Open access • Journal Article • DOI:10.1109/TKDE.2010.235

Anomaly Detection for Discrete Sequences: A Survey — Source link

Varun Chandola, Arindam Banerjee, Vipin Kumar

Institutions: Oak Ridge National Laboratory, University of Minnesota

Published on: 01 May 2012 - IEEE Transactions on Knowledge and Data Engineering (IEEE Computer Society)

Topics: Anomaly detection

Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 09-015

Anomaly Detection for Discrete Sequences: A Survey

Varun Chandola, Arindam Banerjee, and Vipin Kumar

May 22, 2009

Anomaly Detection for Discrete Sequences: A Survey

Varun Chandola, Arindam Banerjee and Vipin Kumar

Abstract—This survey attempts to provide a comprehensive and structured overview of the existing research for the problem of detecting anomalies in discrete sequences. The aim 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 as follows: We identify three distinct formulations of the anomaly detection problem, and review techniques from many disparate and disconnected 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 highlight the applicability of the techniques that handle discrete sequences to other related areas such as online anomaly detection and time series anomaly detection.

1 Introduction

Sequence data is found in a wide variety of application domains such as intrusion detection, bio-informatics, weather prediction, system health management, etc. Hence anomaly detection for sequence data is an important topic of research. There is extensive work on anomaly detection techniques [1]–[3] that look for individual objects that are different from normal objects. These techniques 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. 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 unordered alphabet. For example, a text document is a sequence of words, a computer program is executed as a sequence of system calls, a gene is a sequence of nucleic acids.

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	$\mathbf{login}, \mathbf{passwd}, \mathbf{login}, \mathbf{passwd}, \mathbf{login}, \mathbf{passwd}, \ldots, \mathbf{logout}$

TABLE 1
Sequences of User Commands

Often, one is interested in detecting anomalies in discrete sequences to find possible intrusions, frauds, faults, contaminations. For example, the sequences of commands issued by computer users (as shown in Table 1) are collected to detect possible intrusive activities.

Anomaly detection for discrete sequences is a challenging task, since it involves exploiting the sequential nature of data to detect anomalies. Some of the specific challenges are as follows:

- Multiple definitions of anomalies exist for sequences; an event within a sequence maybe anomalous, a subsequence within a sequence maybe anomalous, or the entire sequence maybe anomalous. Each definition needs to be handled differently. For example, given a data base of protein sequences, one might be interested in sequences that are anomalous. On the other hand, given a long sequence of system calls executed in a computer system, one might be interested in detecting subsequences when the computer system was under a virus attack. A technique that detects anomalous events within a sequence might not be directly applicable to detecting anomalies that are caused by a subsequence of events occurring together.
- The length of the anomaly within a sequence is usually not known and varies significantly across different application domains. Techniques typically have to rely on user-defined lengths, which may or may not be optimal.
- Computational complexity is a significant issue for sequence data, especially since sequences can be very long and the alphabet size can be large.

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. In particular, several techniques have been developed to detect anomalies in operating system call data [4]–[10]. Budalakoti et al [11], [12] present a study of anomaly detection techniques to detect anomalies in flight safety domain. Sun et al [13] present a probabilistic suffix tree based technique and evaluate it on biological sequences. While each technique has been evaluated within a specific domain, there has not been an attempt to analyze the problem in its entirety. The reason such analysis is essential is that the nature of the sequence data as well as the nature of anomalies can differ fundamentally across domains, and hence while a technique is shown to be effective within one domain, it does not guarantee its effectiveness 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. For example, Forrest et al [4] proposed several techniques that detect an anomalous sequence from a database of unlabeled sequences. On the other hand, Chakrabarti et al [14] detect an anomalous subsequence within a long sequence of events. Gwadera et al [15] address the problem of determining if the frequency of occurrence of a short subsequence in a given long test sequence is anomalous with respect to its expected frequency of occurrence in normal sequences. All of these formulations are fundamentally distinct, and hence require exclusive techniques to solve them. It is not only important to identify which problem formulation is addressed by a given technique, but also to understand how techniques that address one formulation can be adapted to solve a different formulation.

The literature on anomaly detection for sequences is scattered across varied application domains, but there has not been any study that provides a global overview of the literature. A comparison of four different anomaly detection techniques for discrete sequences was presented by Forest et al [4], but all of the techniques focussed on system call intrusion detection. Chandola et al [16] provided a comparative evaluation of eight anomaly detection techniques on sequence data collected from different domains, but only focussed on the problem of detecting anomalous sequences from a database of sequences.

1.1 Our Contributions

In this survey we attempt to provide a comprehensive and structured overview of the existing research in this context. We aim to provide a global understanding of the sequence anomaly detection problem and how techniques proposed for different domains relate to each other. We also study the different problem formulations and show how techniques developed for a particular problem formulation can be adapted and applied in a different context.

Our specific contributions are:

- We identify three distinct formulations of the anomaly detection problem, and review techniques from many disparate and disconnected 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 highlight the applicability of the techniques that handle discrete sequences to other related areas such as online anomaly detection and time series anomaly detection.

1.2 Organization

The rest of this survey article is organized as follows.Section 2 describes different application domains where anomaly detection techniques for discrete sequences have been developed. Section 3 defines three different ways in which the anomaly detection problem can be formulated. Sections 4, 5, and 6 provide an overview of techniques that have been proposed to solve each of 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:

- *Sequence of operating system calls/user commands* [5], [17]–[19]. The sequences are defined using an alphabet of all possible system calls (~ 160 for Sun Solaris operating system) or user commands (~ 2500 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.

- *Biological sequences such as DNA sequences, protein sequences* [13], [20]. The sequences are defined using an alphabet that corresponds to nucleic acid bases (4), or amino acid bases ($\sim$ 20). Anomalies in such data sets correspond to either mutations, or a "disease" condition in the biological organism [21].
- *Sensor data from operational systems (such as aircrafts or space shuttles)* [11], [12], [22]. The sequences are collected during the operation of the system through multiple discrete sensors. The alphabet size for such discrete sequences is typically large ($\sim$ 1000 for aircraft data [12]). Anomalies in such data sets corresponds to faults scenarios in the operation of the system or accidents.
- *Sequences of transactions from online banking, customer purchases, and other retail commerce domains* [14]. 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.
- *Navigational click sequences from web sites* [23], [24]. The "symbols" in such sequences correspond to clicked links (web-sites), or categories to which the links belong to. The anomalies in such data correspond to irregular or unauthorized user behavior.

