To the Graduate Council:
I am submitting herewith a dissertation written by Haoran Niu entitled "Anomaly Detection on Complex Health Information Technology Systems." I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Michael A. Langston, Major Professor

We have read this dissertation
and recommend its acceptance:

Olufemi A. Omitaomu

David J. Icove

Catherine D. Schuman

Accepted for the Council:

Vice Provost and Dean of the Graduate School

To the Graduate Council:
I am submitting herewith a dissertation written by Haoran Niu entitled "Anomaly Detection on Complex Health Information Technology Systems." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Michael A. Langston, Major Professor

We have read this dissertation
and recommend its acceptance:

Olufemi A. Omitaomu

David J. Icove

Catherine D. Schuman

Accepted for the Council:

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Anomaly Detection on Complex Health Information Technology Systems

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Haoran Niu

August 2023

© by Haoran Niu, 2023
All Rights Reserved.

To my parents, who have always been my guiding light and source of unwavering love and support.

Acknowledgements

I would like to express my deep gratitude to my PhD advisor, Dr. Michael A. Langston, for his invaluable guidance and support. His mentorship and scientific insights were instrumental in shaping my research and academic career. I also extend my heartfelt gratitude to my mentor at Oak Ridge National Laboratory, Dr. Olufemi A. Omitaomu, for his support and guidance throughout my doctoral studies, providing me with invaluable training opportunities and challenging me with exciting projects.

I would also like to express my appreciation to my dissertation committee members, Dr. Olufemi A. Omitaomu, Dr. David J. Icove and Dr. Catherine D. Schuman, for their insightful comments and constructive feedback, which greatly contributed to the quality of my work.

I am immensely grateful to my parents and family for their unwavering support and encouragement throughout my life. Their love and guidance have been a constant source of strength and motivation for me, and I owe them a debt of gratitude for their unwavering belief in my abilities. I am truly blessed to have them in my life and could not have accomplished any of this without their support and encouragement.

Finally, I would like to express my gratitude to my colleagues and friends for their support, encouragement, and motivation. Their help and companionship have been a source of inspiration and comfort during my doctoral studies.

Abstract

While modern complex computer systems provide enormous benefits to our daily lives, the increasing complexity of these large-scale systems also makes them more susceptible to unexpected software malfunctions and malicious attacks. This is especially true for Health Information Technology (HIT), which has revolutionized healthcare delivery by making it more efficient, effective, and accessible. Nevertheless, the widespread adoption of HIT has introduced new challenges related to ensuring system reliability and security. As a result, the development of novel algorithms and frameworks to detect anomalies in such systems has become increasingly important for enhancing patient safety and improving the efficiency and effectiveness of healthcare services.

This dissertation presents innovative approaches for anomaly detection in HIT systems using Electronic Health Records (EHR), addressing the complexity of HIT and the need for patient data privacy and security. The first approach is an event sequence and subsequence anomaly detection algorithm that utilizes network-based representations, considers higher-order dependencies, and incorporates salient information of sequences for discrimination. By monitoring changes in the graph structure after removing test sequences, the algorithm effectively identifies anomalies and suggests plausible transitions for detected anomalous subsequences. Leveraging the cutting-edge natural language processing model Bidirectional Encoder Representations from Transformers (BERT), the second approach, named EHR-BERT, learns and identifies patterns in sequences bidirectionally, demonstrating

improved accuracy and reduced false negatives through extensive evaluations. The third approach presents a comprehensive framework for detecting system-level abnormal events using high-dimensional system volume data. This framework employs machine learning models to identify outliers through a voting machine strategy, constructs weighted graphs representing the correlation of co-occurring outliers, and utilizes spectral graph theory and the paraclique algorithm to capture clusters with high correlations of co-occurring anomalies.

Collectively, the proposed approaches advance state-of-the-art in anomaly detection in HIT systems and contribute to the development of advanced algorithms that can be applied in various clinical domains, ultimately improving patient outcomes and enhancing the reliability and efficiency of healthcare services. Additionally, the proposed approaches have the potential to be applied to other complex computer systems beyond HIT, where anomaly detection is critical for maintaining system reliability and security.

Table of Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Anomaly Detection in HIT Systems 2

1.3 Challenges 3

1.4 Definitions 6

1.5 Outline 8

2 Discrete Sequence Anomaly Detection using Higher-order Tensor Networks **9**

2.1 Introduction 9

2.2 Related Work 13

2.2.1 Point-Based Anomaly Detection Algorithm 13

2.2.2 Markov-Based Anomaly Detection Algorithm 14

2.3 Methods 15

2.3.1 Representing Event Sequence Data in Higher-Order Tensor Networks 15

2.3.2 Sequence Anomaly Detection Based on Higher-Order Tensor Networks 20

2.3.3 Contiguous Subsequence Anomaly Detection Based on Higher-Order Tensor Networks 23

2.3.4 Spark-Based Implementation 25

2.4 Evaluations and Discussions . . . 27
2.4.1 Baseline Methods . . . 27
2.4.2 Synthetic EHR Dataset . . . 29
2.4.3 Real-World EHR Dataset . . . 34
2.4.4 Performance Scalability . . . 40
2.4.5 Importance of Anomaly Detection for HIT Systems . . . 40
2.5 Conclusion . . . 43

3 EHR-BERT: A BERT-based Model for Effective Anomaly Detection in Electronic Health Records 44
3.1 Introduction . . . 44
3.2 Related Work . . . 48
3.3 Design of EHR-BERT . . . 50
3.3.1 Overview . . . 50
3.3.2 Framework . . . 51
3.3.3 Sequential Masked Token Prediction . . . 58
3.4 Evaluations and Discussions . . . 61
3.4.1 Datasets . . . 62
3.4.2 Baseline Methods . . . 64
3.4.3 Experimental Settings . . . 66
3.4.4 Experimental Results . . . 68
3.4.5 Discussions . . . 70
3.5 Conclusion . . . 73

4 System-Level Anomaly Detection in HIT Systems Using High-Dimensional Volume: A Framework Based on Machine Learning and Graph Algorithms 74
4.1 Introduction . . . 74
4.2 Related Work . . . 77
4.2.1 Statistical Methods . . . 77

4.2.2 Machine Learning-Based Approaches 78
4.3 Framework Design 80
4.3.1 Overview 80
4.3.2 Data Preprocessing 83
4.3.3 Voting Machine 84
4.3.4 Graph Anomaly Detection 90
4.4 Evaluations 95
4.4.1 Datasets 95
4.4.2 Evaluation on Voting Machine 96
4.4.3 Evaluation on Graph Anomaly Detection 101
4.5 Summary 108

