# Anomaly Detection for Discrete Sequences: A Survey

Varun Chandola, Arindam Banerjee, *Member*, *IEEE*, and Vipin Kumar, *Fellow*, *IEEE*

**Abstract**—This survey attempts to provide a comprehensive and structured overview of the existing research for the problem of detecting anomalies in discrete/symbolic sequences. The objective is to provide a global understanding of the sequence anomaly detection problem and how existing techniques relate to each other. The key contribution of this survey is the classification of the existing research into three distinct categories, based on the problem formulation that they are trying to solve. These problem formulations are: 1) identifying anomalous sequences with respect to a database of normal sequences; 2) identifying an anomalous subsequence within a long sequence; and 3) identifying a pattern in a sequence whose frequency of occurrence is anomalous. We show how each of these problem formulations is characteristically distinct from each other and discuss their relevance in various application domains. We review techniques from many disparate and disconnected application domains that address each of these formulations. Within each problem formulation, we group techniques into categories based on the nature of the underlying algorithm. For each category, we provide a basic anomaly detection technique, and show how the existing techniques are variants of the basic technique. This approach shows how different techniques within a category are related or different from each other. Our categorization reveals new variants and combinations that have not been investigated before for anomaly detection. We also provide a discussion of relative strengths and weaknesses of different techniques. We show how techniques developed for one problem formulation can be adapted to solve a different formulation, thereby providing several novel adaptations to solve the different problem formulations. We also highlight the applicability of the techniques that handle discrete sequences to other related areas such as online anomaly detection and time series anomaly detection.

**Index Terms**—Discrete sequences, anomaly detection.

## 1 Introduction

Sequence data are found in a wide variety of application domains such as intrusion detection, bioinformatics, weather prediction, system health management, etc. Anomaly detection for sequence data is an important topic of research as it often leads to detection of actionable and critical information such as intrusions, faults, and system failures. There is extensive work on anomaly detection techniques [1], [2], [3] that look for individual objects that are different from normal objects. These techniques, referred to as *point-based anomaly detection* [1], do not take the sequence structure of the data into consideration. For example, consider the set of user command sequences shown in Table 1. While sequences $S_1$-$S_4$ represent normal daily profiles of a user, the sequence $S_5$ is possibly an attempt to break into a computer by trying different passwords. Though the sequence $S_5$ is anomalous, each command in the sequence by itself is normal.

A sequence is an ordered series of events. Sequences can be of different types, such as binary, discrete, and continuous, depending on the type of events that form the sequences. Discrete and continuous sequences (or time series) are two most important form of sequences encountered in real life applications. In this survey, we will focus on discrete sequences. Discrete or symbolic sequences are ordered sets of events such that the events are symbols belonging to a finite alphabet. For example, a text document is a sequence of words, a computer program is executed as a sequence of system calls, and a gene is a sequence of nucleic acids.

Anomaly detection for discrete sequences has been a focus of many research papers. But most of the anomaly detection techniques have been developed within specific application domains and have been evaluated on specific validation data sets. In particular, several techniques have been developed to detect anomalies in operating system call data [4], [5], [6], [7], [8], [9], [10]. Budalakoti et al. [11], [12] present a study of anomaly detection techniques to detect anomalies in the flight safety domain. Sun et al. [13] present a probabilistic suffix tree-based technique and evaluate it on biological sequences. Note that the nature of the sequence data as well as the nature of anomalies can differ fundamentally across domains, and hence if a technique is shown to be effective within one domain, it does not follow that it will be effective in a different domain.

Even though the existing techniques appear to have the same objective, i.e., to detect anomalies in discrete sequences, a deeper analysis reveals that different techniques actually address different problem formulations. Most of the existing research focuses on one of the following three problem formulations:

1. **Sequence-based anomaly detection**—Detecting anomalous sequences from a database of test sequences.

- *V. Chandola is with the Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, TN 37830. E-mail: chandolav@ornl.gov.*
- *A. Banerjee and V. Kumar are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. E-mail: {banerjee, kumar}@cs.umn.edu.*

*Manuscript received 1 May 2009; revised 10 Dec. 2009; accepted 26 July 2010; published online 18 Nov. 2010.*
*Recommended for acceptance by C. Clifton.*
*For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2009-05-0399.*
*Digital Object Identifier no. 10.1109/TKDE.2010.235.*

TABLE 1
Sequences of User Commands

| | |
|---|---|
| $S_1$ | $login, passwd, mail, ssh, \ldots, mail, web, logout$ |
| $S_2$ | $login, passwd, mail, web, \ldots, web, web, web, logout$ |
| $S_3$ | $login, passwd, mail, ssh, \ldots, mail, web, web, logout$ |
| $S_4$ | $login, passwd, web, mail, ssh, \ldots, web, mail, logout$ |
| $S_5$ | $\mathbf{login}, \mathbf{passwd}, \mathbf{login}, \mathbf{passwd}, \mathbf{login}, \mathbf{passwd}, \ldots, \mathbf{logout}$ |

2. **Contiguous subsequence-based anomaly detection**—Detecting anomalous contiguous subsequences within a long sequence.
3. **Pattern frequency-based anomaly detection**—Detecting patterns in a test sequence with anomalous frequency of occurrence.

These formulations are fundamentally distinct, and hence require exclusive solutions. Even for a single problem formulation, different techniques have been proposed that utilize distinct approaches to detect anomalies. Fig. 1 shows the categorization of existing anomaly detection research for discrete sequences.

To the best of our knowledge, there is no other study that provides a global understanding of the existing research along the lines of Fig. 1. One reason for lack of such a study is because the literature on anomaly detection for sequences is scattered across varied application domains and hence most of the studies have been domain oriented. For example, a comparison of four different anomaly detection techniques for discrete sequences was presented by Forrest et al. [4], but all of the techniques focused on system call intrusion detection. Another reason is that techniques solve different problem formulations. For example, Chandola et al. [14] provided a comparative evaluation of eight anomaly detection techniques on sequence data collected from different domains, but only focused on the sequence-based problem formulation.

### 1.1 Our Contributions

This survey attempts to provide a comprehensive and structured overview of the existing research for the problem of detecting anomalies in discrete/symbolic sequences. The objective is to provide a global understanding of the sequence anomaly detection problem and how techniques proposed for different domains relate to each other.

Our specific contributions are:

- We identify three distinct ways in which the anomaly detection problem has been formulated for discrete sequences and review techniques from many disparate and disconnected application domains that address each of these problem formulations. Such a categorization helps us identify the strengths and limitations of different techniques and could assist researchers in understanding how techniques that address one formulation can be adapted to solve a different formulation.
- Within each problem formulation, we group techniques into categories based on the nature of the underlying algorithm. For each category, we provide a basic anomaly detection technique, and show how the existing techniques are variants of the basic technique. This approach shows how different techniques within a category are related or different from each other. Our categorization reveals new variants and combinations that have not been investigated before for anomaly detection. We also provide a discussion of relative strengths and weaknesses of different techniques.
- We show how techniques developed for one problem formulation can be adapted to solve a different formulation; thereby providing several novel adaptations to solve the different problem formulations.
- We highlight the applicability of the techniques that handle discrete sequences to other related areas such as online anomaly detection and time series anomaly detection.

This survey can be utilized in three ways. First, it provides a well structured overview of the research landscape. Second, it allows researchers in one application domain to understand the existing work done in a different domain. Third, it allows researchers focusing on a particular problem formulation to see how techniques developed for a different formulation can be applied to their problems.

Note that the three formulations covered in this survey do not cover all possible ways in which the anomaly detection problem can be posed for discrete sequences.

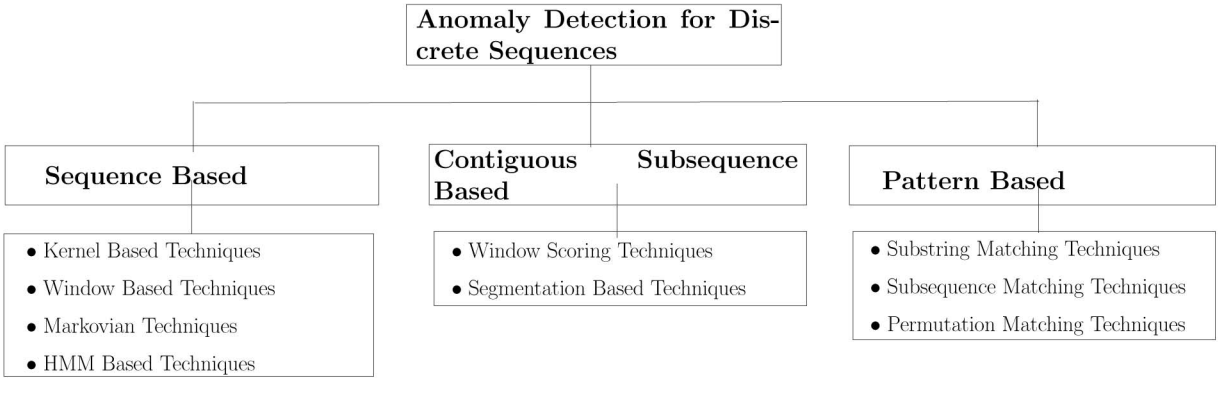


Fig. 1. Overview of existing anomaly detection research for discrete sequences.

However, most of the existing work can be covered under these three formulations.

### 1.2 Terminology

We use the following terminology and notation in this paper:

- A *discrete/symbolic sequence* is defined as a finite sequence of events, such that each event can be represented as a symbol belonging to a finite alphabet. The symbols may belong to a simple alphabet, e.g., events in DNA sequences belong to a four-letter alphabet—$\{A, C, G, T\}$, or to a complex alphabet, e.g., the alphabet for system call sequences (see Table 1) is the set of system calls executed by a particular computer system. For simplicity, we will use sequence to refer to a discrete sequence and symbol to refer to an element of the sequence. Sets of sequences will be denoted using bold capital letters, e.g., $\mathbf{S}, \mathbf{T}$. Individual sequences will be denoted using bold small letters, e.g., $\mathbf{s}, \mathbf{t}$. Length of a sequence $\mathbf{s}$ will be denoted by $l_s$. A sequence within a set will be denoted as $\mathbf{s_i} \in \mathbf{S}, \mathbf{t_j} \in \mathbf{T}$. A symbol within a sequence $\mathbf{s}$ will be denoted as $s_i$.
- A *string* is defined as an ordered sequence of symbols and hence is equivalent to a sequence in this context. We will use the terms strings and sequences interchangeably.
- A *subsequence* of a sequence is defined as a smaller sequence that can be derived from the original sequence by deleting some symbols without changing the order of the remaining symbols [15]. Formally, a sequence $\mathbf{t}(= t_1 t_2 \ldots)$ is a subsequence of $\mathbf{s}(= s_1 s_2 \ldots)$ if there exists a set of integers $(i_1 < i_2 < \ldots)$, such that $s_{i_1} = t_1, s_{i_2} = t_2 \ldots$.
- A *substring* is defined as a contiguous subsequence of a string or a sequence. In some papers, the term *segment* is used instead of a substring. We denote a substring of a sequence $\mathbf{s}$ as $s_{i:j}, i \leq j$ which starts at $i$th location and ends at the $j$th location of the sequence $\mathbf{s}$.

### 1.3 Organization

The rest of this survey paper is organized as follows: Section 2 describes different application domains where anomaly detection techniques for discrete sequences have been developed. Section 3 describes the three different problem formulations. Sections 4, 5, and 6 provide an overview of techniques that have been proposed to solve each one of these problem formulations. Section 7 discusses how the different anomaly detection techniques discussed in the survey are applicable in the context of online anomaly detection. Conclusions and future directions of research are presented in Section 8.