3 Problem Definitions

Here we will discuss three different formulations of the sequence anomaly detection problem that are considered important in many application domains. Each of these formulations are unique in terms of the 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 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*: 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 is tested against this training data.
- *Scenario 2*: A security analyst is interested in detecting if a user's account was misused (hacked) in the past few months. To detect this misuse, the analyst can use the second formulation, in which the user's activity for the past few months is considered as a long sequence, and is tested for any anomalous subsequence.
- *Scenario 3*: 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 first problem formulation has been addressed in the majority of papers, but the other two formulations have also been addressed in certain domains. The next three sections discuss techniques that address these three problem formulations.

4 Determining if a sequence is anomalous w.r.t. a sequence database

The most common formulation of anomaly detection problem for sequences is to determine if a given test sequence is anomalous with respect to a reference database of sequences. One application of this formulation was provided as Scenario 1 in Section 3. Most techniques that solve this formulation, assign an anomaly score to the test sequence. If a set of test sequences is given, the anomaly score is used to rank them, to determine the most anomalous test sequences.

Two variants of this problem formulation exist. In the first variant, the reference (or training) database is assumed to contain only normal sequences and each test sequence is tested against the normal reference database (semi-supervised anomaly detection). In the second variant, the task is to detect anomalous sequences from an unlabeled database of sequences (unsupervised anomaly detection). For the second variant, it is assumed that majority of sequences in the unlabeled database are normal.

The semi-supervised variant of problem formulation 1 can be stated as following:

Definition 1: Given a set of n sequences, $\mathbf{S}$ = $\{S_1, S_2, \ldots, S_n\}$, and a sequence S_q belonging to a test data set $\mathbf{S_q}$, compute an anomaly score for S_q with respect to a training data set $\mathbf{S}$.

The length of sequences in $\mathbf{S}$ and the length of S_q might not be equal. After assigning an anomaly score to the

test sequence, an additional test is required to determine if the anomaly score is significantly large for the test sequence to be considered anomalous.

The semi-supervised variant of problem formulation 1 can be stated as following:

Definition 1a: Given a set of n sequences, $\mathbf{S}$ = $\{S_1, S_2, \ldots, S_n\}$, find all sequences in $\mathbf{S}$ that are anomalous w.r.t. rest of $\mathbf{S}$.

In this section we will primarily discuss techniques that handle the semi-supervised variant. A semi-supervised 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 semi-supervised technique does not get affected by the presence of anomalies in the "training" set.

Techniques that solve problem formulation 1 typically operate in two steps. In the first step, a "model" representing the normal behavior is learnt from sequences in $\mathbf{S}$. In the second step, the "likelihood" of the test sequence S_q to be generated by model $\mathcal{M}$ is estimated. The likelihood is used to assign an anomalous label/score to the test sequence.

We use the term model in a loose fashion and it may refer to a *generative model*, a *non-parametric density estimate*, or a *similarity kernel*. Examples of generative models are finite state automata (FSA), hidden markov models (HMM), and mixture of HMMs. Several techniques use a similarity kernel between the test sequences in $\mathbf{S_q}$ and the normal sequences in $\mathbf{S}$. Additionally, there is a class of techniques in which the likelihood of a test sequence is estimated using the likelihood of short windows contained in the test sequence. Thus the model $\mathcal{M}$ for such techniques is the density estimate of the different short windows that occur in the test and training sequences.

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

- *Kernel 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 kernel for the sequences.
- *Window Based Techniques:* These techniques analyze a short window of symbols—a short subsequence—within the test sequence at a time. Thus such techniques treat a subsequence 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 subsequences 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 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 subsections.

4.1 Kernel Based Techniques

These techniques compute pairwise similarity between sequences using a specific similarity measure and then make use of a traditional point (or instance) based anomaly detection algorithm (which utilizes the similarities) [11], [12], [16].

A basic kernel based technique operates as follows. First, a pairwise similarity matrix is computed for the training sequences in $\mathbf{S}$. Next, the test sequence S_q is compared against the similarity matrix to obtain the anomaly score for S_q, using a point based anomaly detection algorithm.

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.

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 [25] and clustering based [26]. Chandola et al [16] proposed a kNN based technique in which the anomaly score of a test sequence is equal to the inverse of its similarity 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 [27]. 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 et al [28] propose a technique that uses a mixture of *Probabilistic Suffux Trees* [29] as cluster representations. Other probabilistic techniques such as mixture of HMMs [30], [31] or mixture of *Maximum Entropy* (maxent) models [24] can also be employed in the same manner for clustering based anomaly detection.

Using Different Similarity Measures

The simplest similarity measure for comparing a pair of discrete sequences is the *Simple Matching Coefficient* (SMC) [32] 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], [16]. This similarity ($nLCS$) between two sequences S_i and S_j, is computed as:

$$nLCS(S_i, S_j) = \frac{|LCS(S_i, S_j)|}{\sqrt{|S_i||S_j|}} \quad (1)$$

where $LCS(S_i, S_j)$ is the longest common subsequence shared by S_i and S_j.

The disadvantage of using $nLCS$ is the computational complexity involved, which is an order of magnitude more than that of SMC, though faster algorithms to compute the LCS have been proposed [33]. 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.

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 [34] by converting the sequences into bitmaps and comparing the bitmaps to determine the similarity.

Advantages and Disadvantages of Kernel Based Techniques. The advantage of kernel based techniques is that after obtaining a similarity kernel, any similarity based point anomaly detection technique can be applied to detect anomalies. Techniques that involve computing similarity can leverage the existing research on sequence similarity [19], [20], [35], and then apply known nearest neighbor or clustering based anomaly detection techniques [26], [36].

A disadvantage of kernel 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 $nLCS$ fail to pick up such minor difference. Another disadvantage of kernel 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.

The motivation behind window based techniques can be understood by first understanding the limitation of kernel based techniques. Probabilistically speaking, kernel based techniques indirectly estimate $P(S_q|\mathcal{M})$, which is the conditional probability of occurrence of an entire test sequence S_q, given the model $\mathcal{M}$ learnt from the training data set. Researchers have argued that often, the cause of anomaly can be localized to one or more shorter subsequences within the actual sequence [17]. If the entire sequence is analyzed as a whole, the anomaly signal might not be distinguishable 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 S, the extracted windows are denoted by $\omega_1, \omega_2 \ldots \omega_t$ and each window ω_i can also be denoted as $\omega_i[1]\omega_i[2] \ldots \omega_i[k]$.

