. D. Davison. Tracking trends: incorporating term volume into temporal topic models. In *KDD*, pages 484–492, 2011.

[7] T. Iwata, S. Watanabe, T. Yamada, and N. Ueda. Topic tracking model for analyzing consumer purchase behavior. In *IJCAI*, pages 1427–1432, 2009.

[8] T. Iwata, T. Yamada, Y. Sakurai, and N. Ueda. Online multiscale dynamic topic models. In *KDD*, pages 663–672, 2010.

[9] R. Kannan, H. Salmasian, and S. Vempala. The spectral method for general mixture models. In *18th Annual Conference on Learning Theory (COLT)*, pages 444–457, 2005.

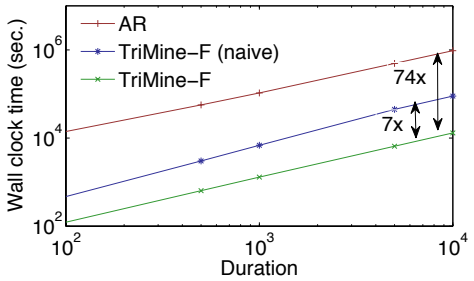


Figure 9: Scalability of event forecasting: wall clock time vs. dataset size (= duration n). Our methods are linear (i.e, slope 1, in log log). *TriMine-F* is 7x times faster than *TriMine-F* (naive), and 74x faster than *AR*. We omit the *T2* and *PLiF* results due to the high computation cost.

[10] T. G. Kolda, B. W. Bader, and J. P. Kenny. Higher-order web link analysis using multilinear algebra. In *ICDM*, pages 242–249, 2005.

[11] F. Korn, S. Muthukrishnan, and Y. Wu. Modeling skew in data streams. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 181–192, New York, NY, USA, 2006. ACM.

[12] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[13] L. Li, B. A. Prakash, and C. Faloutsos. Parsimonious linear fingerprinting for time series. *PVLDB*, 3(1):385–396, 2010.

[14] Y. Matsubara, Y. Sakurai, and M. Yoshikawa. Scalable algorithms for distribution search. In *ICDM*, pages 347–356, 2009.

[15] R. V. Nehme, E. A. Rundensteiner, and E. Bertino. Tagging stream data for rich real-time services. *PVLDB*, 2(1):73–84, 2009.

[16] S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, hands-off stream mining. In *Proceedings of VLDB*, pages 560–571, Berlin, Germany, Sept. 2003.

[17] S. Papadimitriou and P. S. Yu. Optimal multi-scale patterns in time series streams. In *SIGMOD Conference*, pages 647–658, 2006.

[18] I. Porteous, D. Newman, A. T. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *KDD*, pages 569–577, 2008.

[19] S. Rendle, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD*, pages 727–736, 2009.

[20] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *Proceedings of ICDE*, pages 1046–1055, Istanbul, Turkey, April 2007.

[21] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *Proceedings of ACM SIGMOD*, pages 599–610, Baltimore, Maryland, June 2005.

[22] T. Shi, M. Belkin, and B. Yu. Data spectroscopy: learning mixture models using eigenspaces of convolution operators. In *ICML*, pages 936–943, 2008.

[23] G. Tomasi and R. Bro. Parafac and missing values. *Chemometrics and Intelligent Laboratory Systems*, 75(2):163–180, 2005.

[24] X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *KDD*, pages 424–433, 2006.

[25] X. Wei, J. Sun, and X. Wang. Dynamic mixture models for multiple time-series. In *IJCAI*, pages 2909–2914, 2007.

[26] A. Weigend and N. Gershenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1993.

[27] L. Yao, D. M. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD*, pages 937–946, 2009.

[28] Z. Zhang, B. T. Dai, and A. K. H. Tung. Estimating local optimums in em algorithm over gaussian mixture model. In *ICML*, pages 1240–1247, 2008.