5 Conclusions **110**

Vita **128**

List of Tables

2.1 Detection results for synthetic dataset: m means miss (i.e., not detected), and h means hit (i.e., detected). 32

2.2 Abnormal transitions and recommended transitions for synthetic EHR dataset by CSAD-HOTN. 32

2.3 Test recall for each method on synthetic data. 35

3.1 Detail of EHR datasets. 63

3.2 Detail of EHR datasets in evaluations. The columns "# of anomaly" show the number of randomly selected anomalies under each threshold. 65

4.1 Evaluation Results for Voting Machine Using Injected Anomalies. Precision (P), Recall (R), and F-score (F) are reported. The best performing method for each metric (column) is highlighted. 100

4.2 List of medical facilities and corresponding dates for system updates conducted during 2018. 102

4.3 Paracliques detected using lookback period of 2 weeks, MaxDeferTime of 2 days, and a glom factor of 1 and 2. Differences are marked in red. 107

List of Figures

1.1 Anomaly Detection Framework for HIT systems. 4

2.1 Representing a sequence in a HOTN. 12

2.2 Example of BuildHOTN with transition probability and duration as the edge weight. The process order is labeled with circled numbers. . 19

2.3 Diagram of anomaly detection based on HOTN. 21

2.4 Diagram of CSAD-HON. 24

2.5 Building synthetic EHR data. (a) Process of building normal data in which one of the three events is selected in each step and five events in each sequence data. Three rules are specified using red-, blue-, and green-dashed arrows. (b) Five synthetic anomalies with abnormal transitions are also labeled using their corresponding colors. 30

2.6 Performance comparison between AD-HOTN and baselines for large synthetic datasets. The Precision for each method is 1. 35

2.7 Comparison of anomaly detection accuracy between AD-HOTN and the baselines with four different anomaly injection ratios and a real-world dataset. 37

2.8 Three case studies using AD-HOTN on real-world data with a 10% injection ratio. In each case study, the transitions have the same start and end events but different dependencies. By considering the previous start events, AD-HOTN can identify those anomalies that were normal with different dependencies (some of them are labeled using red circles). 39

2.9 Performance comparison across different data sizes and approaches. . 41

3.1 The Overview of EHR-BERT. 52
3.2 BERT-based classifier. 55
3.3 Subfigure (a) illustrates how RMTP can introduce false negatives. On the other hand, subfigure (b) shows the SMTP process. 60
3.4 Figure a-c show the histogram of sequence length for three datasets. Figure d-f present the histogram of sequence pattern frequencies for three data sets. 63
3.5 Comparison evaluation results on EHR-BERT with five baselines on EHR dataset from three medical domains. 69

4.1 Framework of volume anomaly detection. 81
4.2 Diagram showing the voting machine algorithm for combining results from multiple anomaly detection models. 85
4.3 Depictions of four common volume deviations in HIT systems. Each dot denotes a data point. 97
4.4 The Detection Rate of the proposed framework on different settings of MaxDeferTime, Lookback Period, and glom factor. 104

Chapter 1

Introduction

1.1 Motivation

Health Information Technology (HIT) has brought a revolutionary change to healthcare delivery by making it more efficient, effective, and accessible. Over the past decade, healthcare in the U.S. has advanced with the use of HIT such that nearly all hospitals and physician offices transitioned from paper-based medical records to electronic health records (EHRs) [102]. The widespread adoption of HIT has led to the generation of massive amounts of EHR log data, which documents significant events and system states at various stages of operation, and provides valuable information on patient care and clinical processes. But, the increasing complexity and scale of HIT systems, combined with the growing adoption of EHRs, has introduced new challenges related to system reliability and security. Anomaly detection has become an essential task to maintain the integrity and stability of HIT systems, enabling early detection and response to system-level hazards [79].

The security and safety issues associated with HIT systems are becoming increasingly apparent. According to a recent report [47], more than 40 million patient records were compromised in incidents reported to the federal government in 2021. In addition, HIT system failures or operational issues can cause harm to data subjects,

healthcare providers, and other stakeholders. There have been instances where system failures have led to the loss of patients' lives [3]. These incidents highlight the urgent need for robust security measures in HIT systems.

Given these challenges, online monitoring and anomaly detection of EHRs are crucial for improving clinical decision-making and patient care processes, as well as ensuring the security and reliability of HIT systems. This dissertation addresses the challenges of anomaly detection in HIT systems by developing novel algorithms and frameworks that can effectively identify abnormal events and system-level anomalies.

1.2 Anomaly Detection in HIT Systems

Anomaly detection systems are designed to identify events that appear to deviate from normal system behavior. Alerts are issued when the behavior largely deviates from the norm. In this dissertation, the work focus on two common types of EHR data in HIT systems: discrete sequence data and time-series point data.

A discrete sequence is an ordered series of symbols which can be characters, numbers, or words. Symbolic sequences typically do not have meaningful values for individual data points, but they provide valuable information when considered in the context of other symbols in the sequence [115]. In this dissertation, the terms "sequence" and "discrete sequence" are used interchangeably. The sequence data extracted from EHR logs can capture patient care status and significant events, and supports further analysis. Therefore, online monitoring and anomaly detection of EHR sequential data are crucial for improving clinical decision-making and patient care processes. The sequence of events in a HIT system provides insight into the system's operations, and anomalous behavior can be detected by identifying deviations from normal data patterns.

Time-series point data consists of a continuous sequence of data points that are measured at discrete time points. Our study focuses specifically on traffic data and volume anomalies, which can indicate a broad range of operational issues such as

malicious activities like denial-of-service attacks and worm spreads, unusual events, misconfiguration, equipment failures leading to traffic shifts, or abnormal user or customer behaviors. Accurately detecting and responding to such anomalies is critical for maintaining the security and reliability of HIT systems, ultimately improving patient care and decision-making processes.

Figure 1.1 presents the comprehensive framework proposed in this dissertation. The first step of the framework involves the extraction and analysis of EHR data from HIT systems, using both EHR sequence data and time-series volume data of facilities. Our proposed anomaly detection models are then applied to flag any anomalies for domain experts' review. These models are further updated based on feedback from the experts.

To identify sequence anomalies, two novel algorithms are proposed based on higher-order tensor networks and natural language models, specifically Bidirectional Encoder Representations from Transformers (BERT). To identify system-level abnormal events in HIT systems, a novel framework is designed for detecting network-wide high-dimensional volume anomalies. This framework utilizes cutting-edge machine learning models and advanced graph algorithms.

1.3 Challenges

The detection of anomalies in HIT systems presents a range of challenges for quantitative approaches. While there is a growing need for the development of anomaly detection systems in HIT, the complexities of these problems pose significant hurdles. Recent studies have highlighted a number of common challenges encountered when developing alert systems in HIT [79, 78, 81], including the following:

- **Situational dependence:** Anomaly detection problems in HIT are highly dependent on the specific clinical domain, making it challenging to develop a standardized approach. The selection of variables, parameters, and outputs must be carefully tailored to the specific problem under consideration.