Another way to understand the motivation behind the window based techniques. Let us assume that an anomalous test sequence S_q contains a subsequence a_q (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 subsequence a_q will occur (partly or whole) in $|a_q| + k - 1$ windows. Thus the anomalous sequence can be potentially detected by detecting at least one of such windows.

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, S_q. A window ω_i is assigned a likelihood score $L(\omega_i)$ which is equal to the frequency associated with the window ω_i in the normal dictionary. A threshold λ is used to determine if a particular window, ω_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, S_q is given by:

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

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

Several variants of the t-STIDE technique have been proposed, especially for system call intrusion detection [17], [19], [35], [37]–[41]. 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 ω_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 score to a window, ω_i [17]. The normal dictionary is constructed as follows. For each symbol that occurs in $\mathbf{S}$, the symbols that occur j positions after that symbol ($\forall j \in [1, k]$) are recorded. To assign anomaly score to ω_i, the number of symbol pairs of the form $\langle \omega_i[1], \omega_i[j] \rangle, \forall j \in (1, k]$ are counted, such that $\omega_i[1]$ is not followed by $\omega_i[j]$ after j positions in the normal dictionary. The total number of mismatching lookahead pairs divided by k is the anomaly score for the window.

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 ω_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 ω_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 [41], [42]. The latter approach has been adopted by [37], [39], [40], [43].

Lane and Brodley [19], [35], [44] use the following similarity measure to compute similarity $Sim(\omega_i, \vartheta_j)$ between a test window, ω_i and a window in the normal dictionary, ϑ_j:

$$w(\omega_i, \vartheta_j, l) = \begin{cases} 0 & if\ i = 0\ or\ \omega_i[l] \neq \vartheta_j[l] \\ 1 + w(\omega_i, \vartheta_j, l-1) & if\ \omega_i[l] = \vartheta_j[l] \end{cases} \tag{3}$$

and

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

Note that in 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 ω_i is equal to the inverse of maximum similarity between ω_i and windows in the normal dictionary. Lane's PhD. thesis [45] 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 [43], [46]–[49]. For example, Dasgupta et al [43] 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 [50].

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 by [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], [40], [43]. 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 learnt. Different type of classification models have been learnt, such as HMM based [10], neural networks [9], [37], [40], [51], SVM [52], [53], and rule based classifiers [54]. During testing, the anomaly score for each window ω_i obtained from test sequence S_q, $A(\omega_i) = 0$ 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, S_q, are assigned an anomaly score, the next step is to combine the anomaly scores (a vector of length $|S_q| - k + 1$) to obtain an overall anomaly score $A(S_q)$. 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 S_q.

$$A(S_q) = \frac{1}{t} \sum_{i=1}^{t} A(\omega_i) \tag{5}$$

The second method checks if any window in S_q has a mismatch.

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

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 S_q (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 S_q, the LFC method gives a lower anomaly score to S_q. 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 S_q 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 by [40] 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 kernel 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 large amount of memory.

4.3 Markovian Techniques

The third category of techniques that solve problem formulation 1 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 S ($= \langle s_1, s_2, \ldots s_l \rangle$) is factorized using the chain rule:

$$P(S) = \prod_{i=1}^{l} P(s_i | s_1 s_2 \ldots s_{i-1}) \tag{7}$$

where l is the length of the sequence and s_i is the symbol occurring at position i in S.

Markovian techniques utilize the *short memory* property of sequences, which is observed across different domains [29]. The short memory property is essentially a higher-order Markov condition which states that the conditional probability of occurrence of a symbol s_i, given the sequence observed so far can be approximated as:

$$P(s_i | s_1 s_2 \ldots s_{i-1}) = P(s_i | s_{i-k} s_{i-k-1} \ldots s_{i-1}) \quad k > 1 \tag{8}$$

Markovian techniques operate in two phases, training and testing. Training involves learning a probabilistic model using training sequences in $\mathbf{S}$. Testing involves calculating the conditional probabilities for each symbol of the test sequence, using the model, and then combining them to obtain an overall probability for the test sequence using (7). Finally, an anomaly score $A(S_q)$ for the test sequence is calculated as the inverse of the probability of S_q.

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 s_{qi} as:

$$P(s_{qi} | s_{q(i-k)} \ldots s_{q(i-1)}) = \frac{f(s_{q(i-k)} \ldots s_{q(i-1)} s_{qi})}{f(s_{q(i-k)} \ldots s_{q(i-1)})} \tag{9}$$

where $f(s_{q(i-k)} \ldots s_{q(i-1)} s_{qi})$ is the frequency of occurrence of the subsequence $s_{q(i-k)} \ldots s_{q(i-1)} s_{qi}$ in the sequences in $\mathbf{S}$ and $f(s_{q(i-k)} \ldots s_{q(i-1)})$ is the frequency of occurrence of the subsequence $s_{q(i-k)} \ldots s_{q(i-1)}$ in the sequences in $\mathbf{S}$.

Different variants of the basic technique have been proposed. Ye [55] proposed one such technique for $k = 1$.

An issue with the basic fixed Markovian technique is that storing all transition frequencies (See (9)) can potentially require a huge amount of space. Let the alphabet set size for a given sequence data set by σ. The the total number of frequencies required by the basic technique will be $\approx \sigma^{k+1}$.

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 subsequence. A node has a transition to another node only if the subsequence for the second node can be obtained from the subsequence for the first node by

removing the first symbol of the first node and appending any symbol at the end of the subsequence. The number of times the subsequence 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 S_q. 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 by substituting the frequencies stored in the current node and the transition in (9). If the transition is not defined the symbol is ignored. Chandola et al [16] 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.

The EFSA based technique was also extended by Michael and Ghosh [6] to the case in the conditional probability for 2 or more symbols, given the previous k symbols is used to obtain the final anomaly score for S_q.

Another variant of the basic technique was proposed by Marceau [56] which uses *suffix trees*. In this technique the suffix tree only maintains the $k+1$ length subsequences 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 subsequences are extracted from S_q 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 subsequence.