## 2 Applications of Anomaly Detection for Discrete Sequences

A few application domains where anomaly detection techniques have been developed to handle discrete sequences are as follows:

1. *Sequence of operating system calls/user commands* [5], [16], [17], [18]. The sequences are defined using an alphabet of all possible system calls ($\sim$160 for Sun Solaris operating system) or user commands ($\sim$2,500 for a typical UNIX user). Anomalies in such data sets correspond to anomalous program behavior, often caused due to "break-ins" in the computer system by a virus or a malicious user, or unauthorized access of a computer system. Most of the techniques in this domain solve the sequence-based (e.g., identifying anomalous user profiles) or subsequence-based (e.g., online monitoring of system usage) problem formulations.
2. *Biological sequences such as DNA sequences, protein sequences* [13], [15]. The sequences are defined using an alphabet that corresponds to either four nucleic acid bases or close to 20 amino acid bases. Anomalies in such data sets correspond to either mutations, or a condition of disease in the biological organism [19]. Techniques in this domain either solve sequence-based (e.g., identifying anomalous protein sequences [19]) or pattern frequency-based (e.g., identification of mutations) problem formulations.
3. *Sensor data from an operational system (such as aircraft or space shuttle)* [11], [12], [20]. The sequences are collected during the operation of the system through multiple discrete sensors. The alphabet size for such discrete sequences is typically large ($\sim$1,000 for aircraft data [12]). Anomalies in such data sets correspond to fault scenarios in the operation of the system or accidents. Techniques in this domain typically solve the sequence-based problem formulation to identify faulty operational runs.
4. *Sequences of transactions from online banking, customer purchases, and other retail commerce domains* [21]. The "symbols" in such sequences are "actions" by customers and the alphabet size for such sequences can also be large. Anomalies in such data sets correspond to irregular or abnormal behavior by customers. Techniques in this domain typically solve the subsequence-based problem formulation to identify anomalous customer behavior.
5. *Navigational click sequences from websites* [22], [23]. The "symbols" in such sequences correspond to clicked links (websites), or categories to which the links belong. The anomalies in such data correspond to irregular or unauthorized user behavior. Techniques in this domain also solve the subsequence-based problem formulation.

## 3 Problem Formulations

The three formulations of the sequence anomaly detection problem, i.e., sequence-based, contiguous subsequence-based, and pattern frequency-based anomaly detection, are unique in terms of how the anomalies are defined. For the first formulation, an entire sequence is anomalous if it is significantly different from normal sequences. For the second formulation, a contiguous subsequence within a long sequence is anomalous if it is significantly different from other subsequences in the same sequence. For the

third formulation, a given test pattern is anomalous if its frequency of occurrence in a test sequence is significantly different from its expected frequency in a normal sequence.

From the application perspective, the choice of the problem formulation depends on the relevance of anomalies to the requirement of the application domain. For example, consider the following three scenarios that can arise in the domain of operating system intrusion detection:

- *Scenario 1:* **Sequence-based anomaly detection.** A security analyst is interested in detecting "illegal" user sessions on a computer belonging to a corporate network. An illegal user session is caused when an unauthorized person uses the computer with malicious intent. To detect such intrusions, the analyst can use the first formulation, in which the past normal user sessions (sequence of system calls/commands) are used as the training data, and a new user session (a test sequence) is tested for anomalies against these training data.
- *Scenario 2:* **Subsequence-based anomaly detection.** A security analyst is interested in detecting if a user's account was misused (hacked) at some point during a given day. To detect this misuse, the analyst can use the second formulation, in which the user's day long activity is considered as a long sequence, and is tested for any (contiguous) anomalous subsequence.
- *Scenario 3:* **Pattern frequency-based anomaly detection.** A security analyst is interested in determining if the frequency with which a user executed a particular sequence of commands is higher (or lower) than an expected frequency. Going back to the example given in Table 1, the sequence **login, passwd,login,passwd** corresponds to a failed login attempt followed by a successful login attempt. Occurrence of this sequence in a user's daily profile is normal if it occurs occasionally, but is anomalous if it occurs very frequently, since it could correspond to an unauthorized user surreptitiously attempting an entry into the user's computer by trying multiple passwords. To detect such intrusions, the analyst can use the third formulation, in which the sequence of commands is the query pattern, and the frequency of the query pattern in the user sequence for the given day is compared against the expected frequency of the query pattern in the daily sequences for the user in the past, to detect anomalous behavior.

The next three sections discuss various techniques that address these three problem formulations. Most techniques discussed in this survey assign a score to a sequence, contiguous subsequence, or pattern, indicating the magnitude with which the entity is considered to be anomalous or normal by the given technique. This score can be used to construct a ranked list of anomalies, or, with the help of a threshold, converted into a binary label of normal or anomalous. Some techniques directly assign a binary label or normal or anomalous to the individual entities.

## 4 Sequence-Based Anomaly Detection

Two variants of sequence-based anomaly detection formulation exist:

- *Semisupervised anomaly detection*. A reference (or training) database is assumed, containing only normal sequences, and a test sequence is tested against the normal reference database. Formally, this variant can be stated as

**Definition 1.** *Given a set of $n$ sequences, $\mathbf{S} = \{\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_n}\}$, and a sequence $\mathbf{t}$ belonging to a test data set $\mathbf{T}$, assign an anomaly score to $\mathbf{t}$ with respect to the training sequences in $\mathbf{S}$.*

The length of sequences in $\mathbf{S}$ and the length of sequences in $\mathbf{T}$ might not be equal.

- *Unsupervised anomaly detection*. The task is to detect anomalous sequences from an unlabeled database of sequences (*unsupervised anomaly detection*). Formally, this variant can be stated as

**Definition 2.** *Given a set of $n$ sequences, $\mathbf{S} = \{\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_n}\}$, assign an anomaly score to each sequence in $\mathbf{S}$ with respect to the rest of $\mathbf{S}$.*

In this section, we will primarily discuss techniques that handle the semisupervised variant. A semisupervised technique can be adapted to solve the unsupervised problem by treating the entire data set as a training set, and then scoring each sequence with respect to this training set. Such adaptations assume that the given set contains few anomalous sequences, and hence semisupervised technique does not get affected by the presence of anomalies in the training set.

Anomaly detection techniques in this section can be classified based on the unit element of a test sequence that is analyzed by the technique as follows:

- *Similarity-based techniques.* These techniques treat the entire test sequence as a unit element in the analysis, and hence are analogous to point-based anomaly detection techniques. They typically apply a proximity-based point anomaly detection technique by defining an appropriate similarity measure for the sequences.
- *Window-based techniques.* These techniques analyze a short window of symbols—a short substring—within the test sequence at a time. Thus, such techniques treat a substring within the test sequence as a unit element for analysis. These techniques require an additional step in which the anomalous nature of the entire test sequence is determined, based on the analysis on the substrings within the entire sequence.
- *Markovian techniques.* These techniques predict the probability of observing each symbol of the test sequence, using a probabilistic model, and use the per-symbol probabilities to obtain an anomaly score for the test sequence. These techniques analyze each symbol with respect to previous few symbols.
- *Hidden Markov model (HMM)-based techniques.* These techniques transform the input sequences into sequences of hidden states, and then detect anomalies in the transformed sequences.

A detailed discussion of the above four categories of techniques is given in the next four sections.

### 4.1 Similarity-Based Techniques

These techniques compute pairwise similarity between sequences using a specific similarity (or distance) measure and then make use of a traditional proximity-based point-based anomaly detection algorithm [11], [12], [14].

A basic similarity-based technique operates as follows: the similarity of the test sequence, $\mathbf{t}$, to each sequence in the training set, $\mathbf{S}$, is computed. The similarities are then combined, e.g., inverse of average similarity, to obtain an anomaly score for $\mathbf{t}$.

Several variants of the basic technique have been proposed which use different point-based algorithms and different similarity measures to compare a pair of sequences.

#### 4.1.1 Using Different Point-Based Anomaly Detection Algorithms

Two point-based anomaly detection algorithms that have been used for discrete sequences are k-nearest neighbor (kNN) based [24] and clustering based [25]. Chandola et al. [14] proposed a kNN-based technique in which the anomaly score of a test sequence is equal to the dissimilarity[1] to its $k$th nearest neighbor in the training data set $\mathbf{S}$.

Budalakoti et al. [11], [12] proposed a clustering-based technique in which the training sequences are first clustered into a fixed number of clusters using the $k$-medoid algorithm [26]. The anomaly score for a test sequence is then computed as equal to the inverse of its similarity to its closest medoid. Probabilistic clustering techniques, which do not require an explicit similarity matrix to find clusters in the training set, have also been used for anomaly detection. For example, Yang and Wang [27] proposed a technique that uses mixtures of *Probabilistic Suffix Trees* [28] as cluster representations. Other probabilistic techniques such as mixture of HMMs [29], [30] or mixture of *Maximum Entropy* (maxent) models [23] can also be employed in the same manner for clustering-based anomaly detection.

#### 4.1.2 Using Different Similarity Measures

The simplest similarity measure for comparing a pair of discrete sequences is the *Simple Matching Coefficient* (SMC) [31] which counts the number of positions in which the two sequences match as its similarity. While computing the similarity is fast (linear in the length of the sequences), this measure has a drawback that it requires the sequences to be of equal lengths.

Several anomaly detection techniques use the length of the longest common subsequence as a similarity measure since it can compute similarity between two sequences of unequal lengths [11], [12], [14]. This similarity ($nLCS$) between two sequences $\mathbf{s_i}$ and $\mathbf{s_j}$, is computed as

$$nLCS(\mathbf{s_i}, \mathbf{s_j}) = \frac{|LCS(\mathbf{s_i}, \mathbf{s_j})|}{\sqrt{|\mathbf{s_i}||\mathbf{s_j}|}}, \quad (1)$$

where $LCS(\mathbf{s_i}, \mathbf{s_j})$ is the longest common subsequence shared by $\mathbf{s_i}$ and $\mathbf{s_j}$.

The disadvantage of using $nLCS$ is the computational complexity involved, though faster algorithms to compute the $LCS$ have been proposed [32]. A disadvantage of both SMC and $nLCS$ is that they do not incorporate the relation between different pairs of symbols when computing the similarity, i.e., all matches and mismatches are treated equally, though alternative measures such as *edit distance* can be employed to handle this issue.

Other similarity measures that do not require the sequences to be of equal length can also be used instead of $nLCS$. One such measure was proposed by Kumar et al. [33] by converting the sequences into bitmaps and comparing the bitmaps to determine the similarity. Liu et al. [19] used edit distance to determine anomalous protein sequences from a database of sequences corresponding to different organisms.

**Advantages and disadvantages of similarity-based techniques.** The advantage of similarity-based techniques is that one can use any existing or new similarity measure for sequences [15], [18], [34] and any proximity-based point anomaly detection technique [25], [35], and hence can devise a unique anomaly detection which is best suited for a given problem.

A disadvantage of similarity-based techniques is that their performance is highly dependent on the choice of the similarity measure. Similarity measures such as $nLCS$ are able to distinguish between normal and anomalous test sequences only when anomalous sequences are significantly different from the training sequences. If the anomalies are localized in the test sequence, similarity measures such as $nLCS$ may fail to identify them. Another disadvantage of similarity-based techniques is the $\mathcal{O}(n^2)$ complexity involved in computing pairwise similarity between sequences in $\mathbf{S}$.

### 4.2 Window-Based Techniques

Window-based techniques extract fixed-length overlapping windows from a test sequence. Each window is assigned an anomaly score. The anomaly scores of all windows within a test sequence are aggregated to obtain an anomaly score for the entire test sequence. These techniques are particularly useful when the cause of anomaly can be localized to one or more shorter substring within the actual sequence [16]. If the entire sequence is analyzed as a whole, the anomaly signal might not be distinguishable (as in similarity-based techniques) from the inherent variation that exists across sequences. By analyzing a short window at a time, window-based techniques try to localize the cause of anomaly within one or a few windows.

The standard technique to obtain short windows from a sequence is to slide a fixed-length window, one symbol at a time, along the sequence. Let us assume that for a given sequence $\mathbf{s}$, the extracted windows are denoted by $\omega_1, \omega_2 \ldots \omega_t$ and each window $\omega_i$ can also be denoted as $\omega_{i1}\omega_{i2} \ldots \omega_{ik}$.