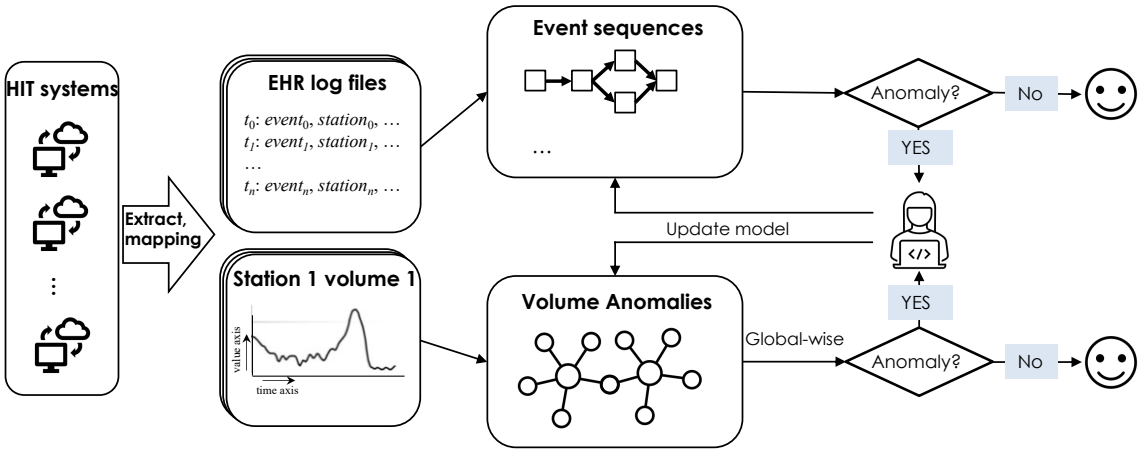


Figure 1.1: Anomaly Detection Framework for HIT systems.

- **Higher degrees of freedom:** Anomaly detection requires an understanding of normal behavior, which is complicated by the presence of mixed normal and abnormal data without labeling. Additionally, there are cyclical, seasonal, and irregular trends present in the data, which must be considered for effective hazard detection frameworks.

- **Class imbalance:** Anomalies are often rare events in HIT systems, resulting in class imbalance in the training data. Appropriate sampling techniques and evaluation metrics must be used to handle the class imbalance problem.

- **Unstructured data:** EHRs often involve large volumes of unstructured free text data, posing a challenge for anomaly detection algorithms that typically require structured data.

- **Lack of labeled data:** Anomaly detection algorithms require labeled data to train the model, but labeled data is often scarce in HIT systems. The process of labeling data is also time-consuming and expensive.

- **High dimensionality of data:** HIT systems generate high-dimensional event and volume data, with intricacies in event relationships that can easily number in the thousands.

- **Concept drift:** HIT systems are dynamic and change over time, and the patterns of normal behavior may change as well. This challenge requires the development of algorithms that can adapt to changes in the data distribution and concept drift.

- **Interpretability:** Anomaly detection algorithms may generate false positives or false negatives, making it important to provide interpretable results to domain experts. This challenge requires the development of algorithms that can provide interpretable results and explainable anomalies.

- **Privacy and security:** HIT data contains sensitive patient information, making it important to develop anomaly detection algorithms that preserve the privacy and security of the data. This challenge requires the development of secure and privacy-preserving algorithms that can operate on encrypted or anonymized data.

To overcome the challenges of anomaly detection in HIT, new algorithms and frameworks must be developed to handle the unique features of HIT data while preserving patient privacy and security. These advanced techniques from machine learning, data mining, and statistics must provide interpretable and explainable models to provide insights for domain experts. This will ultimately improve patient safety and enhance the reliability and efficiency of healthcare services.

1.4 Definitions

This section presents notations and basic concepts used in this dissertation. It also clarifies five problem formulations for sequence anomaly detection, contiguous subsequence-based anomaly detection, normal subsequences recommendation, time-series anomaly detection, and system-level time-series anomaly detection.

Definition 1. In this dissertation, a discrete or continuous *sequence*, is defined as a finite series of events or data points in chronological order accompanied by additional information [20], such as a timestamp. Sets of sequences will be represented by bold uppercase letters (e.g., $\mathbf{S}$, $\mathbf{T}$).

Definition 2. An *event* is defined as a symbol, which is the smallest meaningful element in a sequence. An event within a sequence $\mathbf{s}$, will be denoted as e [20].

Definition 3. A *subsequence* is defined as a smaller sequence that can be derived from the original sequence by deleting some events without changing the order of the remaining sequence [38]. In chapter 2 of the dissertation, a subsequence refers to a *contiguous subsequence* of a sequence. Formally, a subsequence of a sequence, $\mathbf{s}$, is

denoted as $s_{i:j}$, which starts from the i^{th} location and ends at the j^{th} location of the sequence, $\mathbf{s}$.

Definition 4. A *transition*, which is the smallest subsequence in a sequence, is defined as a movement between two contiguous events within a sequence.

Definition 5. A *tensor* can be represented as a multidimensional array and is a higher-way generalization of a vector and a matrix. In chapter 2, for a network consisting of N nodes, transitions are represented as 4D tensors, $\mathscr{T} \in \mathbb{R}^{N \times N \times P \times D}$ (source, target, transition probability, and transition duration). The framework used to represent information in the form of tensors is called a *tensor network*, which will be denoted as $\mathscr{N}$.

Problem Definition 1. Discrete sequence anomaly detection. Given a set of n sequences, $\mathbf{S} = \{\mathbf{s}_1,\mathbf{s}_2,...,\mathbf{s}_n\}$, find anomalous sequences in $\mathbf{S}$ with respect to the rest of $\mathbf{S}$.

Problem Definition 2. Subsequence-based anomaly detection. Detect contiguous subsequences in a detected abnormal sequence, $\mathbf{s}$, that are anomalous with respect to the rest of $\mathbf{s}$.

Problem Definition 3. Normal subsequences recommended for detecting abnormal subsequences. Based on the historical normal data patterns in a given set of sequences, $\mathbf{S}$, detected abnormal subsequences, $s_{i:j}$, are replaced by recommend subsequences, $\overline{s}_{i:j}$.

Problem Definition 4. Time-series anomaly detection. Given a set of time-series points, $V = \{v_1, v_2, ..., v_n\}$, obtained from a consistent type of measurement/observation, the goal is to identify the abnormal data points v that do not follow the common collective trend, seasonal, or cyclic patterns of the entire data and are significantly distinct from the rest of the data V.