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 subsequence, i.e., $(s_{q(i-k)} \dots s_{q(i-1)})$, may not be sufficiently large to provide a reliable estimate of the conditional probability of a symbol that follows this subsequence. For example, let us assume that the subsequence $aabbb$ 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 subsequence $abbb$ occurs significant number of times in the training sequences.

One such technique was proposed by Sun et al [13] using *Probabilistic Suffix Trees* (PST) [29]. 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 subsequence obtained by traversing the path from root to the node, as well as the number of times the subsequence is observed in the training sequences. Each node also stores the conditional probability of observing each symbol in the alphabet, given the subsequence 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 criterion 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.

During the testing phase for the PST based technique the test sequence is scanned and the PST is traversed simultaneously. For a symbol s_{qi} in the test sequence S_q, its conditional probability is estimated by finding the longest suffix of the k length subsequence that precedes s_{qi} (in S_q) and occurs as a path in the PST. Thus, different symbols are conditioned on a different sized history. To obtain the likelihood score, $L(S_q)$ (inverse of anomaly score), of the test sequence, two combination techniques have been proposed:

$$L(S_q) = \frac{1}{|S_q|}\left(\sum_{i=1}^{|S_q|} \log P(s_{qi}|s_{q(i-j+1)\dots s_{q(i-1)}})\right) \quad (10)$$

and

$$L(S_q) = \frac{\prod_{i=1}^{|S_q|} P(s_{qi})}{\prod_{i=1}^{|S_q|} p(s_{qi}|s_{q(i-j+1)\dots s_{q(i-1)}})} \quad (11)$$

where $p(s_{qi})$ is the empirical probability of symbol s_i in $\mathbf{S}$ and $j \leq k$. Sun et al [13] show that the combination technique in (10) performs better than the combination technique in (11) for anomaly detection using protein sequences.

4.3.3 Sparse Markovian Techniques

Variable Markovian techniques described above allow a symbol s_{qi} 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 s_{qi} in S_q. Sparse Markovian techniques are more flexible in the sense that they estimate the conditional probability of s_{qi} based on symbols within the previous k symbols, which are not necessarily contiguous or immediately preceding to s_{qi}. 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 [18], using *Sparse Markov Transducers* (SMT) [57]. 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 learnt 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 [58]) 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 s_{qi}, the first rule that fires for the vector $s_{q(i-k+1)} \dots s_{q(i-1)}$ is chosen. If the target of the selected rule is s_{qi}, then the probability $P(s_{qi}|s_{q(i-k+1)} \dots s_{q(i-1)}) = 1$. If the target of the selected rule is not s_{qi}, then $P(s_{qi}|s_{q(i-k+1)} \dots s_{q(i-1)})$ 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, hence 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 it immediate context. Thus such techniques are able to detect anomalous sequences even if the anomalies are localized. The variable and sparse Markovian techniques provide flexibility in terms of the size of history to observe for every symbol. Thus if in a normal test sequence, the probability of observing a symbol is low with respect to its k length history, the variable and sparse techniques will estimate the probability with respect to a shorter history, and hence the symbol will have a higher probability. Thus such techniques help reduce the false positive rate.

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 [16] 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. Another disadvantage of the Markovian techniques is that they are sensitive to the choice of the length of the history.

4.4 Hidden Markov Models Based Techniques

Hidden Markov Models (HMM) are powerful finite state machines that are widely used for sequence modeling [59]. HMMs have also been applied to sequence anomaly detection [4], [60]–[62]. Such techniques use HMM to transform the input sequences from the symbol space to the hidden state space. The underlying assumption for such techniques is that the nature of the sequences is captured effectively by the hidden space.

The general approach is to construct an HMM with σ hidden states, from the normal sequences in $\mathbf{S}$ using the *Baum Welch* algorithm [63]. For each normal sequence in $\mathbf{S}$, as well as the test sequence S_q, the optimal state transition sequence is obtained using the *Viterbi* algorithm [64]. At this point any sequence anomaly detection technique that has been discussed in previous sections can be applied to estimate the anomaly score for the test sequence.

One such technique is proposed in [4], where 1-order finite Markovian technique is applied on the sequences of hidden states for anomaly detection. Qiao et al [60] apply a window based technique in the hidden state space. An extension to the latter technique was proposed by Zhang et al [61], 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. HMM based techniques transform the input sequences into sequences of hidden states. Thus, if the assumption that the training sequences in $\mathbf{S}$ are generated by a particular HMM hold true, the transformed data (hidden state sequences) will result in better anomaly detection.

If the above assumption regarding sequences in $\mathbf{S}$ to be generated by the HMM does not hold true, the performance of the HMM based technique can be worse than the performance of an anomaly detection technique on the original data. Another disadvantage of HMM based techniques is that initializing the HMM (number of hidden states, initial transition matrix) is often not intuitive, and poor choice for these can often result in sub-optimal performance of the technique.

5 Detecting anomalous subsequences within a long sequence

Techniques under this category solve the following anomaly detection problem:

Definition 2: Detect short subsequences in a long sequence T, that are anomalous with respect to rest of T.

Such anomalous subsequences can also be considered as *discords*, as defined by Keogh et al [65]: "*subsequences of a longer time series that are maximally different to rest of the time series subsequences*". We will refer to such anomalous subsequences as discords in this section.

This problem definition 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 T and stored as a database of fixed length windows, denoted as T_k. Each window is assigned an anomaly score by comparing it with rest of the windows in 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*, the techniques that solve this problem generally assume that the discords are contained within a 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 [65]–[67]. 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) [68].

An issue with the basic technique is that when any window is compared with the other windows in T_k, the windows which overlap with the given window in T, will be highly "similar" to the given window. For example, any window will differ in at most 1 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 [66] propose a simple solution (referred to as *non-self match* in the paper) to this issue by comparing a window with only the windows in 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 score the windows differently, while the second category of techniques address 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, 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 be not be possible to find exact matches of the window in T_k when the value of k is large.