To further understand the motivation behind the window-based techniques, let us assume that an anomalous test sequence $\mathbf{t}$ contains a substring $\mathbf{t}'$ (of a certain length), which is the actual cause of anomaly. In a sliding window-based technique, if the length of the window is $k$, the anomalous substring $\mathbf{t}'$ will occur (partly or whole) in $|\mathbf{t}'| + k - 1$ windows. Thus, the anomalous sequence can be potentially detected by detecting at least one of such windows. On the other hand, if $|\mathbf{t}'| \ll |\mathbf{t}|$, a similarity-based technique might

1. An inverse function of similarity.

not be able to resolve the difference between anomalous and normal test sequences.

A basic window-based technique operates as follows: in the training phase, $k$-length sliding windows are extracted from all training sequences and the frequency of occurrence of each unique window in the training data set is maintained (as a *normal dictionary*). In the test step, sliding windows of length $k$ are extracted from the test sequence, $\mathbf{t}$. A window $\omega_i$ is assigned a likelihood score $L(\omega_i)$ which is equal to the frequency associated with the window $\omega_i$ in the normal dictionary. A threshold $\lambda$ is used to determine if a particular window $\omega_i$ is anomalous ($L(\omega_i) < \lambda$) or not ($L(\omega_i) \geq \lambda$). The anomaly score of the test sequence is proportional to the number of anomalous windows in the test sequence. The exact expression for the anomaly score of a test sequence, $\mathbf{t}$ is given by

$$A(\mathbf{t}) = \frac{|i : L(\omega_i) < \lambda, 1 \leq i \leq |\mathbf{t}||}{|\mathbf{t}|}. \quad (2)$$

The above-mentioned basic window-based technique has been proposed for operating system call intrusion detection and is termed as threshold-based sequence time delay embedding (t-STIDE) [4], [5].

Several variants of the t-STIDE technique have been proposed, especially for system call intrusion detection [16], [18], [34], [36], [37], [38], [39], [40]. These variants differ from t-STIDE in terms of how they assign an anomaly score to a window, and how they combine the scores to obtain a global anomaly score for the test sequence.

#### 4.2.1 Assigning Anomaly Scores to Windows

In the t-STIDE technique, the anomaly score for each window $\omega_i$ (denoted as $A(\omega_i)$) is equal to the inverse of frequency of occurrence of the window in the normal dictionary. Other techniques have been proposed to assign a different anomaly score to a window. We will discuss three types of such techniques here.

**1. Using lookahead pairs.** Techniques under this category make use of *lookahead pairs* to assign a score to a window, $\omega_i$ [16]. A lookahead pair is defined as $\langle \alpha, \beta \rangle_j$, such that the symbol $\beta$ occurs in the $j$th location after the symbol $\alpha$ in at least one of the windows in the normal dictionary. Since the windows are of length $k$, $j$ lies between 1 and $k-1$. All such lookahead pairs are added to a secondary normal dictionary. During testing, for each window $\omega_i$, all potential lookahead pairs are extracted. For a length $k$ window, there can be $\frac{k(k-1)}{2}$ such potential pairs. The number of potential lookahead pairs that do not exist in the secondary normal dictionary is counted as the number of mismatches. The anomaly score of the window is calculated as the number of mismatches divided by total number of potential mismatches ($= \frac{k(k-1)}{2}$).

**2. Comparing against a normal dictionary.** Techniques under this category construct a normal dictionary (of fixed-length windows) from training sequences and compare each window from the test sequence to the normal dictionary to obtain an anomaly score.

Hofmeyr et al. [5] use *Hamming Distance* (or number of mismatches) between the test window $\omega_i$ and the closest window in the normal dictionary as anomaly score $A(\omega_i)$. In the same paper, another technique to compute $A(\omega_i)$ is presented. $A(\omega_i)$ is 1 if the window $\omega_i$ is not present in the normal dictionary, and 0 if the window is present in the normal dictionary. *Hamming Distance* has also been used in [40], [41]. The latter approach has been adopted by [36], [38], [39], [42].

Lane and Brodley [18], [34], [43] use the following similarity measure to compute similarity $Sim(\omega_i, \vartheta_j)$ between a test window, $\omega_i$ and a window in the normal dictionary, $\vartheta_j$:

$$w(\omega_i, \vartheta_j, l) = \begin{cases} 0, & if\ i = 0, \\ & or\ \omega_{il} \neq \vartheta_{jl}, \\ 1 + w(\omega_i, \vartheta_j, l-1), & if\ \omega_{il} = \vartheta_{jl}, \end{cases} \quad (3)$$

and

$$Sim(\omega_i, \vartheta_j) = \sum_{l=0}^{k} w(\omega_i, \vartheta_j, l), \quad (4)$$

where $k$ is the length of the windows. In other words, if the two windows match at a location $l$ and the two windows also matched at previous location, the total similarity will be incremented more than if the two windows did not match at previous location, in which case the increment will be of 1. Thus, in the above similarity measure, a series of consecutive matching symbols can accumulate a large similarity, while even a single mismatch in the middle can greatly reduce the similarity.

The anomaly score assigned to $\omega_i$ is equal to the inverse of maximum similarity between $\omega_i$ and windows in the normal dictionary. Lane's PhD thesis [44] discusses other variants of the above described similarity measure in the context of anomaly detection.

As mentioned earlier, the dictionaries are typically built for normal behavior. Certain techniques construct dictionaries for only the anomalous behavior [42], [45], [46], [47], [48]. For example, Dasgupta and Nino [42] first build a normal dictionary of windows. They then generate random windows from the given alphabet. Any window that does not match a normal window is treated as anomalous window and added to the anomaly dictionary. A comparative study of using normal and anomaly dictionaries is presented in [49].

**3. Using a classifier.** Instead of using the frequency of occurrence of a window to compute its anomaly score, some techniques use a classifier to assign an anomaly label to each window. The training involves learning a classifier from the set of $k$-length windows obtained from the training data set $\mathbf{S}$. If $\mathbf{S}$ contains both normal and anomalous sequences, one way to obtain labeled windows was proposed in [10]:

- Windows extracted from normal sequences of $\mathbf{S}$ are assigned a normal label.
- If a window extracted from an anomalous sequence of $\mathbf{S}$ occurs in any normal sequence also, it is assigned a normal label, else it is assigned anomalous label.

If $\mathbf{S}$ contains only normal sequences, a random anomaly generation approach is used [9], [39], [42]. All windows

extracted from sequences in $\mathbf{S}$ are assigned a normal label. To obtain anomalous windows, random windows are generated. If the window occurs in $\mathbf{S}$, it is ignored, or else it is assigned an anomalous label and added to the training data set.

After constructing a training data set, a classifier is learned. Different types of classification models have been learned, such as HMM-based [10], neural networks [9], [36], [39], [50], SVM [51], [52], and rule-based classifiers [53]. During testing, the anomaly score for each window $\omega_i$ obtained from test sequence $\mathbf{t}$, $A(\omega_i) = 0$ if the classifier labels it as normal, and $A(\omega_i) = 1$ if the classifier labels it as anomalous.

#### 4.2.2 Obtaining Anomaly Score for Test Sequence

Once all windows in a given test sequence, $\mathbf{t}$, are assigned an anomaly score, the next step is to combine the anomaly scores (a vector of length $|\mathbf{t}| - k + 1$) to obtain an overall anomaly score $A(\mathbf{t})$. One such technique was discussed earlier for the t-STIDE technique. Here, we will describe some other combination techniques that have been proposed in the literature.

Hofmeyr et al. [5] propose two methods to compute the overall anomaly score. The first method computes this as the average of the strength of anomaly signal over all windows in $\mathbf{t}$.

$$A(\mathbf{t}) = \frac{1}{t} \sum_{i=1}^{l_t - k + 1} A(\omega_i), \tag{5}$$

where $l_t$ is the length of $\mathbf{t}$. The second method checks if any window in $\mathbf{t}$ has a mismatch.

$$A(\mathbf{t}) = \begin{cases} 1, & if\ \exists\, i : A(\omega_i) \geq 1, \\ 0, & otherwise. \end{cases} \tag{6}$$

Forrest et al. [4] proposed a technique to obtain $A(S_q)$, known as *locality frame count* (LFC). For every mismatching window $A(\omega_i) = 1$, the technique counts the number of mismatching windows in the previous $n$ windows of $\mathbf{t}$ ($n$ is a user defined parameter). If this number is greater than a threshold, $A(S_q)$ is incremented. The LFC technique considers a sequence highly anomalous only when a significant number of anomalous windows occur close to each other. If the anomalous windows are located far apart across $\mathbf{t}$, the LFC method gives a lower anomaly score to $\mathbf{t}$. A similar technique has been proposed by Gao et al. [10].

The intuition behind the LFC approach is that anomalies in actual anomalous sequences typically occur as temporally clustered anomalous symbols; hence, aggregating the anomaly scores across the entire sequence $\mathbf{t}$ might "wash out" the anomaly signal; the local combination techniques on the other hand would capture such behavior.

A combination technique similar to LFC was proposed in [39] called the *leaky bucket*. They use the vector obtained of anomaly scores for the windows in the test sequence. For each anomalous window, 1 is added to the global anomaly score and for each normal window, 1 is subtracted (the score is never allowed to fall below 0). Thus, consecutive anomalies result in the global score to increase. If at any time the global score goes above a threshold, the entire test sequence is declared to be anomalous. This technique provides the same benefit as LFC technique.

**Advantages and disadvantages of window-based techniques.** A key advantage of window-based techniques over similarity-based techniques is that they are better suited to detect anomalies which are localized in a short region in the test sequence. Constructing normal dictionaries is simple to implement and can be done efficiently using appropriate data structures.

One disadvantage of window-based techniques is that they are highly reliant on the value of $k$ (length of the window). If $k$ is chosen to be very small, most $k$-length windows will have a high probability of occurrence in the training sequences, while if $k$ is chosen to be very large, most $k$-length windows will have a low probability of occurrence in the training sequences. Thus, in either case, the discriminatory power of the $k$-length windows to differentiate between normal and anomalous sequences will be low. Setting an optimal value for $k$ is challenging. Another disadvantage is that storing all unique windows that occur in training sequences and their frequencies can require a large amount of memory.

### 4.3 Markovian Techniques

The third category of techniques that solve sequence-based formulation learn a model from the training sequences. The model is used as an approximation of the "true" distribution that generated the normal data. Typically, the probability of a given sequence $\mathbf{t}(= \langle t_1, t_2, \ldots t_{l_t} \rangle)$ is factorized using the chain rule:

$$P(\mathbf{t}) = \prod_{i=1}^{l_t} P(t_i | t_1 t_2 \ldots t_{i-1}), \tag{7}$$

where $l_t$ is the length of $\mathbf{t}$.

Markovian techniques utilize the *short memory* property of sequences, which is observed across different domains [28]. The short memory property is essentially a higher order Markov condition which states that the conditional probability of occurrence of a symbol $t_i$, given the sequence observed so far can be approximated as

$$P(t_i | t_1 t_2 \ldots t_{i-1}) = P(t_i | t_{i-k} t_{i-k+1} \ldots t_{i-1}), \tag{8}$$

where $k > 1$.

Markovian techniques operate in two phases, training and testing. Training involves learning the parameters of a probabilistic model of the training sequences and testing involves computing the likelihood of the test sequence given the parameters (see (7)).

#### 4.3.1 Fixed Markovian Techniques

Such techniques use a fixed history (of length $k$) to estimate the conditional probability of a symbol in the test sequence.

A basic fixed Markovian technique "learns" the conditional probability of occurrence of a given symbol $t_i$ as

$$P(t_i | t_{i-k} \ldots t_{i-1}) = \frac{f(t_{i-k} \ldots t_{i-1} t_i)}{f(t_{i-k} \ldots t_{i-1})}, \tag{9}$$

where $f(t_{i-k} \ldots t_{i-1} t_i)$ is the frequency of occurrence of the substring $t_{i-k} \ldots t_{i-1} t_i$ in the sequences in $\mathbf{S}$ and

$f(t_{i-k} \dots t_{i-1})$ is the frequency of occurrence of the substring $t_{i-k} \dots t_{i-1}$ in the sequences in $\mathbf{S}$.

