Information Systems 39 (2014) 277–289

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Detecting and monitoring abrupt emergences and submergences of episodes over data streams

Min Gan *, Honghua Dai

School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia

ARTICLE INFO

Available online 22 May 2012

Keywords:
Stream mining
Frequent episodes
Dynamic changes
Abrupt emerging episodes

ABSTRACT

Existing studies on episode mining mainly concentrate on the discovery of (global) frequent episodes in sequences. However, frequent episodes are not suited for data streams because they do not capture the dynamic nature of the streams. This paper focuses on detecting dynamic changes in frequencies of episodes over time-evolving streams. We propose an efficient method for the online detection of abrupt emerging episodes and abrupt submerging episodes over streams. Experimental results on synthetic data show that the proposed method can effectively detect the defined patterns and meet the strict requirements of stream processing, such as one-pass, real-time update and return of results, plus limited time and space consumption. Experimental results on real data demonstrate that the patterns detected by our method are natural and meaningful. The proposed method has wide applications in stream monitoring and analysis as the discovered patterns indicate dynamic emergences/disappearances of noteworthy events/phenomena hidden in the streams.

1. Introduction

Episodes introduced by Mannila [1] are important patterns for modelling the relative order of occurrences for different types of data elements over a single data sequence. For instance, the order '*A* occurs before *C*' can be represented as a serial episode denoted as $\langle AC \rangle$. Episodes can be used in a variety of sequences and streams in real applications. For example, over a DNA sequence, an episode represents the relative order of positions for different types of nucleotides; over a stream of HTTP requests received by a Web server, an episode represents the order of access to different Web resources; for event streams gathered in sensor networks, an episode represents the relative order of occurrences for different types of events, such as smoke appearances and temperature increases.

Most existing studies on episodes [1,2,5–8,10] concentrate on frequent episode (FE) discovery. Such studies calculate the frequencies of episodes over a whole sequence, and extract the episodes frequently occurring over the whole sequence. FEs, however, are not suited for data streams due to two reasons: (1) it is impractical to consider the frequencies over a whole stream because the stream is normally unbounded and (2) the frequencies of episodes over a data stream may change (increase or decrease) over time during the whole lifetime of the stream. For example, given a sample stream in Fig. 1, we consider changes in frequencies of episodes over the stream. Intuitively, episode $\langle AC \rangle$ frequently recurs during time interval [1,5], while in time interval [6,10] $\langle AC \rangle$ never appears (the frequency decreases sharply) and $\langle XY \rangle$ appears frequently. It is clear that FEs over the whole stream can not capture the dynamic changes.

These changes missed by the FEs may be of crucial significance and interest to stream monitoring and analysis in some real applications. We consider the scenarios in two typical real applications. First, in monitoring a stream of online HTTP requests received by a Web server, a significant increase in the frequencies of particular episodes may indicate the increase of similar users.

* Corresponding author. Tel.: +61 03 92517709.
E-mail address: min.gan.au@gmail.com (M. Gan).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.is.2012.05.009

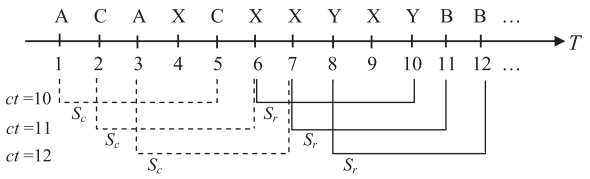

Fig. 1. A sample stream.

The appearances of new episodes may also be a sign of access for a new group of users or abnormal access behaviour such as suspicious intrusions. In this scenario, detection and monitoring of such changes is critical for Web managers to observe any change in the population of user groups and to detect suspicious intrusions. A similar situation exists over streams monitored by a sensor network. Changes detected over streams gathered in a sensor network may indicate that particular events are likely to happen (or have just happened). For instance, in a wireless sensor network for detecting forest fires, an emergence of smoke followed by a sharp increase in temperature may be a sign of a fire. Therefore, it is important to detect the changes in the frequency of episodes over data streams.

In this paper, we focus on detecting and tracing the changes in the frequencies of episodes over data streams. Episodes of significant frequency increases are called 'emerging episodes' (EEs), and episodes of significant frequency decreases are called 'submerging episodes' (SEs). In order to detect the significant changes as early as possible, we only identify abrupt EEs (aEEs) and abrupt SEs (aSEs), i.e., the latest first emergence and submergence of the episodes.