Another possible technique for scoring the windows would be that the anomaly score of any window is calculated as equal to its distance to its m^{th} nearest neighbor in T_k. One such variant, called HOTSAX [66], was proposed by Keogh et al, in which the $m = 1$, i.e., the anomaly score of a window is equal to its distance to its nearest neighbor in T_k (only the non-self 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 [69]. To assign anomaly score to each window in the sequence T, the window is compared against rest of the sequence (say T') using an information theoretic measure called *Compression Based Dissimilarity* (CDM). For a window $\omega_i \in T_k$, the CDM is defined as:

$$CDM(\omega_i, T') = \frac{\mathcal{C}(\omega_i T')}{\mathcal{C}(\omega_i) + \mathcal{C}(T')} \tag{12}$$

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 (12) is that if the window ω_i is normal, it will match the rest of the sequence very well and hence will not require extra space when both ω_i and T' are compressed together. On the other hand, if ω_i is a discord, it will not match the rest of the sequence and hence the value of $\mathcal{C}(\omega_i T')$, and hence the anomaly score, will be high.

Wei et al [70] 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* (SMC), *edit distance*, *length of longest common subsequence*, and *weighted SMC* can be used. Wei et al

[70] use a visualization technique called *time series* or *chaos bitmaps* to compute similarity between windows. The authors assume that the sequences consist of four symbols. Each window is represented as a $l \times l$ bitmap ($l \leq k$), which is a matrix that contains the number of times any l length subsequence occurs in the given window. The matrix is normalized by dividing each value by the maximum value exists in the matrix.

5.2 Optimizing the Complexity of the Basic Technique

An issue with the basic technique that solves problem formulation 2 is that it requires $O(l^2)$ comparisons of window pairs, where l is the length of the sequence T. Several faster variations have been proposed that can run in approximately linear time in the context of continuous time series [66], [67], [71]–[73], 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. Such pruning has been successfully used for traditional anomaly detection [74], and has been applied to discord detection [66], [67], [71]. 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 from the given sequence T, as proposed by Chakrabarti et al [14]. 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 $O(|T|^2)$ solution exists to find such segmentation when number of items is 1, i.e., for a binary sequence. Approximate algorithms are proposed to handle the case when number of items is more than 1, though the technique is limited to small item set sizes, or small alphabets.

5.4 Relationship Between Techniques Solving Problem Formulation 1 and Problem Formulation 2

Techniques that handle problem formulation 1 and the techniques that handle problem formulation 2 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.

Using Techniques for Problem Formulation 1 to Solve Problem Formulation 2: The basic anomaly detection technique described earlier in this section assigns an anomaly score to a given window ω_i with respect to a data base of windows T_k (a majority of which are assumed to be normal). Any anomaly detection technique that solves problem formulation 1, as 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 ω_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 ω_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.

Using Techniques for Problem Formulation 2 to solve Problem Formulation 1: If all training sequences in $\mathbf{S}$ and the given test sequence S_q in problem formulation 1 are arranged linearly, in no particular order, a long sequence can be constructed. An anomaly detection technique discussed to handle problem formulation 2 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 the training sequences, and hence will face same challenges as the kernel 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 Determining if the frequency of a query pattern in a given sequence is anomalous w.r.t its expected frequency

Techniques under this category solve the following anomaly detection problem:

Definition 3: Given a short query pattern s, a long test sequence S and a training set of long sequences $\mathbf{S}$, determine if the frequency of occurrence of s in S is anomalous with respect to frequency of occurrence of s in $\mathbf{S}$.

This problem has also been referred to as *surprise detection* in the context of time series data [68], [75]. Keogh et al [75] 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 above 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. Another possible motivation for this formulation could be in the *case vs control* analysis [76], popular in epidemiological studies. The idea is to detect patterns whose 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 (s) existing in a known anomalous sequence (S) based on their frequency in S.

6.1 A Basic Technique to Solve Problem Formulation 1

A basic technique to solve the above problem assigns an anomaly score to the query pattern s, as follows: Count the number of occurrences of the query pattern in the sequence S and all sequences in $\mathbf{S}$, and compute the anomaly score for s as the difference between the number of times s occurs in S and the expected number of times s occurs in any sequence in $\mathbf{S}$.

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

$\bar{f}_S(s)$ is the frequency with which the query pattern occurs in the test sequence S, normalized by the length of S, and can be directly computed as:

$$\bar{f}_S(s) = \frac{f_S(s)}{|S|} \tag{13}$$

$\bar{f}_{\mathbf{S}}(s)$ is the expected frequency of the query pattern to occur in a sequence in $\mathbf{S}$ normalized by the length of the sequence. $\bar{f}_{\mathbf{S}}(s)$ can be estimated as:

$$\bar{f}_{\mathbf{S}}(s) = \frac{1}{|\mathbf{S}|} \sum_{\forall S_i \in \mathbf{S}} \frac{f_{S_i}(s)}{|S_i|} \tag{14}$$

The anomaly score of the query pattern s is computed as:

$$A(s) = |\bar{f}_S(s) - \bar{f}_{\mathbf{S}}(s)| \tag{15}$$

6.2 Variations of the Basic Technique

In the basic technique a query pattern s is considered to "occur" in a sequence if s is a *substring* of the sequence. An issue with considering substrings is that it forces the query pattern to occur exactly. If s is long, it is unlikely for entire s to occur in a training sequence. Another issue is that in several domains, it might be 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 s occurs in a sequence*. Keogh et al [75] find the largest l in the interval $[1, |s|)$, such that every l length substring of s occurs at least once in the training sequences. The frequency $f_{\mathbf{S}}(s)$ from (14) is then replaced with the following quantity:

$$f_{\mathbf{S}}(s) \Rightarrow \frac{\prod_{j=1}^{m-l} f_{\mathbf{S}}(s[j, j+l])}{\prod_{j=2}^{m-l} f_{\mathbf{S}}(s[j, j+l-1])}$$

  where $s[j,k]$ denotes the substring of s starting at j^{th} location of s and ending at the k^{th} location. The idea is to estimate the occurrence of the query pattern s using a set of substrings of s. 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 s occurs as a* subsequence *in a sequence*. An issue with the basic technique is that counting the number of times s occurs as a subsequence in a long sequence is expensive. To make the formulation more tractable, Gwadera et al [15] extract all windows of a fixed length (greater than the length of s), from the sequence and then determine all those windows that contain s as a subsequence. Thus a query pattern s is considered to "occur" in a sequence if there exists a window such that s is a subsequence of the given window. The number of fixed-length windows which contain s as a subsequence are counted. These counts are used as $f_S(s)$ and $f_{S_i}(s)$ in (13) and (14), respectively.
- *Counting the number of times any permutation of the query pattern s occurs in a sequence*. This alternative was proposed by Gwadera et al [77] as an extension to the subsequence based technique [15]. For the permutation approach, the number of fixed-length windows which contain any permutation of s are counted. The motivation behind considering all permutations is that in certain scenarios, the ordering of events (or symbols) within the query pattern s do not matter.