Different variants of the basic technique have been proposed. Ye [54] proposed one such technique for $k = 1$. For this special case, the conditional probability of occurrence of given symbol $t_i$ can be written as

$$P(t_i|t_{i-1}) = \frac{f(t_{i-1}t_i)}{f(t_{i-1})}. \quad (10)$$

An issue with the basic fixed Markovian technique is that storing all transition frequencies (see (9)) can potentially require a huge amount of space. For an alphabet set size of $\sigma$, the total number of frequencies required by the basic technique will be $= (\sigma - 1)\sigma^k$.

Michael and Ghosh [6] address this issue by storing the frequencies in an *Extended Finite State Automata* (EFSA) [6]. The EFSA is like a traditional FSA but with frequencies associated with the nodes and the transitions from one node to another. Each node denotes a $k$-length substring. A node has a transition to another node only if the substring for the second node can be obtained from the substring for the first node by removing the first symbol of the first node and appending any symbol at the end of the substring. The number of times the substring for a node occurs in the sequences in $\mathbf{S}$ is stored at the node. The number of times a transition from one node to another is observed in the sequences in $\mathbf{S}$ are stored at the transitions. Only those nodes and transitions that are observed in the training sequences are stored in the EFSA. Thus, the size of EFSA is typically smaller than the total possible size.

During testing, $(k + 1)$ sized windows are extracted from the test sequence $\mathbf{t}$. The first $k$ symbols determine the current state in the EFSA while the last $k$ symbols denote the next state in the EFSA. If a transition between the two states is defined, the conditional probability for the last symbol of the window is computed by using the frequencies stored in the current node and the next node (see (9)). If the transition is not defined, the symbol is ignored. Chandola et al. [14] proposed a variant of the EFSA technique in which if the transition is not defined, the conditional probability of the last symbol of the window is set to 0.

Michael and Ghosh [6] extended the EFSA-based technique to compute the conditional probability for more than one symbol, given the previous $k$ symbols, to obtain the final anomaly score for $\mathbf{t}$.

Another variant of the basic technique was proposed by Marceau [55] which uses *suffix trees*. In this technique, the suffix tree only maintains the $(k + 1)$ length substrings and their $k$-length suffixes that exist in the sequences in $\mathbf{S}$. Thus, this technique learns a traditional FSA. During testing, all $(k + 1)$ length substrings are extracted from $\mathbf{t}$ and fed into the FSA. An anomaly is detected if the FSA reaches a state from where there is no outgoing edge corresponding to the last symbol of the current substring.

#### 4.3.2 Variable Markovian Techniques

An issue with fixed Markovian techniques is that they force each symbol of a test sequence to be conditioned on the previous $k$ symbols (see (9)). Often, the frequency of a $k$-length substring, i.e., $(t_{i-k} \dots t_{i-1})$, may not be sufficiently large to provide a reliable estimate of the conditional probability of a symbol that follows this substring. For example, let us assume that the substring $aabbb$ occurs once across all training sequences and is followed by symbol $b$ in that solitary occurrence. A fixed Markovian technique ($k = 5$) will assign a conditional probability of 1 to $P(b|aabbb)$. But this conditional probability is not reliable, and hence might give undesirable results. Variable Markovian techniques try to address this issue by allowing symbols to be conditioned on a variable length history. For the above example, $P(b|aabbb)$ might be substituted with $P(b|abbb)$ if the context $abbb$ is more optimal to a given pruning criterion, e.g., frequency of $abbb$ is greater than a certain threshold. Models such as *Probabilisitic Suffix Trees* (PSTs) [28] and *Interpolated Markov Models* (IMM) can be used to efficiently compute the variable length conditional probabilities of a symbol.

One such technique was proposed by Sun et al. [13] using PSTs. A PST is a compact tree representation of a variable Markov chain, which uses classical *suffix trees* as its index structure. In a PST, each edge is labeled using a symbol, and each node represents the substring obtained by traversing the path from root to the node, as well as the number of times the substring is observed in the training sequences. Each node also stores the conditional probability of observing each symbol in the alphabet, given the substring represented by the node. The PST is grown (training phase) by scanning the training sequences. The maximum depth of the tree is fixed at $k$, which is a user defined parameter. Several pruning criteria are applied to the PST to ensure that the PST contains only those paths that occur significantly enough number of times in the training sequences. The pruning can be done by applying thresholds to the frequency of a node label, or to the conditional probability of a symbol emanating from a given node. If no pruning is applied, the PST is equivalent to the EFSA technique discussed in Section 4.3.1.

The PST-based technique computes the *likelihood* for the test sequence $\mathbf{t}$ with respect to the PST. Sun et al. [13] investigate two likelihood-based measures, *Odds* measure and *Normalized* measure. The *Normalized* measure is computed as

$$L(\mathbf{t}) = \frac{1}{l_t}\left(\sum_{i=1}^{l_t} \log P(t_i|t_{i-k+1} \dots t_{i-1})\right). \quad (11)$$

The PST provides an efficient data structure to compute the conditional probability, $P(t_i|t_{i-k+1} \dots t_{i-1}) = P(t_i|t_{i-j+1} \dots t_{i-1})$, where $(j \leq k)$, such that $t_{i-j+1} \dots t_{i-1}$ is the longest suffix of $t_{i-k+1} \dots t_{i-1}$ which occurs as a path in the PST.

The *Odds* measure is computed as

$$L(\mathbf{t}) = \frac{\prod_{i=1}^{l_t} P(t_i)}{\prod_{i=1}^{l_t} p(t_i|t_{i-k+1} \dots t_{i-1})}, \quad (12)$$

where $p(t_i)$ is the empirical probability of symbol $t_i$ in $\mathbf{S}$ and $j \leq k$.

Sun et al. [13] have shown that the *Normalized* measure in (11) performs better than the *Odds* measure in (12) for anomaly detection using protein sequences.

#### 4.3.3 Sparse Markovian Techniques

Variable Markovian techniques described above allow a symbol $t_i$ to be analyzed with respect to a history that could be of different lengths for different symbols; but they still choose contiguous and immediately preceding symbols to $t_i \in \mathbf{t}$. Sparse Markovian techniques are more flexible in the sense that they estimate the conditional probability of $t_i$ based on symbols within the previous $k$ symbols, which are not necessarily contiguous or immediately preceding to $t_i$. In other words, the symbols are conditioned on a sparse history. Using the example from Section 4.3.2, if the sequence "aabbb" occurs rarely in the training sequence, the conditional probability $P(b|aabbb)$ can be potentially replaced with $P(b|aXbXb)$ where $X$ can be replaced with any symbol of the alphabet.

One such sparse technique has been proposed by Eskin et al. [17], using *Sparse Markov Transducers* (SMTs) [56]. SMTs, similar to probabilistic suffix trees, estimate a probability distribution conditioned on an input sequence. SMTs generalize probabilistic suffix trees by allowing for wild cards in the conditioning sequences. The proposed technique estimates the probability distribution of an output symbol, conditioned on an input sequence. SMTs are sparse in the sense that they allow some positions in the conditioning input sequence to be ignored by introducing wild cards. In the training phase, a forest of SMTs is learned from the training sequences to account for wild cards at different positions in the paths from the root. The depth of the SMTs is fixed at a user defined depth $k$. The testing phase is exactly like the testing phase for the PST technique described earlier.

Lee et al. [7] proposed a different sparse Markovian technique that uses a classification algorithm (RIPPER [57]) to build sparse models for sequences. In this technique, $k$-length windows are extracted from the training sequences in $\mathbf{S}$. The first $(k-1)$ positions of these windows are treated as $(k-1)$ categorical attributes, and the $k$th position is treated as the target class. RIPPER is used to learn rules from this categorical data set to predict the $k$th symbol given the first $(k-1)$ symbols. During testing, for each symbol $t_i$, the first rule that fires for the vector $t_{i-k+1} \ldots t_{i-1}$ is chosen. If the target of the selected rule is $t_i$, then the probability $P(t_i|t_{i-k+1} \ldots t_{i-1}) = 1$. If the target of the selected rule is not $t_i$, then $P(t_i|t_{i-k+1} \ldots t_{i-1})$ is the inverse of the confidence associated with the selected rule. Since the precedent of a RIPPER rule is an instance of the subset of the $(k-1)$ attributes, the RIPPER-based technique learns a sparse model from the training data.

**Advantages and disadvantages of Markovian techniques.** The key advantage of Markovian techniques is that they analyze each event with respect to its immediate context. Thus, such techniques are able to detect anomalous sequences even if the anomalies are localized within a long sequence. The variable and sparse Markovian techniques provide flexibility in terms of the size of context (or the length of the history, $k$), with respect to which a symbol is analyzed. The advantage of such flexibility can be illustrated using following example. Let us assume that in a database of normal sequences, the probability of observing a particular symbol after any $k$-length substring is significantly low, but the probability of observing that symbol after one or more $\frac{k}{2}$ length substrings is sufficiently high. In that case, the fixed Markovian techniques with history $k$ will assign a high anomaly score to any occurrence of that symbol in a test sequence, thereby increasing the false positive rate. On the other hand, the variable and sparse techniques will condition that symbol with respect to a $\frac{k}{2}$ length history, and hence will assign it a low anomaly score in a test sequence.

The above-mentioned strength of variable and sparse Markovian techniques can also become a disadvantage. For these techniques, the probability of a "truly" anomalous symbol will be boosted since it will be conditioned on a shorter history, whereas the fixed Markovian technique will still assign a low probability to such a symbol. Thus, the variable and sparse techniques might suffer with higher false negative rate. Chandola et al. [14] compare a fixed Markovian technique with a variable and a sparse Markovian technique on data from different application domains and show that the fixed Markovian technique (using EFSA) outperforms the variable (using PST) and the sparse (using RIPPER) techniques on many data sets. The same paper also provides scenarios in which the variable and sparse Markovian techniques can be better than the fixed Markovian techniques.

### 4.4 Hidden Markov Models-Based Techniques

*Hidden Markov Models* are powerful finite state machines that are widely used for sequence modeling [58]. An HMM is parameterized by a hidden state transition matrix and an observation matrix. The three key learning tasks associated with HMMs are: 1) for a given set of observation sequences, learn the most likely HMM parameters which result in maximum likelihood for the observation sequences, 2) for a given HMM, compute the hidden state sequence that is most likely to have generated a given test sequence, and 3) for a given HMM, with given state transition and observation matrices, compute the probability of a given test sequence. Thus, HMMs can be used not only to estimate the probability of a test sequence (task 3) but also to transform an observation sequence into a hidden state sequence (task 2). Both of these capabilities of HMMs can be used to solve the anomaly detection problem for discrete sequences.

One approach to use HMMs for anomaly detection is to first learn an HMM that best describes the normal training sequences (task 1), and then compute the probability of each test sequence using the learned HMM. The negative log of the probability can be used as the anomaly score for the test sequence. The learning step is typically done using the *Baum Welch* algorithm [59], while the probability computation is done using the *forward algorithm*. This approach has been used by Lane [44], [60] to identify anomalous computer usage in system call data corresponding to user behavior. Yamanishi and Maruyama use a similar approach for identifying anomalies in system log data [61], the only difference being that they use a mixture of HMMs to model the normal sequences.

Another approach to using HMMs for anomaly detection is to analyze the most likely or optimal hidden state sequence (task 2), which can be obtained using the *Viterbi*

algorithm [62]. Florez et al. [63] label each state transition in the optimal state sequence as normal or anomalous by applying a threshold on the state transition probability. The number of anomalous state transitions for a test sequence is used as its anomaly score. Alternatively, Forrest et al. [4] compute the average state transition probability for the entire test sequence and use the average value to compute the anomaly score for the test sequence.

The computation of the optimal hidden state sequence from an observation sequence can also be viewed as a data transformation step. Once all training and test observation sequences are converted to corresponding state sequences, any sequence anomaly detection technique that has been discussed in previous sections can be applied to estimate the anomaly score for the test sequence. Qiao et al. [64] apply a window-based technique in the hidden state space. An extension to the latter technique was proposed by Zhang et al. [65], in which the state transition sequences from the HMM are used to train another HMM. The state transition sequences from the second HMM are then used as input to a window-based anomaly detection technique.