Problem Definition 5. System-level time-series anomaly detection (also known as network-wide anomaly detection). Given a network comprising multiple nodes, where each node generates univariate or multivariate time-series data, the

goal is to identify a cluster of nodes that exhibit behaviors significantly different from the normal operational patterns observed in the majority of the network nodes.

1.5 Outline

The dissertation comprises three works, organized as follows: In chapter 2, event sequence and subsequence anomaly detection algorithms are proposed that use network-based representations of interactions, account for higher-order dependencies, and incorporate event duration for accurate discrimination. Chapter 3 presents EHR-BERT, a state-of-the-art framework for EHR sequence anomaly detection leveraging Sequential Masked Token Prediction. Chapter 4 introduces a comprehensive framework for detecting network-wide volume anomalies using machine learning models and graph algorithms. These works address significant challenges in HIT anomaly detection and offer novel solutions to improve patient safety and enhance healthcare services.

Chapter 2

Discrete Sequence Anomaly Detection using Higher-order Tensor Networks *

2.1 Introduction

Detecting anomalies in sequence data has become a crucial task for many real-world applications, as sequences occur naturally in various domains. Anomaly detection methods aim to identify irregular patterns in the data by characterizing the underlying sequence structure [20]. Sequence data are pervasive in bioinformatics [4], healthcare [75, 32], cybersecurity [85, 84], and financial fraud detection [64], among other fields [94]. With the advent of big data, efficient identification of anomalies or implausible patterns in sequence data has become more critical than ever.

Healthcare sequential data anomaly detection techniques can help medical professionals make informed decisions and prevent potential hazards by detecting and presenting irregular patterns in a useful format. Although most HIT systems do not generate EHR logs, these logs can be obtained through reverse engineering to provide valuable insights into the state of the healthcare process. The sequence data extracted from EHR logs can capture patient care status and significant events and

*This chapter is featured in a manuscript [78] authored by Haoran Niu, Olufemi A. Omitaomu, and Michael A. Langston, et al., which has been published in the *Journal of Biomedical Informatics*.

support further analysis. Therefore, online monitoring and anomaly detection of EHR sequential data are crucial for improving clinical decision-making and patient care processes. The sequence of events in a HIT system provides insight into the system's operations, and anomalous behavior can be detected by identifying deviations from normal data patterns.

Plenty of research has been dedicated to sequence-based anomaly detection. Point-based techniques that compute the similarity between sequences and present an anomaly score (e.g., k^{th} nearest neighbor (KNN) [87], k-means [32], and clustering methods [21, 73, 5]) are well studied in the time-series anomaly detection domain. Feature-embedding methods (e.g., word2vec [25], sequence graph transform (SGT) [88], and recurrent neural networks [36, 107]) are required to provide machine-interpretable representations for sequences before adapting point-based techniques in event-sequence anomaly detection. Markov-based methods, in which all transitions between events are analyzed, are widely used for sequence anomaly detection. Fixed Markovian methods (e.g., extended finite state automata [71], suffix trees [66]) use a fixed history of sequences to calculate the conditional probability of an event. Nevertheless, these methods lack the flexibility to provide a reliable estimate of transition probability and require a huge amount of space to store all transitions. Variable Markov-based approaches are proposed to address these challenges by storing variable length histories of sequences [98, 111, 77]. Other popular methods include the hidden Markov model–based approaches [23, 49, 100] and neural network–based methods [30, 15, 45].

Nonetheless, many existing anomaly detection methods for high-dimensional sequence data only consider traditional transition probability and ignore additional valuable information such as the duration between prescription dosages in medical records, event location in credit card fraud detection, and multiple events in a cyber network. Incorporating such additional information can provide a detailed description of each event, leading to a more accurate characterization of individual events and better anomaly detection. To address this, this chapter proposes a higher-order

tensor network (HOTN) that captures both traditional transition probability and salient transition information under high-order dependencies. By considering the current event along with previous events, HOTN can efficiently build accurate network representations and capture the potential higher-order information that may have been missed by first-order network representations.

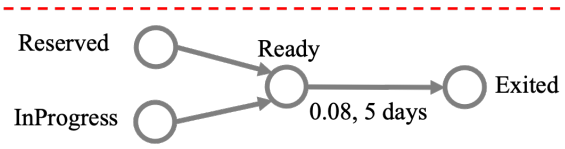
Motivating Example: Figure 2.1 shows an example from our real-world dataset of a radiology test order from an ischemic heart disease cohort. The table in Fig. 2.1a shows that the transition probability from `Ready` to `Exited` is 0.08, and the average transition duration is 5 days. In contrast, more accurate information could be leveraged by considering the previous events (high-order dependencies): if the previous event of `Ready` is `Reserved`, then the transition probability is 0.05 with 8 days as the average duration to `Exited`; if the previous event of `Ready` is `InProgress`, then the transition probability is 0.27 with 42 seconds as the average duration to `Exited`. By splitting the `Ready` node into two higher-order nodes, the network representation is more accurate and more insightful.

This additional information can also be used to design triggers (e.g., abnormal duration between events, potential drug overdoses, drug interactions in patients) for monitored systems which still remain a challenge. To integrate this additional information into a knowledge discovery process, we propose the Anomaly Detection algorithm based on Higher-Order Tensor Networks (AD-HOTN), an unsupervised method that uses a HOTN representation to fully characterize data and detect anomalous sequences in high-dimensional event sequence data. We also propose the Contiguous Subsequence Anomaly Detection algorithm based on Higher-Order Tensor Networks (CSAD-HOTN) to locate the abnormal subsequences within the detected anomalous sequences. The main contributions of this work are summarized as follows:

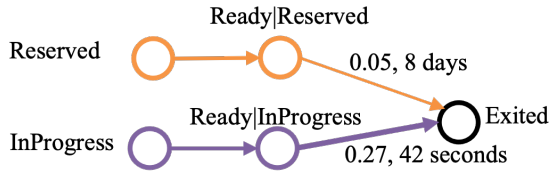
- A network representation method, HOTN, is introduced to fully capture the potential information of high-dimensional event sequence data. We demonstrate

ID	EHR Data	Date Time
0	Reserved	2017-5-12 10:12:30
0	Ready	2017-5-18 15:05:03
0	Exited	2017-5-22 11:12:30
1	InProgress	2017-5-12 10:12:30
…	…	...

(a) Raw EHR data



(b) First-order network



(c) Higher-order network

Figure 2.1: Representing a sequence in a HOTN.

that HOTN is efficient in scaling to higher orders because the order grows only when necessary.

- A graph-based novel framework, AD-HOTN, is proposed to explicitly identify anomalies in data sequences. The method can detect abnormal pathways and anomalies in salient information. This framework is also implemented on the Apache Spark platform to increase its scalability to big sequence data applications.

