Fast Mining and Forecasting of Complex Time-Stamped Events

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

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

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

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

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?

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 O matrix, (b) to each user (in the A matrix), and (c)
to each time-tick (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 time-plot 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 (√) 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).

**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 document collections, such as the dynamic topic model (DTM) [3], which has been proposed for analyzing the time evolution of topics in documents, 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 document collections, such as the dynamic topic model (DTM) [3], which has been proposed for analyzing the time evolution of topics in documents, 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 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 defined 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., \(X \in \mathbb{N}^{n \times v \times u}\), where \(n\) is the duration of events.

**Definition 1 (Complex Event Tensor).** Let \(X\) be a 3rd-order tensor of complex time-stamped events. The element \(x_{i,j,t}\) of \(X\) 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, \(\{\text{‘Smith’, ‘cnn.com’, 13am 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 \(O, A, C\) with the following definitions and properties:

**Definition 2 (Object Matrix \(O\) \((u \times k)\).** Each entry \(x_{i,j}\) 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 \(\sum o_{i,j} = 1\). The definitions of \(A\) and \(C\) are analogous, and omitted for brevity. We shall refer to \(O, A\) and \(C\) as participation matrices, exactly because they show how strongly each actor, object, time stamp participates in topics \(\#1, \#2, \ldots, \#k\), 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 \(M\)-th-order tensor, which is decomposed in \(M\) matrices, \(O, A(1), \ldots, A(3-M-2), C\). 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 \(X\), and forecast future events. Specifically, the sub-problem that we want to solve is as follows:

**Problem 1 (Pattern discovery in complex events).** Given a tensor \(X\) of complex events of (actor, object, time stamp) triplets, find the hidden topics that best summarize the events in \(X\), and the corresponding actor- and object- groupings.

Once we have the main trends from \(X\), we can proceed to solve the forecasting problem:

**Problem 2 (Traffic forecasting).** Given a tensor \(X\) 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 usage in recent years. A main challenge is to understand user behavior and how it evolves over time. The TriFact framework (Fig. 1) naturally applies to this problem, and could be used to create personalized content recommendations for users. The key idea is to collect data from a large number of users and then apply the tensor analysis techniques to extract hidden patterns and trends. This can be used to improve the quality of recommendations, for example, by identifying common interests among users.

**Event forecasting.** Another important application of tensor analysis is event forecasting, where the goal is to predict future events based on historical data. For example, in finance, the TriFact framework can be used to predict stock prices by analyzing transaction data from multiple sources. Similarly, in the field of marketing, the framework can be used to predict sales of a particular product by analyzing customer purchase history.

**Social network analysis.** Social networks are another area where tensor analysis can be applied. For example, the framework can be used to analyze the interactions between users on social media platforms. By analyzing the tensor of interactions, one can identify key influencers, communities, and trends in the network.

**Healthcare applications.** Finally, tensor analysis can also be applied to healthcare applications. For example, the framework can be used to analyze patient records to identify patterns in disease progression or to predict the effectiveness of different treatments. This can be particularly useful in personalized medicine, where treatments can be tailored to individual patients based on their specific characteristics.
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 a 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 on-demand 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 \( O \), \( u \times k \)), to the actors (actor matrix \( A \), \( k \times v \)), and to time-ticks (time matrix \( C \), \( k \times n \)).

- **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 \( \{ C^{(0)}, C^{(1)}, \ldots \} \) (say, hours, days, etc.), insisting on fixed object and actor matrices \( O, A \).

Figure 1(a)-(c) illustrates the \( O, A \) and \( C \) matrices, respectively, for the WebClick dataset. **objects** and **actors** are categorical, and the \( O \) and \( A \) 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 \( O \) takes account of all the objects over the entire time range, which produces an accurate summary of the object-topic relationship. The actor matrix \( A \) shows the frequency of entries over actors for the \( i \)-th topic \((i = 1, \ldots, k)\). The time matrix \( C \) 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 \( C^{(0)} \), then increase the window size \( l_2 \), and repeatedly obtain \( C^{(h)} \) for various sizes. This enables the forecasting method (discussed later in Section 5) to exploit the relationship between \( C^{(h)} \) across different scales.

#### 4.2 Proposed solution - TriMine

Given an \( M \)th-order complex event tensor \( \mathcal{X} \), our goal is to find a meaningful summary with \( k \) hidden topics that best describes \( \mathcal{X} \).
variable for every event entry. The latent variable \( z_{i,j,t} \) is assigned for element \( x_{i,j,t} \) in the following manner:

\[
p(z_{i,j,t} = r | \lambda', O', A', C', \alpha, \beta, \gamma) \propto \sum_r \alpha_{r,i} + \alpha k \sum_r \alpha_{r,j} + \beta v \sum_r \alpha_{r,t} + \gamma n
\]

where \( \alpha_{r,i}, \alpha_{r,j}, \) and \( \alpha_{r,t} \) are the total counts that topic \( r \) is assigned to the \( i \)-th object, \( j \)-th actor, and \( t \)-th time, respectively. Note that the prime (e.g., \( \lambda' \)) indicates that the current datum has been excluded from the sum counts, that is, it indicates the count with the entry of the \( i \)-th object and the \( j \)-th actor at time \( t \) removed. Here, \( \alpha, \beta, \) and \( \gamma \) are the parameters of the Dirichlet priors for \( O, A, \) and \( C \), respectively. After the sampler has burned-in, we can produce estimates of \( \hat{O}, \hat{A}, \) and \( \hat{C} \) as follows:

\[
\hat{a}_{i,r} \propto \frac{a_{i,r} + \alpha}{\sum_i a_{i,r} + \alpha k} \quad \hat{a}_{r,j} \propto \frac{a_{r,j} + \beta}{\sum_j a_{r,j} + \beta v} \quad \hat{c}_{r,t} \propto \frac{a_{r,t} + \gamma}{\sum_t a_{r,t} + \gamma n}.
\]

The time complexity of each sampling iteration is \( O(N) \), where \( N \) is the total number of event entries in \( \lambda' \) (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 \( h = 0, 1, 2, ..., \lfloor \log n \rfloor \), 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 \( L \) 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 \( \{\lambda^{(0)}, \lambda^{(1)}, \ldots\} \) 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, \( O, A, \) and \( C^{(0)} \) with the tensor \( \lambda^{(0)} \) (= \( \lambda' \)). 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 \( O \) and \( A \) for all levels, and (b) compute the time matrix \( C^{(h)} \) by using the set of sampling results for the first level, as follows:

\[
\hat{c}_{r,t}^{(h)} \propto \sum_{i=1}^{l_h} \hat{c}_{r,t-i}^{(0)}
\]

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 \( \lambda^{(0)} \) at the first level, we assign a hidden variable \( z \) according to Equation (1). After the sampling, we compute the triplet matrices \( O, A, \) and \( C^{(0)} \) 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.

### 5. TRIFINE-F: FORECASTING

We have already presented some of our meaningful features (see Figure 1, ternary plots of \( O, A, \) time-topic sequences \( C \)), 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 derrails all typical forecasting methods because they look like noise (e.g., \( \{0, 0, 0, 0, 0, 0, 0, 0, 0, 1, \ldots\} \)), 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 \( C \) for each topic \( r (r = 1, \ldots, k) \), and then translate it to URL (object), or user (actor) activity, using the participation matrices \( O \) and \( A \), respectively.

**Preliminary: single-level equation.** If we had a single time granularity (say, window width \( l_h \)), 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 \( c_{r,t-1}, c_{r,t-2}, \ldots, c_{r,t-w}, \) plus, (filtered) noise \( \epsilon \):

\[
c_{r,t} = \lambda_1 c_{r,t-1} + \cdots + \lambda_w c_{r,t-w} + \epsilon_t,
\]
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., $C^{(0)}$, $C^{(1)}$, ...). In a multi-level case, our forecasting method uses $\log n$ additional levels (see Figure 3). Thus, we try to fit models of the following form:

$$c^{(0)}_{r,t} = \sum_{h=0}^{\log n} \sum_{i=1}^{w} x^{(h)}_{i,r,t} + \epsilon_t. \quad (5)$$

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 $\{object, actor, time\}$ 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 $x_{i,j,t}$ 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 number $c_{r,t}$ for every topic $r = 1, \ldots, k$ at time-tick $t$, and (b) using the participation matrices $O$ and $A$ to translate from topic activity to URL-and-user activity (= clicks). We can then estimate the count of an element $(i,j,t)$ of the $i$th object and the $j$th actor at time $t$:

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

where $n$ is the duration of forecasted events, and $\hat{x}_{t}$ 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 $\hat{c}_{r,t}$ and then draw multinomial samples using the participation matrices $O$, $A$, and $C$. 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 ($\hat{x}_1, \ldots, \hat{x}_u, n, O, A, C$)**

/* $\hat{E}$ is a set of generated entries of form $\{object, actor, time\}$ */

$\hat{E} \leftarrow \emptyset$

for each object $i = 1, \ldots, u$ do

for each entry $j = 1, \ldots, nx_{t}$ do

Draw a hidden variable $z_{i,j} \sim \text{Multinomial}(O_i)$

Draw an actor $e \sim \text{Multinomial}(A_{z_{i,j}})$

Draw a timestamp $t \sim \text{Multinomial}(C_{z_{i,j}})$

$\hat{E}_i \leftarrow \hat{E}_i \cup \{i, e, t\}$

end for

end for

Return $\hat{E}$

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 dataset contains a list of 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 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’.
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, 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 $X$ into a set of $n \times v$ 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-
space limitations. We omit the results due to the art methods in terms of forecasting accuracy, and similar trends accuracy for every time-tick.

like the alternative methods, our method achieves high forecasting each sequence was too sparse to capture the cyclic patterns. Un-

methods, however, it frequently have only a few click entries at this time interval.

Note that cyclic dips occur in the middle of the night, because we time-tick

ure 7 (c)-(d) describes the results of “marginal” forecasts: x_{i,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 pe-

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

Figure 7: Forecasting accuracy for individual x_{i,j,t} (top) and marginal x_{i,j} (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 (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


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 (naive), and 74x faster than AR. We omit the T2 and PLoF results due to the high computation cost.