**Advantages and disadvantages of HMM-based techniques.** Key strength of HMM-based techniques is that they can model complex systems. Even when the normal observation sequences appear significantly different from each other, the corresponding hidden state sequences will be, in principle, more similar to each other and different from the anomalous sequences. Forrest et al. [4] compared HMMs against window-based and Markovian techniques for system call sequences and concluded that the performance of HMM was comparable, and sometimes better than the other techniques.

The key assumption for HMM-based techniques is that the normal sequences are generated from a probabilistic model, i.e., the HMM. If this assumption does not hold true or the parameters are not estimated accurately (due to suboptimal initializing conditions), the HMM-based technique will not be able to effectively distinguish between normal and anomalous sequences. The computational complexity associated with HMM-based learning is another drawback of such techniques, as noted by Forrest et al. [4].

## 5 Contiguous Subsequence-Based Anomaly Detection

Techniques under this category solve the following anomaly detection problem:

**Definition 3.** *Detect short contiguous subsequences in a long sequence* $\mathbf{t}$*, that are anomalous with respect to rest of* $\mathbf{t}$.

Techniques in this category try to identify anomalous contiguous subsequences (or substrings) of a long sequence, though the formulation can also be applied to detecting noncontiguous subsequences. The anomalous contiguous subsequences are also defined as *discords* by Keogh et al. [66]: *discords are subsequences of a longer time series that are maximally different from the rest of the time series subsequences*. Note that, many papers in this category [66] use the term "subsequence" (which can have gaps) even though they primarily deal with contiguous subsequences (or substrings). To avoid confusion, we will use the term discord to denote "contiguous subsequences" even though the original papers may use the term subsequence only.

The contiguous subsequence-based problem formulation is natural for several application domains, where an activity is being monitored over a long period of time. For example, in credit card fraud detection, an individual's credit card transactions are continuously monitored, and a discord, i.e., an anomalous sequence of actions (purchases), may indicate a case of identify theft/misuse.

A basic anomaly detection technique can be described as follows: first, all $k$-length windows are extracted from the given sequence $\mathbf{t}$ and stored as a database of fixed-length windows, denoted as $\mathcal{T}_k$. Each window is assigned an anomaly score by comparing it with rest of the windows in $\mathcal{T}_k$, e.g., computing average distance of the window to the windows in $\mathcal{T}_k$. The windows with anomaly scores above a user defined threshold are chosen as the top discords. Since the length of the "true" discord is not known a priori, it is generally assumed that the discords are contained within a contiguous subsequence (or a window) of fixed length $k$.

This basic technique is at the heart of a number of techniques investigated by Keogh et al. [66], [67], [68]. Note that these techniques were originally presented in the context of time series data, but can be extended to discrete sequences by discretizing the time series using techniques such as *Symbolic ApproXimation* (SAX) [69].

An issue with the basic technique is that when any window is compared with the other windows in $\mathcal{T}_k$, the windows which overlap with the given window in $\mathbf{t}$, will be highly "similar" to the given window. For example, any window will differ in at most one position with the immediately next window in the given sequence. This property will be exhibited by both normal windows as well as windows that contain discords, and can bias the computation of anomaly score for a window. Keogh et al. [67] propose a simple solution (referred to as *nonself match* in the paper) to this issue by comparing a window with only the windows in $\mathcal{T}_k$ that do not overlap with the given window.

Several variants of the basic techniques have been proposed, and can be broadly grouped into two categories. The first category of techniques scores the windows differently, while the second category of techniques addresses the time complexity of the basic technique.

### 5.1 Window Scoring Techniques

One possible technique for scoring the windows is to count the number of times a window occurs in the database of all windows, $\mathcal{T}_k$ (this is similar to the window-based techniques such as t-STIDE, discussed in Section 4.2). The anomaly score of the window will be the inverse of this count. While for smaller values of $k$ this is a reasonable technique, it might not be possible to find exact matches of the window in $\mathcal{T}_k$ when the value of $k$ is large.

In another possible technique, the anomaly score of a window is calculated as equal to its distance to its $m$th nearest neighbor in $\mathcal{T}_k$. One such variant, called HOTSAX [67], was proposed by Keogh et al., in which $m = 1$, i.e., the anomaly score of a window is equal to its distance to its nearest neighbor in $\mathcal{T}_k$ (only the nonself

matches are considered). One drawback of the nearest neighbor-based technique is that they involve an additional parameter, $m$, which needs to be set carefully, though approaches such as using the weighted sum of distance to the $m$ nearest neighbors to compute the anomaly score can be used to reduce the sensitivity on $m$.

Another variation of the basic anomaly detection technique, known as *Window Comparison Anomaly Detection* (WCAD), was proposed by Keogh et al. [70]. To assign an anomaly score to each window in the sequence $\mathbf{t}$, the window is compared against rest of the sequence (say $\mathbf{t}'$) using an information theoretic measure called *Compression-Based Dissimilarity* (CDM). For a window $\omega_i \in \mathcal{T}_k$, the CDM is defined as

$$CDM(\omega_i, \mathbf{t}') = \frac{\mathcal{C}(\omega_i \mathbf{t}')}{\mathcal{C}(\omega_i) + \mathcal{C}(\mathbf{t}')}, \tag{13}$$

where $\mathcal{C}(x)$ is the compression attained by any standard compression algorithm on a given string $x$. The intuition behind using the measure defined in (13) is that if the window $\omega_i$ is normal, it will match the rest of the sequence very well and hence will not require extra space when both $\omega_i$ and $\mathbf{t}'$ are compressed together. On the other hand, if $\omega_i$ is a discord, it will not match the rest of the sequence and hence the value of $\mathcal{C}(\omega_i \mathbf{t}')$, and hence the anomaly score, will be high.

Wei et al. [71] proposed a variation of the basic technique by sliding two adjacent windows along the sequence. The anomaly score of each window is computed by comparing its bitmap with the bitmap of the previously adjacent window. The length of the preceding window is not required to be same as the current window. The underlying assumption for this technique is that a normal window will be similar to the previously adjacent window, while a window containing a discord will be significantly different.

Several of the above-mentioned variants measure similarity/distance between a pair of windows. A number of similarity/distance measures such as *Simple Matching Coefficient*, *edit distance*, *length of longest common subsequence*, *weighted SMC*, *chaos bitmaps* [71] can be used.

### 5.2 Optimizing the Complexity of the Basic Technique

An issue with the basic technique that solves contiguous subsequence-based formulation is that it requires $O(l_t^2)$ comparisons of window pairs, where $l_t$ is the length of the sequence $\mathbf{t}$. Several faster variations have been proposed that can run in approximately linear time in the context of continuous time series [67], [68], [72], [73], [74], and can be extended to discrete sequences.

One general technique for reducing complexity of the basic technique makes use of the following fact. Instead of scoring all windows, they score as many windows as required to get the top $p$ anomalous windows. Thus, a window can be pruned if at anytime its distance to its current $m$th nearest neighbor is lower than the anomaly score of the current window with $p$th largest anomaly score. Since the distance of the window to its actual $m$th nearest neighbor is upper bounded by the current distance, this window will never figure in the top $p$ anomalous windows, and hence can be pruned.

Since a vast majority of windows tend to be normal, the technique has the potential for substantial pruning, especially if the anomalous windows are discovered early.

Such pruning has been successfully used for traditional anomaly detection [75], and has been applied to discord detection [67], [68], [72]. It should be noted that this pruning method guarantees the same result as the basic technique, but can result in lower execution time.

### 5.3 Segmentation Techniques

Choosing an optimal value of $k$ is challenging. If $k$ is set to be very small, all $k$-length windows might appear highly probable, resulting in high false negative rates. If $k$ is set to be very large, all $k$-length windows might have a low occurrence probabilities, resulting high false positive rates. This challenge becomes more significant if the sequence contains multiple discords, of varying length. In this case, a single value of $k$ might not be enough to detect all discords.

One approach to address this challenge is to extract unequal length segments (or substrings) from the given sequence $\mathbf{t}$, as proposed by Chakrabarti et al. [21]. Originally, this segmentation-based technique was proposed for a sequence of market baskets, but this can be easily generalized to a discrete sequence, by encoding the alphabets as bit vectors, and hence treating them as market baskets. In this approach, $T$ is segmented into unequal length segments, such that the sum of the number of bits required to encode each segment (using *Shannon's Source Coding Theorem*) is minimized. The segments which require highest number of bits for encoding are treated as discords. It is shown that an optimal $O(l_t^2)$ solution exists to find such segmentation when the number of items is 1, i.e., for a binary sequence. Approximate algorithms are proposed to handle the case when the number of items is more than 1, though the technique is limited to small item set sizes, or small alphabets. Gwadera et al. [76] proposed a variable Markov chain (similar to the variable Markovian model discussed in Section 4.3.2) based segmentation technique for discrete sequences which can also be employed in a similar manner to identify discords.

### 5.4 Relationship between Sequence-Based and Contiguous Contiguous Subsequence-Based Anomaly Detection Techniques

Techniques that handle sequence-based formulation and the techniques that handle contiguous subsequence-based formulation have been developed independently since they solve distinct problems. Here, we will discuss how techniques belonging to the first category can be used to solve the second problem, and vice versa.

#### 5.4.1 Using Sequence-Based Anomaly Detection Techniques to Detect Contiguous Contiguous Subsequence-Based Anomalies

The basic contiguous subsequence-based anomaly detection technique assigns an anomaly score to a given window $\omega_i$ with respect to a database of windows $T_k$ (a majority of which are assumed to be normal). Any sequence-based anomaly detection technique, described in Section 4, can be

applied here by considering $T_k - \omega_i$ as the training data set $\mathbf{S}$ and the given window $\omega_i$ as a test sequence $S_q$. For Markovian techniques, such as EFSA and PST, extracting windows from $T$ is not explicitly required, since a model can be constructed using the single long sequence $T$, and then a window can be tested against the model to obtain an anomaly score.

The key issue with using techniques, such as EFSA and PST, is that a model has to be constructed for every window $\omega_i$, using the data set $T_k - \omega_i$. A simple solution to this issue is to construct a single model using the entire sequence $T_k$, and then score each window against this model, but the model will get biased from the discords that exist in the original sequence $T$. However, this bias will be small as long as the size of the discord is small compared to the entire sequence. Unsupervised techniques such as clustering and k-nearest neighbor can also be applied to assign an anomaly score to each window, though the *self* match issue, discussed earlier, needs to be explicitly handled.

#### 5.4.2 Using Contiguous Contiguous Subsequence-Based Anomaly Detection Techniques to Detect Sequence-Based Anomalies

If all training sequences in $\mathbf{S}$ and the given test sequence $S_q$ in sequence-based formulation are arranged linearly, in no particular order, a long sequence can be constructed. An anomaly detection technique discussed to handle contiguous contiguous subsequence-based formulation can be applied to detect all anomalous fixed-length windows. The anomaly score of the test sequence $S_q$ can be assigned as equal to the number of windows of $S_q$ that are detected as discords. The key issue with this adaptation is that the entire test sequence will be compared (as a window) with other training sequences, and hence will face same challenges as the similarity-based techniques discussed in Section 4.1. Another issue with this approach is that the windows which span across two sequences are an artifact of the concatenation and might affect the performance of the technique.

## 6 Pattern Frequency-Based Anomaly Detection

Techniques under this category solve the following anomaly detection problem:

**Definition 4.** *Given a short query pattern $\alpha$, a long test sequence $\mathbf{t}$, and a training set of long sequences $\mathbf{S}$, determine if the frequency of occurrence of $\alpha$ in $\mathbf{t}$ is anomalous with respect to frequency of occurrence of $\alpha$ in $\mathbf{S}$.*

This problem has also been referred to as *surprise detection* in the context of time series data [69], [77]. Keogh et al. [77] define a pattern to be surprising "*if the frequency of the pattern differs substantially from that expected by chance, given some previously seen data.*" Note that patterns with both greater or smaller frequencies are of interest.