- A subsequence anomaly detection algorithm, CSAD-HOTN, is proposed to accurately locate the abnormal segments within the anomalous sequences.

- The proposed methods are demonstrated to outperform the state-of-art baseline methods by using synthetic sequence data and real-world EHRs.

The remainder of this chapter is organized as follows: Section 2.2 briefly summarizes recent related work. Section 2.3 presents the proposed algorithms and describes the implementation of the algorithms via Apache Spark. Section 2.4 describes the evaluation of the proposed algorithms against the state-of-the-art baseline methods using synthetic and real-world data. Section 2.5 concludes the chapter with a brief summary.

2.2 Related Work

Anomaly detection methods for event sequences have been studied extensively over the last decade. The following two sections briefly review the related methods.

2.2.1 Point-Based Anomaly Detection Algorithm

Many studies have been conducted on event-sequence anomaly detection using point-based methods [21, 87, 73, 5]. These methods calculate the similarity between a test sequence and training sequences and use the dissimilarity scores as anomaly scores.

Chandola et al. [21] proposed a kNN-based method that calculates the dissimilarity to its k^{th} nearest neighbor. K-means, N-gram, and other clustering-based methods have been discussed in the literature [120, 19, 16]. Still, with those methods, the effectiveness of the detection process can be affected by the similarity metrics. On the other hand, many proposed feature-embedding methods can largely improve the performance of the traditional methods [25, 107, 36, 88]. Ranjan et al. [88] proposed an effective embedding method, SGT, which can extract a varying amount of short- to long-term dependencies with high accuracy when used with traditional point-based approaches.

2.2.2 Markov-Based Anomaly Detection Algorithm

Markov-based methods are popular sequence anomaly detection approaches that use the short-term property of sequences by building a probabilistic model from the conditional probability for events. Boldt et al. [10] applied Markov models on streaming sequence data by considering different temporal resolutions, but they used fixed-order Markov models. The fixed-Markovian approach has limitations because the methods use a fixed history of a sequence for building the probabilistic model, and the methods always require a huge amount of storage [71, 66]. Variable Markov-based methods [75, 77, 111] are proposed to address these limitations. The detector based on probabilistic suffix tree (PST) was proposed by Sun et al. [98] for representing a variable-length history of a sequence as a compact tree representation. The study by Niu et al. [77] took dynamic anomalies into consideration using variable-order network representations. Xu et al. [111] proposed a sequence anomaly detection algorithm for detecting new events of data by building an efficient variable Markov-based network representation that generated higher-order dependencies on demand. Nevertheless, these methods focus only on the events of the sequences and ignore the additional information (e.g., event duration) associated with the events.

Jin et al. [48] demonstrated the edge-labeled graph readability query problem in which edges are associated with labels to denote different types of relationships between vertices. This method also demonstrates the importance of label/salient information in graph problems. Xie et al. [110] discussed the advantage of using tensor graph models for traffic-monitoring data. Their model adds time between traffic data as a tensor to their graph model, which can provide an accurate recovery model for the traffic data. Yet, the model does not consider the higher-order dependencies in the data.

2.3 Methods

This section presents the proposed algorithms, which aim to efficiently detect anomalies in high-dimensional sequence data. Firstly, the HOTN representation is introduced for high-dimensional sequence data. Secondly, the AD-HOTN event sequence anomaly detection algorithm is proposed, which detects anomalies based on abnormal transitions and implausible transition durations. Lastly, the CSAD-HOTN algorithm is presented, which identifies abnormal contiguous subsequences from the detected anomalies. By combining these approaches, anomalous sequences with implausible pathways or transition durations are flagged, abnormal subsequences within the flagged sequences are detected, and plausible transitions and durations for anomalies are proposed. These algorithms have the potential to provide valuable insights into complex systems and support decision-making processes.

2.3.1 Representing Event Sequence Data in Higher-Order Tensor Networks

The higher-order network (HON) representation [112] was proposed to ensure accurate representation of the underlying phenomena in complex systems. HON representations eliminate the potential information loss introduced by a conventional

Markov network representation [112]. On the other hand, the HON representation only contains transition probabilities or frequencies without accounting for transition duration or other pertinent information in the data. To address this limitation, the HOTN representation was proposed for multidimensional event sequence data. The approach can preserve more information in the data by using transition probabilities and average transition durations as an edge weight. This method also scales to more variables.

Generally, a network is a graph represented as $G = (V, E, \psi)$, where V represents vertices or nodes, and E represents edges or links (i.e., connections between pairs of nodes). If the event in a network at time t is denoted as $e_t(n_t \in V)$, then the transition probability of movement from node n_t to node n_{t+1} is proportional to the count between the two nodes, $C(n_t \to n_{t+1})$, which is given as

$$P(e_{t+1} = n_{t+1} | e_t = n_t) = \frac{C(n_t \to n_{t+1})}{\sum_j C(n_t \to n_j)}, \tag{2.1}$$

where $n_j \in V$. The probability of a movement between two nodes is part of the edge weight of the network. An edge weight of a transition can also preserve other information (e.g., average transition duration). If the timestamp of node n_t is ts_t, and the timestamp of node n_{t+1} is ts_{t+1}, then the transition duration $D_s(n_t \to n_{t+1})$, which is calculated as $ts_{t+1} - ts_t$, represents the time duration of movement from nodes n_t to n_{t+1} in sequence $\mathbf{s}$. In a sequence set, $\mathbf{S}$, the average duration (AvgD) between nodes n_t and n_{t+1}, where the event in a network at time t is denoted as $e_t(n_t \in V)$, is given as

$$AvgD(e_{t+1} = n_{t+1} | e_t = n_t) = \frac{\sum_{s \in \mathbf{S}} D_s(n_t \to n_{t+1})}{C(n_t \to n_{t+1})}. \tag{2.2}$$

For the tensor network representation, a vector is assumed to contain the transition probability, Eq. (2.1), and average duration, Eq. (2.2), as the edge weight. The vector can account for more variables.

The conventional network representation assumes that the next event depends only on the current event, Eq. (2.2), thereby ignoring that previous events can also influence the next event [112]. Consequently, the Markovian assumption has been well studied in network representation [71, 66, 91], in which each node can also represent a sequence of nodes. A node that contains its contiguous historical nodes is defined as a higher-order node, and the node preserves higher-order dependencies. A k^{th}-order node is denoted as $n_t|n_{t-1}...n_{t-k-1}$, where the previous movements of node n_t are $n_{t-k-1} \rightarrow ... \rightarrow n_{t-1}$. The more previous events contained in a node, the higher its order dependencies.