6.3 Issues with the Basic Technique

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

Computational Complexity

For each query pattern, the time required to compute its anomaly score is linear in the length of S and the length and number of sequences in $\mathbf{S}$. If there are multiple query patterns, e.g., all short subsequences that occur in S, 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 [68], [75] which uses *suffix trees* to efficiently compute the frequency of occurrence of a query pattern in a given sequence. Suffix trees are created for S and for each sequence in $\mathbf{S}$[1]. Only two suffix trees are required, one for the sequences in $\mathbf{S}$ and for the sequence S, and can be constructed with complexity linear in the length of the sequences. The counts for a query pattern s, can be obtained with complexity linear in the length of s. Gwadera et al [15] [77] use *Interpolated Markov Models* (IMM) [78] to efficiently find the number of windows extracted from a sequence that contain the query pattern or its permutations, as a subsequence.

Scoring of Anomalies

The basic technique assigns an anomaly score to the query pattern (14) 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 [15] address this issue by assuming a statistical distribution for $\bar{f}_{\mathbf{S}}(s)$. Instead of computing the anomaly score for a query pattern as shown in (15), a statistical test is applied to determine if $\bar{f}_S(s)$ is generated by the same distribution.

6.4 Relationship Between Techniques Solving Problem Formulation 1 and Problem Formulation 3

The techniques that solve problem formulation 3 assign an anomaly score to a short pattern s. The basic window based technique discussed in Section 4.2, that handles problem formulation 1, 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 expected frequency in normal sequences, the basic window based technique assigns an anomaly score to a window based on only its expected frequency in normal sequences. To be more precise, the basic window based technique that solves problem formulation 1 essentially uses $\bar{f}_{\mathbf{S}}(s)$ as the anomaly score of each window, while the basic technique that solves problem formulation 3 compares this value with the normalized frequency of occurrence of the window in the test sequence to compute its anomaly score (See (15)).

Nevertheless, techniques for problem formulation 3 can also be used to find certain types of anomalies in the context of problem formulation 1. An alternate cause of 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 solves problem formulation 3. The anomaly scores of all windows are aggregated to obtain an overall anomaly score for the test sequence.

7 Online Anomaly Detection

Several application domains collect sequence data in a streaming fashion [79], 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 [80], [81]. 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 problem formulation 1, 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. In contrast, kernel based techniques measure the similarity of entire test sequence with training sequences, and hence are not suitable for online detection problem.

Techniques that handle the problem formulation 2 can be easily adapted to detect anomalies in an online fashion. In the online setting each successive subsequence of symbols is 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 might require

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

to observe a significantly long portion of the sequence initially before scoring the incoming subsequences.

Problem formulation 3 is more difficult to handle in an online setting, in which the test sequence S is being collected in an online fashion. This setting poses a challenge to techniques discussed in Section 6, because 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

This survey has attempted to provide a structured and comprehensive overview of the research on the problem of anomaly detection for discrete sequences. The survey covers techniques developed independently for different domains and thus opens up the possibility of applying techniques developed within one domain to a completely different setting. In addition, this broad survey makes it possible to envision a rich set of possibilities for extension of the current state of art along different dimensions. Besides serving as a structured and comprehensive review of the existing research, the survey provides a rich set of possible techniques that can be developed by extending the current state of art in different directions. Such possibilities are not obvious without conducting such global study of the problem. 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.

For each problem formulation we have identified distinct groups of techniques that use a specific approach to detect anomalies. Within each of these groups we have provided a basic technique and shown how different existing techniques are variations of the corresponding basic technique. We also provide the motivation behind the different variations and the strengths and weaknesses associated with the different categories of techniques. 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 kernel based techniques for problem formulation 1 (Section 4.1), we observe that by 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, we have also discussed how 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 article has been on discrete sequences, many of the techniques discussed here are also applicable to continuous sequences (or time series). For example, kernel based techniques can be adapted for continuous sequences by using an appropriate similarity/distance kernel, such as *Euclidean Distance* [82], [83] and *Cross Correlation* [84], [85]. 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 [83]. 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 [86]. HMM based techniques can be adapted by using a continuous version of HMM [87]. While some of these adaptations have already been proposed, others are subject of future research.

This article 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, continuous, or a mixed set of observations [88]. Some of the techniques discussed in this article are relevant to such settings, though additional thought still needs to be given to how to handle such complex sequences. For example, kernel and window based techniques for problem formulation 1, as well as the basic technique for problem formulation 2 can be easily extended to multivariate sequences as long as a similarity measure can be developed to compare two multivariate discrete sequences. Such techniques have not been tried yet, but need to be investigated in future.

Acknowledgement

This work was supported by NASA under award NNX08AC36A, NSF Grant CNS-0551551 and NSF Grant IIS-0713227. Access to computing facilities was provided by the Digital Technology Consortium.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection – a survey," *ACM Computing Surveys (To Appear)*, 2008.

[2] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, 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," in *Proceedings of SIAM International Conference on Data Mining*. SIAM, May 2003.

[4] S. Forrest, C. Warrender, and B. Pearlmutter, "Detecting intrusions using system calls: Alternate data models," in *Proceedings of the 1999 IEEE ISRSP*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 133–145.

[5] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998. [Online]. Available: citeseer.ist.psu.edu/hofmeyr98intrusion.html

[6] C. C. Michael and A. Ghosh, "Two state-based approaches to program-based anomaly detection," in *Proceedings of the 16th Annual Computer Security Applications Conference*. IEEE Computer Society, 2000, p. 21.

[7] W. Lee, S. Stolfo, and P. Chan, "Learning patterns from unix process execution traces for intrusion detection," in *Proceedings of the AAAI 97 workshop on AI methods in Fraud and risk management*, 1997.

[8] W. Lee and S. Stolfo, "Data mining approaches for intrusion detection," in *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 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," in *Proceedings of International Conference on Machine Learning and Cybernetics*. IEEE Computer Society, 2002, pp. 381–385.

[11] S. Budalakoti, A. Srivastava, R. Akella, and E. Turkov, "Anomaly detection in large sets of high-dimensional symbol sequences," NASA Ames Research Center, Tech. Rep. NASA TM-2006-214553, 2006.