The pattern frequency-based formulation is motivated from Scenario 3 discussed in Section 3 where a pattern is anomalous if its frequency in the given sequence is significantly different from its expected frequency in normal sequences. This formulation is also related to *case versus control* analysis [78], used in epidemiological studies. The idea is to detect patterns whose frequency of occurrence in a given test data set (case) is different from its occurrence in a normal data set (control). Such techniques aim to prioritize the patterns ($\alpha$) occurring in a known anomalous sequence ($\mathbf{t}$) based on their frequency in the sequences in $\mathbf{S}$.

### 6.1 A Basic Technique to Solve Pattern Frequency-Based Formulation

A basic technique to solve the above problem assigns an anomaly score to the query pattern, $\alpha$, as the difference between the frequency of occurrence of $\alpha$ in the sequence $\mathbf{t}$ and the average frequency of occurrence of $\alpha$ in the sequences in set $\mathbf{S}$.

More formally, the following two quantities may be defined, $\bar{f}_{\mathbf{t}}(\alpha)$ and $\bar{f}_{\mathbf{S}}(\alpha)$:

$\bar{f}_{\mathbf{t}}(\alpha)$, also called the *relative frequency* of the query pattern, is the frequency with which the query pattern occurs in the test sequence $\mathbf{t}$, normalized by the length of $\mathbf{t}$, and can be directly computed as

$$\bar{f}_{\mathbf{t}}(\alpha) = \frac{f_{\mathbf{t}}(\alpha)}{l_t}, \tag{14}$$

where $f_{\mathbf{t}}(\alpha)$ is the frequency of the query pattern in the test sequence. $\bar{f}_{\mathbf{S}}(\alpha)$ is the average frequency of the query pattern to occur in a sequence in $\mathbf{S}$ normalized by the length of the sequence. $\bar{f}_{\mathbf{S}}(\alpha)$ can be estimated as

$$\bar{f}_{\mathbf{S}}(\alpha) = \frac{1}{|\mathbf{S}|} \sum_{\forall \mathbf{s_i} \in \mathbf{S}} \frac{f_{\mathbf{s_i}}(\alpha)}{|l_{s_i}|}. \tag{15}$$

The anomaly score of the query pattern $\alpha$ is computed as

$$A(\alpha) = |\bar{f}_S(\alpha) - \bar{f}_{\mathbf{S}}(\alpha)|. \tag{16}$$

### 6.2 Variations of the Basic Technique

In the basic technique, a query pattern $\alpha$ is considered to "occur" in a sequence if $\alpha$ is a *substring* of the sequence. An issue with considering substrings is that it forces the query pattern to occur exactly. If $\alpha$ is long, it is unlikely for entire $\alpha$ to occur in a training sequence. Another issue is that in many domains, it is reasonable to assume that the symbols of the query pattern can occur interspersed with other symbols, and hence only considering substring matches will miss such occurrences. To address these issues, following three variations of the basic technique have been proposed:

- *Counting the number of times a*substring*of the query pattern $\alpha$ occurs in a sequence*. Keogh et al. [77] find the largest $l$ in the interval $[1, l_\alpha)$, such that every $l$ length substring of $\alpha$ occurs at least once in the training sequences.

  The frequency $f_{\mathbf{S}}(\alpha)$ from (15) is then replaced with the following quantity:

  $$f_{\mathbf{S}}(\alpha) \Rightarrow \frac{\prod_{j=1}^{m-l} f_{\mathbf{S}}(\alpha_{j:j+l})}{\prod_{j=2}^{m-l} f_{\mathbf{S}}(\alpha_{j:j+l-1})},$$

  where $\alpha_{j:k}$ denotes the substring of $\alpha$ starting at $j$th location and ending at the $k$th location.

The idea is to estimate the occurrence of the query pattern $\alpha$ using a set of substrings of $\alpha$. By searching for a set in which all substrings occur at least once, the frequency estimation is more reliable.

- *Counting the number of times the query pattern $\alpha$ occurs as a noncontiguous subsequence in a sequence*. An issue with the basic technique is that counting the number of times $\alpha$ occurs as a subsequence in a long sequence is expensive. To make the formulation more tractable, Gwadera et al. [79] extract all windows of a fixed length (greater than the length of $\alpha$), from the sequence and then determine all those windows that contain $\alpha$ as a subsequence. Thus, a query pattern $\alpha$ is considered to "occur" in a sequence if there exists a window such that $\alpha$ is a subsequence of the given window. The number of fixed-length windows which contain $\alpha$ as a subsequence is counted. These counts are used as $f_{\mathbf{t}}(\alpha)$ and $f_{\mathbf{s_i}}(\alpha)$ in (14) and (15), respectively.
- *Counting the number of times any permutation of the query pattern $\alpha$ occurs as a noncontiguous subsequence in a sequence*. This alternative was proposed by Gwadera et al. [80] as an extension to the subsequence matching technique [79]. For the permutation approach, the number of fixed-length windows which contain any permutation of $\alpha$ is counted. The motivation behind considering all permutations is that in certain scenarios, the ordering of events (or symbols) within the query pattern $\alpha$ does not matter.

### 6.3 Issues with the Basic Technique and Techniques for Addressing Them

The basic technique and its variations have following two key issues:

#### 6.3.1 Computational Complexity

For each query pattern, the time required to compute its anomaly score is linear in the length of $\mathbf{t}$ and the length and number of sequences in $\mathbf{S}$. If there are multiple query patterns, e.g., all short contiguous subsequences that occur in $\mathbf{t}$, the total time required to score all query patterns adds up to a high value. To address this issue, Keogh et al. proposed a technique called TARZAN [69], [77] which uses *suffix trees* to efficiently compute the frequency of occurrence of a query pattern in a given sequence. Suffix trees are created for $\mathbf{t}$ and for each sequence in $\mathbf{S}$.[2] Only two suffix trees are required, one for the sequences in $\mathbf{S}$ and for the sequence $\mathbf{t}$, and can be constructed with complexity linear in the length of the sequences. The counts for a query pattern $\alpha$, can be obtained with complexity linear in the length of $\alpha$. Gwadera et al. [81] use *Interpolated Markov Models* to efficiently find the number of windows extracted from a sequence that contain the query pattern.

#### 6.3.2 Scoring of Anomalies

The basic technique assigns an anomaly score to the query pattern (15) but does not declare if the query pattern is anomalous or not. For the TARZAN technique, since there are multiple query patterns to be scored, a relative ordering can be obtained using the anomaly scores, and top few patterns with highest scores could be declared as anomalous. But if there is only one query pattern, a method is needed for declaring the query pattern to be anomalous or normal, based on its anomaly score. Gwadera et al. [79] address this issue by assuming that the relative frequency, $\bar{f}_{\mathbf{t}}(\alpha)$, is generated from a normal distribution $\sim\mathcal{N}(E[\bar{f}_{\mathbf{S}}(\alpha)], Var[\bar{f}_{\mathbf{S}}(\alpha)])$. The expected value or mean, $E[\bar{f}_{\mathbf{S}}(\alpha)]$, and the variance, $Var[\bar{f}_{\mathbf{S}}(\alpha)]$, of the distribution are estimated using sequences in $\mathbf{S}$. The anomaly score for the query pattern is computed as the z-score of the observed relative frequency:

$$zscore = \frac{\bar{f}_{\mathbf{t}}(\alpha) - E[\bar{f}_{\mathbf{S}}(\alpha)]}{\sqrt{Var[\bar{f}_{\mathbf{S}}(\alpha)]}}. \tag{17}$$

A threshold on the $zscore$ is used to determine if the occurrence of $\alpha$ is anomalous or not.

### 6.4 Relationship between Pattern Frequency-Based and Sequence-Based Anomaly Detection Techniques

Pattern frequency-based anomaly detection techniques assign an anomaly score to a short pattern $\alpha$, while the basic window-based sequence anomaly detection technique (see Section 4.2) assigns an anomaly score to each short window belonging to the test sequence. Both techniques appear to be similar in this step, but they are actually quite distinct. While the basic technique discussed in this section assigns an anomaly score to a pattern based on its frequency in the test sequence and its average frequency in normal sequences, the basic window-based technique assigns an anomaly score to a window based on its similarity to the windows in the normal sequences. To be more precise, the basic window-based technique for sequence-based anomaly detection uses $\bar{f}_{\mathbf{S}}(\alpha)$ as the anomaly score of each window, while the basic technique for pattern frequency-based anomaly detection compares this value with the relative frequency of occurrence of the window in the test sequence to compute its anomaly score (see (16)).

Nevertheless, techniques for pattern frequency-based formulation can also be used to find certain types of anomalies in the context of sequence-based anomaly detection. An alternate cause of an anomaly could be that a test sequence is anomalous because it contains one or more patterns whose frequency of occurrence in the test sequence is significantly different from their frequency of occurrence in the training sequences. Such anomalous sequences can be detected as follows: for the given test sequence, fixed-length windows are extracted. Each window is assigned an anomaly score using the basic technique that deals with the pattern frequency-based formulation. The anomaly scores of all windows are aggregated to obtain an overall anomaly score for the test sequence.

A direction adaptation of contiguous subsequence-based anomaly detection techniques to solve the pattern frequency-based formulation, and vice versa, is not feasible, since contiguous subsequence-based formulation deals with only one sequence while pattern frequency-based

2. TARZAN was originally proposed to handle the case when $\mathbf{S}$ contains only one sequence.

formulation requires a training set of sequences, a test sequence, and a query pattern.

## 7 Online Anomaly Detection

Several application domains collect sequence data in a streaming fashion [82], e.g., system call data collected by a computer system, data generated by an aircraft during its flight, etc. Such domains often require the anomalies to be detected in such sequences in an online fashion, i.e., as soon as they occur [83], [84]. Online anomaly detection has the advantage that it can allow analysts to undertake preventive or corrective measures as soon as the anomaly is manifested in the sequence data.

For example, in aircraft health monitoring, the current flight sequence for an aircraft is tested if it is anomalous or not, with respect to a database of historical normal flight sequences of the aircraft. Determining that the current flight sequence has an anomaly, as soon as it occurs (even before the entire flight sequence is collected) might help the health monitoring system to raise an early alarm.

Among the different categories of techniques that handle the sequence-based formulation, some can be easily adapted to operate in an online fashion. The window-based and Markovian techniques assign anomaly score to windows (or symbols) as they are collected. By applying a threshold on the anomaly score, the sequence can be declared to be anomalous even before observing the entire sequence. Hence, such techniques can be easily adapted to operate in an online fashion. For example, Larrahondo et al. [85] have proposed an online HMM-based anomaly detection technique which can compute the optimal state sequence for a given observation sequence in an online fashion by modifying the forward backward algorithm. In contrast, similarity-based techniques measure the similarity of entire test sequence with training sequences, and hence are not suitable for online detection problem.

Techniques for the subsequence-based formulation can also be easily adapted to detect anomalies in an online fashion. In the online setting, each successive subsequence of symbols could be assigned an anomaly score with respect to the sequence observed so far. A threshold on the score computed for each window can be used to declare it to be normal or anomalous, as soon as it is observed. To obtain reliable anomaly scores, such techniques will require observation of a significantly long portion of the sequence initially before scoring the incoming subsequences.

The pattern frequency-based formulation is more difficult to handle in an online setting, in which the test sequence $S$ is collected in an online fashion. The reason is that the frequency of occurrence of the query pattern in the test sequence will have to be estimated without observing the entire test sequence.

## 8 Concluding Remarks

One of the most interesting aspect of the problem of anomaly detection for sequences is the rich diversity of problem formulations. In this survey, we have discussed three different problem formulations that are relevant in varied application domains. We note that these three formulations are not exhaustive and the anomaly detection problem might be formulated in other ways also, though most of the existing work can be covered under the three formulations discussed here.