Hence, the probability and average duration of moving to the next node, n_{t+1}, from a higher-order node, $n_t|n_{t-1}...n_{t-k}$, are given as

$$\begin{aligned} &P(e_{t+1} = n_{t+1}|e_t = (n_t|n_{t-1}...n_{t-k})) \\ &= \frac{C(n_t|n_{t-1}...n_{t-k} \rightarrow n_{t+1})}{\sum_\ell C(n_t|n_{t-1}...n_{t-k} \rightarrow n_\ell)}, \end{aligned} \tag{2.3}$$

where $n_\ell \in V$ and $k < t$,

$$\begin{aligned} &AvgD(e_{t+1} = n_{t+1}|e_t = (n_t|n_{t-1}...n_{t-k})) \\ &= \frac{\sum_{s \in \mathbf{S}} D_s((n_t|n_{t-1}...n_{t-k}) \rightarrow n_{t+1})}{C((n_t|n_{t-1}...n_{t-k}) \rightarrow n_{t+1})}. \end{aligned} \tag{2.4}$$

Xu et al. [112, 111] proposed BuildHON+ for an HON representation of sequence data. A key step in BuildHON+ is to determine whether increasing a node's order from a lower-order transition to a higher-order transition will produce significant dependencies. Kullback-Leibler divergence between current-order and higher-order nodes is used to weight the significance. The change is significant if it satisfies the

following divergence criterion [111]:

$$\mathscr{D}_{KL}(P_{cur}, P_{ExtNode}) > \frac{Order_{ExtNode}}{\log_2(1 + Support_{ExtNode})}, \tag{2.5}$$

where P_{cur} is a set of distributions of the current node, $P_{ExtNode}$ is a set of distributions of the extended node, and $Support_{ExtNode}$ is the number of observations of the extended node.

Adapting the idea of building a network from BuildHON+, the design of BuildHOTN was introduced for HOTN representations for multidimensional event sequence data. BuildHOTN uses a tensor network representation that can preserve more information, which leads to higher accuracy. Specifically, frequency and transition probability were used as the edge weight in the network representation, but more values (e.g., transition duration) were also included in the edge weight.

Figure 2.2 illustrates the BuildHOTN process, in which there are six events in six sequence data, and each sequence has three events. The duration of each transition is assumed to be 1 minute. BuildHOTN first builds all first-order observations, which contain counts and the total duration of the transition (step ①). Then, all first-order distributions are built, and each edge has the transition probability and average duration (steps ②→③). Because the Kullback-Leibler divergence is less than the threshold for paths $1 \to 3$, $2 \to 3$, and $6 \to 3$, which suggests that they have no higher-order dependencies, those paths will not be extended. Only paths $3 \to 4$ and $3 \to 5$ will be extended (step ④). Because the second-order observations and distributions of the two paths are not yet known, BuildHOTN selectively builds second-order distributions that significantly change the raw sequence data (steps ⑤→⑦). At this point, BuildHOTN stops building because there is no dependencies higher than second order. Otherwise, BuildHOTN will continue building (i.e., growing) until it reaches the maximum order.

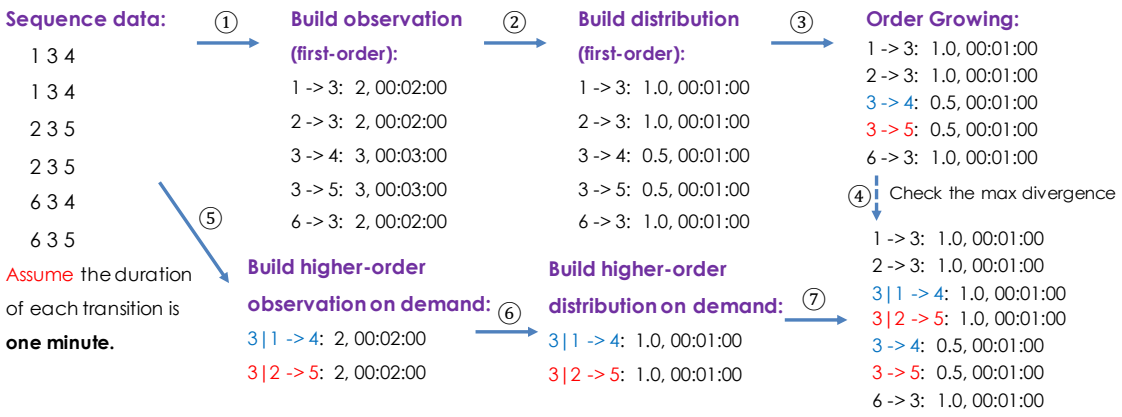


Figure 2.2: Example of BuildHOTN with transition probability and duration as the edge weight. The process order is labeled with circled numbers.

2.3.2 Sequence Anomaly Detection Based on Higher-Order Tensor Networks

The AD-HOTN algorithm presented in Alg. 1 uses a HOTN representation and graph-distance metrics. Figure 2.3 also presents a diagram of the AD-HOTN method when given a sequence set, $\mathbf{S} = \{\mathbf{s}_1,\mathbf{s}_2,...,\mathbf{s}_n\}$, which contains n sequences along with their event timestamps. First, the sequential data, $\mathbf{S}$, are represented as a tensor network, $\mathscr{N}_s$ (line 3, Alg. 1). Then, by removing a sequence $\mathbf{s}_i$ ($i \in (0, n]$) from data $\mathbf{S}$ each time, HOTN $\mathscr{N}_i$ is built for the sequence set $\mathbf{S}_i$ ($\mathbf{S}\backslash\{\mathbf{s}_i\}$) (lines 4–5, Alg. 1). By iterating the n sequences in $\mathbf{S}$, n HOTNs are built, $N_s = \{\mathscr{N}_1, \mathscr{N}_2, ..., \mathscr{N}_n\}$ (line 6, Alg. 1). Sequence $\mathbf{s}_i$ is removed from sequence set $\mathbf{S}$ because the network change will be revealed by comparing network $\mathscr{N}_s$ with network $\mathscr{N}_i$. If a normal sequence is removed, then the network will change less than when an outlier is removed. Because outliers always contain unique data patterns, eliminating those unique patterns from the network will remove some nodes or edges and cause large network changes.

The network changes are then quantified by calculating the network differences using network distance metrics, $\mathscr{NSD}$ [95]. The network metrics reveal the dissimilarity between two networks. The larger the network distance, the more dissimilar the networks are, and the more likely that the removed sequence, $\mathbf{s}_i$, is an anomaly. This work used the weight distance metric [95]; thus, the network distance between network G and network H is given as

$$\mathscr{NSD}(G, H) = |E_G \cup E_H|^{-1} \sum_{u,v \in V} \frac{|w_E^G(u, v) - w_E^H(u, v)|}{\max(w_E^G(u, v), w_E^H(u, v))}, \tag{2.6}$$

where $w_E^G(u, v)$ is the edge weight between nodes u and v for network G, and $|E_G \cup E_H|$ is the value of the union of edges in networks G and H.