[12] S. Budalakoti, A. Srivastava, and M. Otey, "Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 37, no. 6, 2007.

[13] P. Sun, S. Chawla, and B. Arunasalam, "Mining for outliers in sequential databases," in *In SIAM International Conference on Data Mining*, 2006.

[14] S. Chakrabarti, S. Sarawagi, and B. Dom, "Mining surprising patterns using temporal description length," in *Proceedings of the 24rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 606–617.

[15] 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.

[16] V. Chandola, V. Mithal, and V. Kumar, "A comparative evaluation of anomaly detection techniques for sequence data," in *Proceedings of International Conference on Data Mining*, 2008.

[17] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Proceedinges of the ISRSP96*, 1996, pp. 120–128. [Online]. Available: citeseer.ist.psu.edu/forrest96sense.html

[18] E. Eskin, W. Lee, and S. Stolfo, "Modeling system call for intrusion detection using dynamic window sizes," in *Proceedings of DISCEX*, 2001. [Online]. Available: citeseer.ist.psu.edu/portnoy01intrusion.html

[19] T. Lane and C. E. Brodley, "Temporal sequence learning and data reduction for anomaly detection," *ACM Transactions on Information Systems and Security*, vol. 2, no. 3, pp. 295–331, 1999.

[20] D. Gusfield, *Algorithms on strings, trees, and sequences: computer science and computational biology*. New York, NY, USA: Cambridge University Press, 1997.

[21] G. Liu, T. K. McDaniel, S. Falkow, and S. Karlin, "Sequence anomalies in the cag7 gene of the helicobacter pylori pathogenicity island," in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96 (12), 1999, pp. 7011–7016.

[22] A. N. Srivastava, "Discovering system health anomalies using data mining techniques," in *Proceedings of 2005 Joint Army Navy NASA Airforce Conference on Propulsion*, 2005.

[23] D. Pavlov and D. Pennock, "A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains," in *Proceedings of Advances in Neural Information Processing*. MIT Press, 2002.

[24] D. Pavlov, "Sequence modeling with mixtures of conditional maximum entropy distributions," in *Proceedings of the Third IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2003, p. 251.

[25] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. ACM Press, 2000, pp. 427–438.

[26] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Proceedings of Applications of Data Mining in Computer Security*. Kluwer Academics, 2002, pp. 78–100.

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

[28] J. Yang and W. Wang, "Cluseq: Efficient and effective sequence clustering." in *Proceddings of International Conference on Data Engineering*, 2003, pp. 101–112.

[29] D. Ron, Y. Singer, and N. Tishby, "The power of amnesia: learning probabilistic automata with variable memory length," *Machine Learning*, vol. 25, no. 2-3, pp. 117–149, 1996.

[30] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Visualization of navigation patterns on a web site using model-based clustering," in *Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 280–284.

[31] P. Smyth, "Clustering sequences with hidden markov models," in *Advances in Neural Information Processing*, vol. 9. MIT Press, 1997.

[32] R. R. Sokal and C. D. Michener, "A statistical method for evaluating systematic relationships," *University of Kansas Scientific Bulletin*, vol. 38, pp. 1409–1438, 1958.

[33] J. W. Hunt and T. G. Szymanski, "A fast algorithm for computing longest common subsequences," *Commun. ACM*, vol. 20, no. 5, pp. 350–353, 1977.

[34] 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," in *SDM*, 2005.

[35] T. Lane and C. E. Brodley, "Sequence matching and learning in anomaly detection for computer security," in *Proceedings of AI Approaches to Fraud Detection and Risk Management*, Fawcett, Haimowitz, Provost, and Stolfo, Eds. AAAI Press, 1997, pp. 43–49.

[36] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of 2000 ACM SIGMOD International Conference on Management of Data*. ACM Press, 2000, pp. 93–104.

[37] D. Endler, "Intrusion detection: Applying machine learning to solaris audit data," in *Proceedings of the 14th Annual Computer Security Applications Conference*. IEEE Computer Society, 1998, p. 268.

[38] H. Debar, M. Dacier, M. Nassehi, and A. Wespi, "Fixed vs. variable-length patterns for detecting suspicious process behavior," in *Proceedings of the 5th European Symposium on Research in Computer Security*. London, UK: Springer-Verlag, 1998, pp. 1–15.

[39] A. K. Ghosh, A. Schwartzbard, and M. Schatz, "Using program behavior profiles for intrusion detection," in *Proceedings of SANS Third Conference and Workshop on Intrusion Detection and Response*, february 1999. [Online]. Available: citeseer.ist.psu.edu/ghosh99learning.html

[40] A. Ghosh, A. Schwartzbard, and M. Schatz, "Learning program behavior profiles for intrusion detection," in *Proceedings of 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, apr 1999, pp. 51–62.

[41] J. B. D. Cabrera, L. Lewis, and R. K. Mehra, "Detection and classification of intrusions and faults using sequences of system calls," *SIGMOD Records*, vol. 30, no. 4, pp. 25–34, 2001.

[42] A. P. Kosoresow and S. A. Hofmeyr, "Intrusion detection via system call traces," *IEEE Software*, vol. 14, no. 5, pp. 35–42, 1997.

[43] D. Dasgupta and F. Nino, "A comparison of negative and positive selection algorithms in novel pattern detection," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, Nashville, TN, 2000, pp. 125–130.

[44] T. Lane and C. E. Brodley, "An application of machine learning to anomaly detection," in *Proceedings of 20th NIST-NCSC National Information Systems Security Conference*, 1997, pp. 366–380.

[45] T. Lane, "Machine learning techniques for the computer security domain of anomaly detection," Ph.D. dissertation, Purdue University, 2000.

[46] D. Dasgupta and N. Majumdar, "Anomaly detection in multidimensional data using negative selection algorithm," in *Proceedings of the IEEE Conference on Evolutionary Computation*, Hawaii, may 2002, pp. 1039–1044.

[47] S. Forrest, P. D'haeseleer, and P. Helman, "An immunological approach to change detection: Algorithms, analysis and implications," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 1996, p. 110.

[48] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proceedings of the 1994 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 1994, p. 202.

[49] S. Forrest and D. Dasgupta, "Novelty detection in time series data using ideas from immunology," in *Proceedings of the 5th International Conference on Intelligence Systems*. Reno, Nevada, USA: IEEE Computer Society, 1996.