For each problem formulation, there are distinct groups of techniques that use a specific approach to detect anomalies. Within each group, we have identified a basic technique and shown how different existing techniques are variations of the corresponding basic technique. This results in a better understanding of the current state of research as well as allows future researchers to develop novel variations. For example, in the similarity-based techniques for sequence-based formulation (Section 4.1), using a different combination of a point-based anomaly detection algorithm and a similarity measure, a different anomaly detection technique can be developed. Similarly, by using different techniques to compare a pair of windows, one can develop novel variations of the basic window-based technique discussed in Section 5.1. In addition, techniques from one problem formulation can be adapted to solve a different formulation, thereby enriching the set of techniques to choose from, for a given formulation.

Although the focus of this survey paper has been on discrete sequences, many of the techniques discussed here are also applicable to continuous sequences (or time series). For example, similarity-based techniques can be adapted for continuous sequences by using an appropriate similarity/distance measure, such as *euclidean Distance* [86], [87] and *Cross Correlation* [88], [89]. Window-based techniques can be adapted by using the euclidean distance to $k$th closest window from the training sequence set to assign anomaly score to the window [87]. Markovian techniques can be adapted by substituting the Markovian model with a time series prediction model, which can determine the likelihood of observing a real valued event, given a finite history of events [90]. HMM-based techniques can be adapted by using a continuous version of HMM [91]. While some of these adaptations have already been proposed, others are subject of future research.

This paper has focused on techniques that deal with univariate discrete sequences. Many real-world applications deal with sequences of multivariate events, where the events might contain discrete [92], continuous [93], or a mixed set of observations [94]. Some of the techniques discussed in this paper are relevant to such settings, though additional thought still needs to be given on how to handle such complex sequences. For example, similarity and window-based techniques for sequence-based formulation, as well as the basic technique for subsequence-based problem formulation can be easily extended to multivariate sequences as long as a similarity measure can be developed to compare two multivariate discrete sequences. Recent work in the area of ranking patterns in sequences of item sets [95] can be extended to identify subsequence-based anomalies (low frequency patterns) in multivariate discrete sequences. Similarly, the recently proposed multievent pattern discovery [92] can be employed to identify pattern frequency-based anomalies.

While the literature on anomaly detection for discrete sequences is rich, there are several research directions that need to be explored in the future. In this survey, we have provided a high level comparison of strengths and limitations of various techniques. An experimental study of these

techniques is essential for a more in-depth understanding of their characteristics. Adapting existing solutions to other related problems, such as online anomaly detection and handling multivariate sequences, is also an important direction for future research.

## ACKNOWLEDGMENTS

This work was supported by NASA under award NNX08AC36A and US National Science Foundation (NSF) Grants CNS-0551551 and IIS-0713227. Access to computing facilities was provided by the Digital Technology Consortium. Varun Chandola was with the Department of Computer Science and Engineering, University of Minnesota, during the time of writing this paper.

## REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection - A Survey," *ACM Computing Surveys,* vol. 41, no. 3, pp. 1-58, July 2009.

[2] V. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Rev.,* vol. 22, no. 2, pp. 85-126, 2004.

[3] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," *Proc. SIAM Int'l Conf. Data Mining,* May 2003.

[4] S. Forrest, C. Warrender, and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternate Data Models," *Proc. IEEE Symp. Security and Privacy (ISRSP),* pp. 133-145, 1999.

[5] S.A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion Detection Using Sequences of System Calls," *J. Computer Security,* vol. 6, no. 3, pp. 151-180, citeseer.ist.psu.edu/hofmeyr98intrusion.html, 1998.

[6] C.C. Michael and A. Ghosh, "Two State-Based Approaches to Program-Based Anomaly Detection," *Proc. 16th Ann. Computer Security Applications Conf.,* p. 21, 2000.

[7] W. Lee, S. Stolfo, and P. Chan, "Learning Patterns from Unix Process Execution Traces for Intrusion Detection," *Proc. AAAI 97 Workshop AI Methods in Fraud and Risk Management,* 1997.

[8] W. Lee and S. Stolfo, "Data Mining Approaches for Intrusion Detection," *Proc. Seventh USENIX Security Symp.,* 1998.

[9] F.A. Gonzalez and D. Dasgupta, "Anomaly Detection Using Real-Valued Negative Selection," *Genetic Programming and Evolvable Machines,* vol. 4, no. 4, pp. 383-403, 2003.

[10] B. Gao, H.-Y. Ma, and Y.-H. Yang, "Hmms (Hidden Markov Models) Based on Anomaly Intrusion Detection Method," *Proc. Int'l Conf. Machine Learning and Cybernetics,* pp. 381-385, 2002.

[11] S. Budalakoti, A. Srivastava, R. Akella, and E. Turkov, "Anomaly Detection in Large Sets of High-Dimensional Symbol Sequences," Technical Report NASA TM-2006-214553, NASA Ames Research Center, 2006.

[12] S. Budalakoti, A. Srivastava, and M. Otey, "Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics,* vol. 37, no. 6, 2007.

[13] P. Sun, S. Chawla, and B. Arunasalam, "Mining for Outliers in Sequential Databases," *Proc. SIAM Int'l Conf. Data Mining,* 2006.

[14] V. Chandola, V. Mithal, and V. Kumar, "A Comparative Evaluation of Anomaly Detection Techniques for Sequence Data," *Proc. Int'l Conf. Data Mining,* 2008.

[15] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge Univ. Press, 1997.

[16] S. Forrest, S.A. Hofmeyr, A. Somayaji, and T.A. Longstaff, "A Sense of Self for Unix Processes," *Proc. IEEE Symp. Security and Privacy (ISRSP '96),* pp. 120-128, citeseer.ist.psu.edu/forrest96sense.html, 1996.

[17] E. Eskin, W. Lee, and S. Stolfo, "Modeling System Call for Intrusion Detection Using Dynamic Window Sizes," *Proc. DARPA Information Survivability Conf. and Exposition (DISCEX),* citeseer.ist.psu.edu/portnoy01intrusion.html, 2001.

[18] T. Lane and C.E. Brodley, "Temporal Sequence Learning and Data Reduction for Anomaly Detection," *ACM Trans. Information Systems and Security,* vol. 2, no. 3, pp. 295-331, 1999.

[19] G. Liu, T.K. McDaniel, S. Falkow, and S. Karlin, "Sequence Anomalies in the cag7 Gene of the Helicobacter Pylori Pathogenicity Island," *Proc. Nat'l Academy of Sciences USA,* vol. 96, no. 12, pp. 7011-7016, 1999.

[20] A.N. Srivastava, "Discovering System Health Anomalies Using Data Mining Techniques," *Proc. Joint Army Navy NASA Airforce Conf. Propulsion,* 2005.

[21] S. Chakrabarti, S. Sarawagi, and B. Dom, "Mining Surprising Patterns Using Temporal Description Length," *Proc. 24th Int'l Conf. Very Large Data Bases,* pp. 606-617, 1998.

[22] D. Pavlov and D. Pennock, "A Maximum Entropy Approach to Collaborative Filtering in Dynamic, Sparse, High-Dimensional Domains," *Proc. Advances in Neural Information Processing Systems,* 2002.

[23] D. Pavlov, "Sequence Modeling with Mixtures of Conditional Maximum Entropy Distributions," *Proc. Third IEEE Int'l Conf. Data Mining,* pp. 251-258, 2003.

[24] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 427-438, 2000.

[25] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection," *Applications of Data Mining in Computer Security,* pp. 78-100, Kluwer Academics, 2002.

[26] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data.* Prentice-Hall, Inc., 1988.

[27] J. Yang and W. Wang, "CLUSEQ: Efficient and Effective Sequence Clustering," *Proc. Int'l Conf. Data Eng.,* pp. 101-112, 2003.

[28] D. Ron, Y. Singer, and N. Tishby, "The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length," *Machine Learning,* vol. 25, nos. 2/3, pp. 117-149, 1996.

[29] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Visualization of Navigation Patterns on a Web Site Using Model-Based Clustering," *Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 280-284, 2000.

[30] P. Smyth, "Clustering Sequences with Hidden Markov Models," *Proc. Advances in Neural Information Processing Systems,* vol. 9, 1997.

[31] R.R. Sokal and C.D. Michener, "A Statistical Method for Evaluating Systematic Relationships," *Univ. of Kansas Scientific Bull.,* vol. 38, pp. 1409-1438, 1958.

[32] J.W. Hunt and T.G. Szymanski, "A Fast Algorithm for Computing Longest Common Subsequences," *Comm. ACM,* vol. 20, no. 5, pp. 350-353, 1977.

[33] N. Kumar, V.N. Lolla, E.J. Keogh, S. Lonardi, and C.A. Ratanamahatana, "Time-Series Bitmaps: A Practical Visualization Tool for Working with Large Time Series Databases," *Proc. SIAM Int'l Conf. Data Mining (SDM),* 2005.

[34] T. Lane and C.E. Brodley, "Sequence Matching and Learning in Anomaly Detection for Computer Security," *Proc. AI Approaches to Fraud Detection and Risk Management,* Fawcett, Haimowitz, Provost, and Stolfo, eds., pp. 43-49, 1997.

[35] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, "Lof: Identifying Density-Based Local Outliers," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 93-104, 2000.

[36] D. Endler, "Intrusion Detection: Applying Machine Learning to Solaris Audit Data," *Proc. 14th Ann. Computer Security Applications Conf.,* pp. 268-279, 1998.

[37] H. Debar, M. Dacier, M. Nassehi, and A. Wespi, "Fixed vs. Variable-Length Patterns for Detecting Suspicious Process Behavior," *Proc. Fifth European Symp. Research in Computer Security,* pp. 1-15, 1998.

[38] A.K. Ghosh, A. Schwartzbard, and M. Schatz, "Using Program Behavior Profiles for Intrusion Detection," *Proc. SANS Third Conf. and Workshop Intrusion Detection and Response,* citeseer.ist.psu.edu/ghosh99learning.html, Feb. 1999.

[39] A. Ghosh, A. Schwartzbard, and M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection," *Proc. First USENIX Workshop Intrusion Detection and Network Monitoring,* pp. 51-62, Apr. 1999.

[40] J.B.D. Cabrera, L. Lewis, and R.K. Mehra, "Detection and Classification of Intrusions and Faults Using Sequences of System Calls," *SIGMOD Record,* vol. 30, no. 4, pp. 25-34, 2001.

[41] A.P. Kosoresow and S.A. Hofmeyr, "Intrusion Detection via System Call Traces," *IEEE Software,* vol. 14, no. 5, pp. 35-42, Sept./Oct. 1997.

[42] D. Dasgupta and F. Nino, "A Comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics,* vol. 1, pp. 125-130, 2000.

[43] T. Lane and C.E. Brodley, "An Application of Machine Learning to Anomaly Detection," *Proc. 20th Nat'l Information Systems Security Conf.,* pp. 366-380, 1997.

[44] T. Lane, "Machine Learning Techniques for the Computer Security Domain of Anomaly Detection," PhD dissertation, Purdue Univ., 2000.

[45] D. Dasgupta and N. Majumdar, "Anomaly Detection in Multidimensional Data Using Negative Selection Algorithm," *Proc. IEEE Conf. Evolutionary Computation,* pp. 1039-1044, May 2002.

[46] S. Forrest, P. D'haeseleer, and P. Helman, "An Immunological Approach to Change Detection: Algorithms, Analysis and Implications," *Proc. IEEE Symp. Security and Privacy,* pp. 110-119, 1996.

[47] S. Forrest, A.S. Perelson, L. Allen, and R. Cherukuri, "Self-Nonself Discrimination in a Computer," *Proc. IEEE Symp. Security and Privacy,* pp. 202-212, 1994.

[48] S. Forrest and D. Dasgupta, "Novelty Detection in Time Series Data Using Ideas from Immunology," *Proc. Fifth Int'l Conf. Intelligence Systems,* 1996.

[49] S. Forrest, F. Esponda, and P. Helman, "A Formal Framework for Positive and Negative Detection Schemes," *IEEE Trans. Systems, Man and Cybernetics, Part B,* vol. 34, no. 1, pp. 357-373, Feb. 2004.