Thus, the network distance between $\mathscr{N}_s$ and $\mathscr{N}_i$ will be calculated as $ND_i = \mathscr{NSD}(\mathscr{N}_i, \mathscr{N}_s)$ (line 7, Alg. 1). ND_i contains the network change transition probability and average duration (ND_i^P, ND_i^{AD}) because the edge weight contains the transition

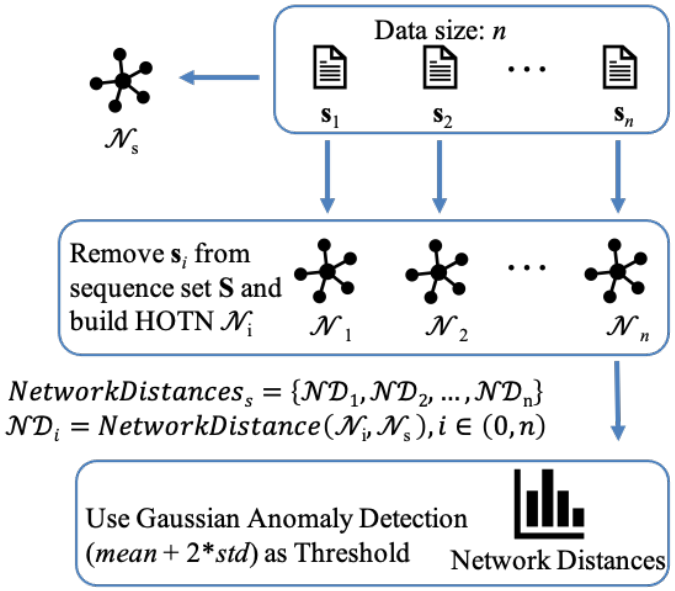


Figure 2.3: Diagram of anomaly detection based on HOTN.

probability, P, Eq. (2.3), and average transition duration, $AvgD$, Eq. (2.4). By iterating all networks in N_s, the network distance set, $ND_S = \{ND_1, ND_2, ..., ND_n\}$, is obtained (line 7, Alg. 1).

Detecting anomalies in sequences then becomes anomaly detection in point data, ND_S. Gaussian anomaly detection was applied to identify the anomalous sequences with transitions and/or duration. The threshold is calculated as $threshold = \text{Mean}(NDs) + 2 * \text{STD}(NDs)$, where STD refers to standard deviation (line 9, Alg. 1). If the network distance, ND_i, of the transition probability is larger than its threshold, then it will be identified as an anomaly with abnormal transitions (line 10, Alg. 1). Similarly, if the network distance, ND_i, of the transition duration is larger than its threshold, then it will be identified as an anomaly with abnormal duration (line 11, Alg. 1).

Algorithm 1 AD-HOTN.

1: **procedure** AD_HOTN(S)
2: $ND_s \leftarrow [\,]$ ▷ a list of pairs of network distances
3: $\mathscr{N}_s \leftarrow$BuildHOTN(S) ▷ build HOTN for S
4: **for** $i = 1 \rightarrow |S|$ **do** ▷ $|S|$ is the number of sequences
5: $S_i \leftarrow S$.pop(s_i) ▷ remove sequence s_i from S
6: $\mathscr{N}_i \leftarrow$ BuildHOTN(S_i) ▷ build HOTN for S_i
7: $ND_i \leftarrow \mathscr{ND}(\mathscr{N}_i, \mathscr{N}_s)$ ▷ calculate network distance between $\mathscr{N}_i$ and $\mathscr{N}_s$
8: append(ND_i) to ND_s
9: $threshold \leftarrow$ mean(ND_S) + 2* std(ND_S)
10: $pred_p \leftarrow$ [1 **if** $nd[0] > threshold[0]$ **else** 0 **for** nd in ND_s] ▷ anomalies with abnormal transition
11: $pred_d \leftarrow$ [1 **if** $nd[1] > threshold[1]$ **else** 0 **for** nd in ND_s] ▷ anomalies with abnormal duration
12: $pred \leftarrow$ zip($pred_p$, $pred_d$) ▷ zip two predictions
13: **return** $pred$ ▷ return prediction

2.3.3 Contiguous Subsequence Anomaly Detection Based on Higher-Order Tensor Networks

This section introduces the CSAD-HOTN approach for identifying anomalous subsequences within an anomalous sequence. The objectives of this approach are to (1) identify anomalous transitions and durations within the outliers detected by AD-HOTN and (2) predict the most plausible transition and duration for each anomalous transition and duration. Unfortunately, the existing subsequence detection algorithm (e.g., CSAD [77]) is not suited for this purpose because the conventional HON does not preserve enough information. The HOTN representation enables the opportunity for duration detection by carrying transition duration information in the network. CSAD-HOTN can also scale to higher dimensional sequence data beyond just transition probability and duration.

Algorithm 2 explains CSAD-HOTN in detail. To detect abnormal transitions and durations for an anomaly, $\mathbf{s}_x$, the $\mathscr{N}_x$ and $\mathscr{N}_s$ are generated in AD-HOTN; this step is illustrated in Fig. 2.4a. By comparing the edges in $\mathscr{N}_x$ and $\mathscr{N}_s$, the edges that exhibit significant differences between the two networks can be identified. Specifically, if the change in transition probability of an edge is larger than a threshold, then the edge is identified as an abnormal transition; if the change of transition duration is greater than a threshold, then the edge has an abnormal transition duration. The edge difference is quantified using the edge distance metric between edges of the network $\mathscr{N}_x$ and $\mathscr{N}_s$ (lines 20–26, Alg. 2). The edge distance between the edges of networks G and H is given as

$$\text{EdgeDistance}(G, H) = \frac{|w_E^G - w_E^H|}{\max(w_E^G, w_E^H)}, \tag{2.7}$$

where w_E is the weight for edge E (line 21, Alg. 2). If an edge is not in a network, then the edge weight will be (0, 0) (lines 23–24, Alg. 2). By using Gaussian anomaly detection, abnormal transitions and durations are identified (lines 30–31, Alg. 2), where the threshold is calculated as $threshold = \text{Mean}+2*\text{STD}$ (standard deviation).