[50] S. Forrest, F. Esponda, and P. Helman, "A formal framework for positive and negative detection schemes," in *IEEE Transactions on Systems, Man and Cybernetics, Part B*. IEEE, 2004, pp. 357–373.

[51] A. K. Ghosh, J. Wanken, and F. Charron, "Detecting anomalous and unknown intrusions against programs," in *Proceedings of the 14th Annual Computer Security Applications Conference*. IEEE Computer Society, 1998, p. 259.

[52] M. Wang, C. Zhang, and J. Yu, "Native api based windows anomaly intrusion detection method using svm," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 1. Washington, DC, USA: IEEE Computer Society, 2006, pp. 514–519.

[53] S. Tian, S. Mu, and C. Yin, "Sequence-similarity kernels for svms to detect anomalies in system calls," *Neurocomputing*, vol. 70, no. 4-6, pp. 859–866, 2007.

[54] X. Li, J. Han, S. Kim, and H. Gonzalez, "Roam: Rule- and motif-based anomaly detection in massive moving object data sets," in *Proceedings of 7th SIAM International Conference on Data Mining*, 2007.

[55] N. Ye, "A markov chain model of temporal behavior for anomaly detection," in *Proceedings of the 5th Annual IEEE Information Assurance Workshop*. IEEE, 2004.

[56] C. Marceau, "Characterizing the behavior of a program using multiple-length n-grams," in *Proceedings of the 2000 workshop on New Security Paradigms*. New York, NY, USA: ACM Press, 2000, pp. 101–110.

[57] E. Eskin, W. N. Grundy, and Y. Singer, "Protein family classification using sparse markov transducers," in *Proceedings of ISMB08*. AAAI Press, 2000, pp. 134–145.

[58] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the 12th International Conference on Machine Learning*, A. Prieditis and S. Russell, Eds. Tahoe City, CA: Morgan Kaufmann, jul 1995, pp. 115–123.

[59] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[60] 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.

[61] X. Zhang, P. Fan, and Z. Zhu, "A new anomaly detection method based on hierarchical hmm," in *Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2003, pp. 249–252.

[62] 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.

[63] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains," in *Annals of Mathematical Statistics*, vol. 41(1), 1970, pp. 164–171.

[64] J. Forney, G.D., "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

[65] 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.

[66] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *Proceedings of the Fifth IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 226–233.

[67] J. Lin, E. Keogh, A. Fu, and H. V. Herle, "Approximations to magic: Finding unusual medical time series," in *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 329–334.

[68] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Min. Knowl. Discov.*, vol. 15, no. 2, pp. 107–144, 2007.

[69] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," in *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2004, pp. 206–215.

[70] L. Wei, N. Kumar, V. Lolla, E. J. Keogh, S. Lonardi, and C. Ratanamahatana, "Assumption-free anomaly detection in time series," in *Proceedings of the 17th international conference on Scientific and statistical database management*. Berkeley, CA, US: Lawrence Berkeley Laboratory, 2005, pp. 237–240.

[71] L. Wei, E. Keogh, and X. Xi, "Saxually explicit images: Finding unusual shapes," in *Proceedings of the Sixth International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 711–720.

[72] Y. Bu, T.-W. Leung, A. Fu, E. Keogh, J. Pei, and S. Meshkin, "Wat: Finding top-k discords in time series database," in *Proceedings of 7th SIAM International Conference on Data Mining*, 2007.

[73] A. W.-C. Fu, O. T.-W. Leung, E. J. Keogh, and J. Lin, "Finding time series discords based on haar transform," in *Proceeding of the 2nd International Conference on Advanced Data Mining and Applications*. Springer Verlag, 2006, pp. 31–41.

[74] M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula, and D. Panda, "Towards nic-based intrusion detection," in *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2003, pp. 723–728.

[75] E. Keogh, S. Lonardi, and B. Y. chi' Chiu, "Finding surprising patterns in a time series database in linear time and space," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2002, pp. 550–556.

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

[77] R. Gwadera, M. Atallah, and W. Szpankowskii, "Detection of significant sets of episodes in event sequences," in *Proceedings of the 4th IEEE International Conference on Data Mining*, 2004, pp. 3–10.

[78] R. Gwadera, M. J. Atallah, and W. Szpankowski, "Markov models for identification of significant episodes," in *Proceedings of 5th SIAM International Conference on Data Mining*, 2005.

[79] R. A. Maxion and K. M. C. Tan, "Benchmarking anomaly-based detection systems," in *Proceedings of the 2000 International Conference on Dependable Systems and Networks (formerly FTCS-30 and DCCA-8)*. Washington, DC, USA: IEEE Computer Society, 2000, pp. 623–630.

[80] A. Pawling, P. Yan, J. Candia, T. Schoenharl, and G. Madey, "Anomaly detection in streaming sensor data," in *Intelligent Techniques for Warehousing and Mining Sensor Network Data*. IGI Global, 2008.

[81] D. Pokrajac, A. Lazarevic, and L. J. Latecki, "Incremental local outlier detection for data streams," in *Proceedings of IEEE Symposium on Computational Intelligence and Data Mining*, 2007.

[82] G. K. Palshikar, "Distance-based outliers in sequences," *Lecture Notes in Computer Science*, vol. 3816, pp. 547–552, 2005.

[83] D. Yankov, E. J. Keogh, and U. Rebbapragada, "Disk aware discord discovery: Finding unusual time series in terabyte sized datasets," in *Proceedings of International Conference on Data Mining*, 2007, pp. 381–390.

[84] 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 Society*, vol. 369, no. 2, pp. 677–696, 2006.

[85] U. Rebbapragada, P. Protopapas, C. E. Brodley, and C. Alcock, "Finding anomalous periodic time series," 2008, submitted to the Machine Learning Journal.

[86] J. Ma and S. Perkins, "Online novelty detection on temporal sequences," in *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2003, pp. 613–618.

[87] Z. Liu, J. X. Yu, L. Chen, and D. Wu, "Detection of shape anomalies: A probabilistic approach using hidden markov models," in *IEEE 24th International Conference on Data Engineering*, April 2008, pp. 1325–1327.

[88] H. Cheng, P.-N. Tan, C. Potter, and S. Klooster, "Detection and characterization of anomalies in multivariate time series," in *Proceedings of the ninth SIAM International Conference on Data Mining*, 2009.