[50] A.K. Ghosh, J. Wanken, and F. Charron, "Detecting Anomalous and Unknown Intrusions against Programs," *Proc. 14th Ann. Computer Security Applications Conf.,* pp. 259-267, 1998.

[51] M. Wang, C. Zhang, and J. Yu, "Native Api Based Windows Anomaly Intrusion Detection Method Using SVM," *Proc. IEEE Int'l Conf. Sensor Networks, Ubiquitous, and Trustworthy Computing,* vol. 1, pp. 514-519, 2006.

[52] S. Tian, S. Mu, and C. Yin, "Sequence-Similarity Kernels for Svms to Detect Anomalies in System Calls," *Neurocomputing,* vol. 70, nos. 4-6, pp. 859-866, 2007.

[53] X. Li, J. Han, S. Kim, and H. Gonzalez, "Roam: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets," *Proc. Seventh SIAM Int'l Conf. Data Mining,* 2007.

[54] N. Ye, "A Markov Chain Model of Temporal Behavior for Anomaly Detection," *Proc. Fifth Ann. IEEE Information Assurance Workshop,* 2004.

[55] C. Marceau, "Characterizing the Behavior of a Program Using Multiple-Length N-Grams," *Proc. Workshop New Security Paradigms,* pp. 101-110, 2000.

[56] E. Eskin, W.N. Grundy, and Y. Singer, "Protein Family Classification Using Sparse Markov Transducers," *Proc. Int'l Conf. Intelligent Systems for Molecular Biology (ISMB '08),* pp. 134-145, 2000.

[57] W.W. Cohen, "Fast Effective Rule Induction," *Proc. 12th Int'l Conf. Machine Learning,* A. Prieditis and S. Russell, eds., pp. 115-123, July 1995.

[58] L.R. Rabiner and B.H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine,* vol. 3, no. 1, pp. 4-16, Jan. 1986.

[59] L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Math. Statistics,* vol. 41, no. 1, pp. 164-171, 1970.

[60] T. Lane, "Hidden Markov Models for Human/Computer Interface Modeling," *Proc. IJCAI-99 Workshop Learning about Users,* pp. 35-44, 1999.

[61] K. Yamanishi and Y. Maruyama, "Dynamic Syslog Mining for Network Failure Monitoring," *KDD '05: Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery in Data Mining,* pp. 499-508, 2005.

[62] J. Forney, G.D., "The Viterbi Algorithm," *Proc. IEEE,* vol. 61, no. 3, pp. 268-278, Mar. 1973.

[63] G. Florez, Z. Liu, S. Bridges, A. Skjellum, and R. Vaughn, "Lightweight Monitoring of Mpi Programs in Real Time," *Concurrency and Computation: Practice and Experience,* vol. 17, no. 13, pp. 1547-1578, 2005.

[64] Y. Qiao, X.W. Xin, Y. Bin, and S. Ge, "Anomaly Intrusion Detection Method Based on Hmm," *Electronics Letters,* vol. 38, no. 13, pp. 663-664, 2002.

[65] X. Zhang, P. Fan, and Z. Zhu, "A New Anomaly Detection Method Based on Hierarchical Hmm," *Proc. Fourth Int'l Conf. Parallel and Distributed Computing, Applications and Technologies,* pp. 249-252, 2003.

[66] E. Keogh, J. Lin, S.-H. Lee, and H.V. Herle, "Finding the Most Unusual Time Series Subsequence: Algorithms and Applications," *Knowledge and Information Systems,* vol. 11, no. 1, pp. 1-27, 2006.

[67] E. Keogh, J. Lin, and A. Fu, "Hot SAX: Efficiently Finding the Most Unusual Time Series Subsequence," *Proc. Fifth IEEE Int'l Conf. Data Mining,* pp. 226-233, 2005.

[68] J. Lin, E. Keogh, A. Fu, and H.V. Herle, "Approximations to Magic: Finding Unusual Medical Time Series," *Proc. 18th IEEE Symp. Computer-Based Medical Systems,* pp. 329-334, 2005.

[69] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A Novel Symbolic Representation of Time Series," *Data Mining and Knowledge Discovery,* vol. 15, no. 2, pp. 107-144, 2007.

[70] E. Keogh, S. Lonardi, and C.A. Ratanamahatana, "Towards Parameter-Free Data Mining," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 206-215, 2004.

[71] L. Wei, N. Kumar, V. Lolla, E.J. Keogh, S. Lonardi, and C. Ratanamahatana, "Assumption-Free Anomaly Detection in Time Series," *Proc. 17th Int'l Conf. Scientific and Statistical Database Management,* pp. 237-240, 2005.

[72] L. Wei, E. Keogh, and X. Xi, "Saxually Explicit Images: Finding Unusual Shapes," *Proc. Sixth Int'l Conf. Data Mining,* pp. 711-720, 2006.

[73] Y. Bu, T.-W. Leung, A. Fu, E. Keogh, J. Pei, and S. Meshkin, "Wat: Finding Top-k Discords in Time Series Database," *Proc. Seventh SIAM Int'l Conf. Data Mining,* 2007.

[74] A.W.-C. Fu, O.T.-W. Leung, E.J. Keogh, and J. Lin, "Finding Time Series Discords Based on Haar Transform," *Proc. Second Int'l Conf. Advanced Data Mining and Applications,* pp. 31-41, 2006.

[75] A. Ghoting, S. Parthasarathy, and M.E. Otey, "Fast Mining of Distance-Based Outliers in High-Dimensional Datasets," *Proc. SIAM Data Mining Conf.,* 2006.

[76] R. Gwadera, A. Gionis, and H. Mannila, "Optimal Segmentation Using Tree Models," *ICDM '06: Proc. Sixth Int'l Conf. Data Mining,* pp. 244-253, 2006.

[77] E. Keogh, S. Lonardi, and B.Y.C. Chiu, "Finding Surprising Patterns in a Time Series Database in Linear Time and Space," *Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 550-556, 2002.

[78] J.J. Schlesselman, *Case-Control Studies: Design, Conduct, Analysis (Monographs in Epidemiology and Biostatistics).* Oxford Univ. Press, 1982.

[79] R. Gwadera, M. Atallah, and W. Szpankowski, "Reliable Detection of Episodes in Event Sequences," *Knowledge and Information Systems,* vol. 7, no. 4, pp. 415-437, 2005.

[80] R. Gwadera, M. Atallah, and W. Szpankowskii, "Detection of Significant Sets of Episodes in Event Sequences," *Proc. Fourth IEEE Int'l Conf. Data Mining,* pp. 3-10, 2004.

[81] R. Gwadera, M.J. Atallah, and W. Szpankowski, "Markov Models for Identification of Significant Episodes," *Proc. Fifth SIAM Int'l Conf. Data Mining,* 2005.

[82] R.A. Maxion and K.M.C. Tan, "Benchmarking Anomaly-Based Detection Systems," *Proc. Int'l Conf. Dependable Systems and Networks,* pp. 623-630, 2000.

[83] A. Pawling, P. Yan, J. Candia, T. Schoenharl, and G. Madey, "Anomaly Detection in Streaming Sensor Data," *Intelligent Techniques for Warehousing and Mining Sensor Network Data,* IGI Global, 2008.

[84] D. Pokrajac, A. Lazarevic, and L.J. Latecki, "Incremental Local Outlier Detection for Data Streams," *Proc. IEEE Symp. Computational Intelligence and Data Mining,* 2007.

[85] G. Florez-Larrahondo, S.M. Bridges, and R. Vaughn, "Efficient Modeling of Discrete Events for Anomaly Detection Using Hidden Markov Models," *Information Security,* vol. 3650, pp. 506-514, 2005.

[86] G.K. Palshikar, "Distance-Based Outliers in Sequences," *Proc. Second Int'l Conf. Distributed Computing and Internet Technology,* pp. 547-552, 2005.

[87] D. Yankov, E.J. Keogh, and U. Rebbapragada, "Disk Aware Discord Discovery: Finding Unusual Time Series in Terabyte Sized Datasets," *Proc. Int'l Conf. Data Mining,* pp. 381-390, 2007.

[88] P. Protopapas, J.M. Giammarco, L. Faccioli, M.F. Struble, R. Dave, and C. Alcock, "Finding Outlier Light Curves in Catalogues of Periodic Variable Stars," *Monthly Notices of the Royal Astronomical Soc.,* vol. 369, no. 2, pp. 677-696, 2006.

[89] U. Rebbapragada, P. Protopapas, C.E. Brodley, and C. Alcock, "Finding Anomalous Periodic Time Series," *Machine Learning,* vol. 74, pp. 281-313, 2009.

[90] J. Ma and S. Perkins, "Online Novelty Detection on Temporal Sequences," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 613-618, 2003.
[91] Z. Liu, J.X. Yu, L. Chen, and D. Wu, "Detection of Shape Anomalies: A Probabilistic Approach Using Hidden Markov Models," *Proc. IEEE 24th Int'l Conf. Data Eng.*, pp. 1325-1327, Apr. 2008.
[92] R. Gwadera and F. Crestani, "Discovering Significant Patterns in Multi-Stream Sequences," *Proc. Eighth IEEE Int'l Conf. Data Mining*, pp. 827-832, 2008.
[93] H. Cheng, P.-N. Tan, C. Potter, and S. Klooster, "Detection and Characterization of Anomalies in Multivariate Time Series," *Proc. Ninth SIAM Int'l Conf. Data Mining*, 2009.
[94] R. Fujimaki, T. Nakata, H. Tsukahara, and A. Sato, "Mining Abnormal Patterns from Heterogeneous Time-Series with Irrelevant Features for Fault Event Detection," *Proc. SIAM Int'l Conf. Data Mining*, pp. 472-482, 2008.
[95] R. Gwadera and F. Crestani, "Ranking Sequential Patterns with Respect to Significance," *Proc. 14th Pacific-Asia Conf. Knowledge Discovery and Data Mining, (PAKDD '[10)*, pp. 286-299, 2010.

**Varun Chandola** received the PhD degree from the University of Minnesota, Twin Cities, in 2009. He is a postdoctoral research associate in the Geographic Information Science and Technology group at Oak Ridge National Labs. His areas of expertise include data mining, time series data analysis, machine learning, and algorithm development. His research focus is in the area of anomaly detection applied to discrete sequences and time series data. He has applied his research, with significant success, to varying application domains, such as biomass monitoring, aviation safety, cyber intrusion detection, tax fraud analysis, cardiac health monitoring, and click fraud analysis.

**Arindam Banerjee** received the PhD degree from the University of Texas at Austin in 2005. He is an assistant professor and a McKnight Land grant professor in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. His research interests are in data mining and machine learning, and their applications to real-world problems. His work currently focuses on statistical and graphical models for learning and predictive modeling with large-scale data. His research interests also include information theory and convex analysis, and applications in complex real-world learning problems including problems in text and web mining, bioinformatics, and social network analysis. He is a member of the IEEE.

**Vipin Kumar** received the BE degree in electronics and communication engineering from Indian Institute of Technology Roorkee (formerly, University of Roorkee), India, in 1977, the ME degree in electronics engineering from Philips International Institute, Eindhoven, Netherlands, in 1979, and the PhD degree in computer science from the University of Maryland, College Park, in 1982. He is currently William Norris professor and head of the Computer Science and Engineering Department at the University of Minnesota. His current research interests include data mining, high-performance computing, and their applications in Climate/Ecosystems and Biomedical domains. He has authored more than 250 research articles, and has coedited or coauthored 11 books including widely used text books *Introduction to Parallel Computing* and *Introduction to Data Mining*. He is a founding co-editor-in-chief of *Journal of Statistical Analysis*, a cofounder of SIAM International Conference on Data Mining, and editor of *Data Mining and Knowledge Discovery* Book Series published by CRC Press/Chapman Hall. He received the 2009 Distinguished Alumnus Award from the Computer Science Department, University of Maryland, College Park, and 2005 IEEE Computer Society's Technical Achievement Award for contributions to the design and analysis of parallel algorithms, graph partitioning, and data mining. He is a fellow of the ACM, IEEE, and AAAS.