The discovery of aEEs and aSEs faces a number of challenging requirements, such as one-pass of the stream, real-time update and return of results, limited consumption of time and space, and energy saving. This paper aims to propose an aEE–aSE mining method that satisfies the challenging requirements. We choose *T-freq* [2] to measure frequencies of episodes, and propose an efficient mining method by utilising the novel properties of *T-freq*. Our main contributions are summarised as follows.

1. We extend existing studies on frequent episode mining by defining and investigating a new problem, aEE–aSE discovery, which detects and traces dynamic changes in frequencies of episodes over time-evolving streams.
2. An efficient one-pass method is proposed for the discovery of aEEs and aSEs.
3. Through extensive experiments, we evaluate the effectiveness and efficiency of the proposed method, demonstrate the discovered patterns are meaningful and natural, and show its distinct advantages against existing frequent episode mining approaches.

The rest of this paper is organised as follows. Section 2 presents preliminaries, the frequency measure and the problem definition. The mining method is proposed in Section 3. Experimental results are addressed in Section 4. Section 5 reviews related work and the conclusion of this paper is presented in Section 6.

2. Preliminaries, frequency measure and problem definition

Section 2.1 presents preliminaries. Section 2.2 addresses frequency measure, *T-freq* [2], adopted in this paper. The mining problem is defined in Section 2.3.

2.1. Preliminaries

This section presents ordinary terminologies in FE mining [1].

Definition 1 (*Data stream, stream segment*). Let I be a finite set of items, where each item denotes a distinct type of data element. A data stream $\mathcal{S}$ defined over I is an unbounded ordered list of data elements continuously arriving at a rapid rate, denoted as $\mathcal{S}=(e_1)_1(e_2)_2,\ldots$, where each data element is identified by both its type $e_j \in I$ and its timestamp $j \in \{1,2,\ldots\}$. A stream segment is referred to a consecutive substream, e.g., the stream segment arrived so far is denoted as $S=(e_1)_1(e_2)_2,\ldots,(e_{ct})_{ct}$, where ct is the current timestamp.

For example, HTTP requests received by a Web server can be represented as a data stream over I, in which each item represents a Web resource like a Web page or a figure.

Definition 2 (*Sliding window*). Given $S=(e_1)_1(e_2)_2,\ldots,(e_{ct})_{ct}$, a sliding window with width w over S from starting timestamp st, denoted as $win(S,st,w)$, is a stream segment defined as

$$win(S,st,w)=\begin{cases}(e_{st})_{st}(e_{st+1})_{st+1},\ldots,(e_{st+w-1})_{st+w-1} & \text{if } st+w-1\le ct\\ (e_{st})_{st}(e_{st+1})_{st+1},\ldots,(e_{ct})_{ct} & \text{otherwise}\end{cases} \quad (1)$$

Episodes can de divided into three basic classes, serial episodes, parallel episodes and general episodes [1]. In this paper, we only consider serial episodes.

Definition 3 (*Serial episode*). A serial episode α over I is an ordered list of types of data elements from I, denoted as $\alpha=\langle a_1a_2,\ldots,a_m\rangle$, where $a_j \in I$ $(j=1,2,\ldots,m)$. The length of α, denoted as $\alpha \cdot L$, is defined as m.

In the rest of this paper, episodes are referred to as serial episodes. Episode of length m are called length-m episodes. An episode $\alpha=\langle a_1a_2,\ldots,a_m\rangle$ is a sub-episode of another episode $\beta=\langle b_1b_2,\ldots,b_n\rangle$, denoted as $\alpha \sqsubseteq \beta$, if there exists $1\le i_1<i_2<\cdots<i_m\le n$ such that $a_j=b_{i_j}$ for all $j=1,2,\ldots,m$, e.g., $\langle AB\rangle \sqsubseteq \langle ACB\rangle$.

Definition 4 (*Occurrences of episodes*). Given $S=(e_1)_1(e_2)_2,\ldots,(e_{ct})_{ct}$ and $\alpha=\langle a_1a_2,\ldots,a_m\rangle$, we say α occurs in S, and $o=(e_{i_1})_{i_1}(e_{i_2})_{i_2},\ldots,(e_{i_m})_{i_m}$ is an occurrence of α in S, if there exists $1\le i_1<i_2<\cdots<i_m\le ct$ such that $a_j=e_{i_j}$ for all $j=1,2,\ldots,m$. For simplicity, timestamp list $\langle i_1,i_2,\ldots,i_m\rangle$ is used to denote occurrence o. The set of all occurrences of α in S is denoted as $O(S,\alpha)$.