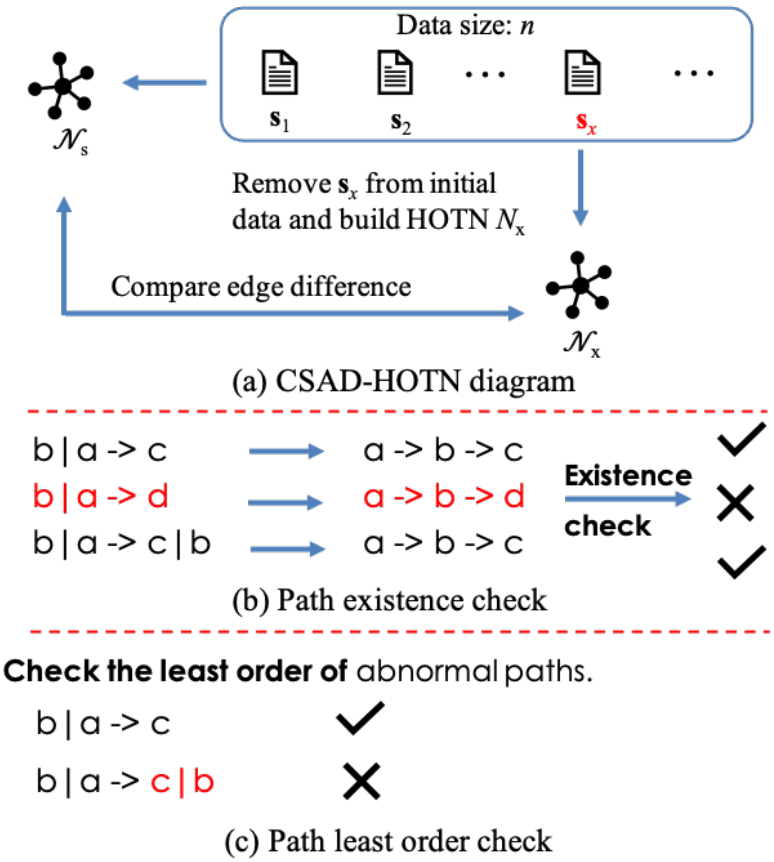


Figure 2.4: Diagram of CSAD-HON.

After the abnormal transitions and durations are identified in $\mathbf{s}_x$, the existence of the abnormal paths in the sequence are verified (line 8, Alg. 2). Then, only the least-order edges with the same source node are kept (line 9, Alg. 2). For example, in Fig. 2b, if edges $(b|a \rightarrow c|)$, $(b|a \rightarrow d|)$, and $(b|a \rightarrow c|b)$ are detected, then edge $(b|a \rightarrow d)$ will be discarded because the subsequence does not exist in sequence $\mathbf{s}_x$. In addition, edge $(b|a \rightarrow c|b)$ will be discarded because node $c|b$ is a higher-order node of $c|$ (i.e., both are from the same root node, c), and the least-order node is kept to retain more information and reduce redundancy.

After detecting the abnormal transitions or durations, CSAD-HOTN also recommends appropriate transitions (line 10, Alg. 2) and durations (line 12, Alg. 2) according to the edge weight in the network. By searching the edges with the same source node in the network, $\mathscr{N}_x$, CSAD-HOTN recommends plausible transitions according to the transition probability of edge weights—a larger transition probability of the edge weight means higher probability for the recommended transition. Similarly, an appropriate duration is also recommended by searching the path in the network, $\mathscr{N}_x$.

2.3.4 Spark-Based Implementation

For achieving algorithm scalability on large sequence data, the algorithms were implemented on the Apache Spark platform [116], which is an open-source software framework designed for distributed processing and analysis of big data applications. In this study, Apache Spark was used to process network representations for big sequence data while speeding up the anomaly detection process.

The size of a raw sequence data is given as L when building HOTNs, all first-order dependencies take $\Theta(2L)$ space, second-order dependencies take $\Theta(3L)$ space, and so on. Assuming that the number of k^{th}-order dependencies of each sequence is R_k $(1 \leq R_k < L)$, K is the maximum order, and N is the total number of sequences, then the space complexity to build an HOTN for N sequence is given

Algorithm 2 CSAD-HOTN.

Input: s_x: detected anomalous sequence; *pred*: prediction of s_x; $\mathscr{N}_s$: HOTN of S; $\mathscr{N}_x$: HOTN of S after removing s_x.

Output: AT: subsequences of s_x that have abnormal transition; RT: Recommend transitions. AD: subsequences of s_x that have abnormal duration; RD: Recommend duration.

procedure CSAD($\mathscr{N}_s, \mathscr{N}_x$)
  $Edges_s \leftarrow$ExtractEdges($\mathscr{N}_s$)
  $Edges_x \leftarrow$ExtractEdges($\mathscr{N}_x$)
  $AT, AD \leftarrow$AnomalyDetect($Edges_s, Edges_x$)
  if (*pred*[0] **and** *pred*[1]) **then**
    $AD \leftarrow$ remove item from AD that exists in AT
  if *pred*[0] **then** ▷ if s_x contains abnormal transition
    $AT \leftarrow$CheckPathExistence(AT)
    $AT \leftarrow$KeepLeastOrderPath(AT)
    $RT \leftarrow$Recommend($AT, Edges_x$,'T')
  if *pred*[1] **then** ▷ if s_x contains abnormal duration
    $RD \leftarrow$Recommend($AD, Edges_x$,'D')
  return AT, RT, AD, RD
function ExtractEdges($\mathscr{N}$)
  for *path* in $\mathscr{N}$ **do**
    $source, target, tp, avgd \leftarrow$ ExtractPath(*path*)
    ▷ tp: transition probability; avgd: average duration
    $Edges[(source, target)] \leftarrow (tp, avgd)$
  return *Edges*
function AnomalyDetect(G, H)
  $Distance \leftarrow [\,]$ ▷ a list of edges distance info
  $AllEdges \leftarrow G \cup H$ ▷ a list of all edges
  for *edge* in *AllEdges* **do**
    $weight_G \leftarrow G[edge]$ **if** *edge* in G **else** $(0, 0)$
    $weight_H \leftarrow H[edge]$ **if** *edge* in H **else** $(0, 0)$
    $distance \leftarrow$ EdgeDistance($weight_G, weight_H$)
    append $(distance, edge, weight_G)$ to *Distance*
  sort *Distance* by value *Distance*[0] descending
  $d \leftarrow Distance[:, 0]$ ▷ a list of all edge distance
  $threshold \leftarrow$ mean(d) + 2∗ std(d)
  $AT \leftarrow [[D[1], D[2]]$ **for** D in *Distance* **if** $(D[0][0] > threshold[0])$ ▷ identify abnormal transition
  $AD \leftarrow [[D[1], D[2]]$ **for** D in *Distance* **if** $(D[0][1] > threshold[1])]$ ▷ identify abnormal duration
  return AT, AD
function Recommend($Paths, Edges, mode$)
  $R \leftarrow [\,]$ ▷ a list of recommend paths
  for *path* in *Paths* **do**
    $source, target \leftarrow$ ExtractPath(*path*)
    if *mode* ==‘T’ **then** ▷ transition mode
      $Targets, weights \leftarrow Edges(source)$
      R.append $(source, Targets, weights)$
    if *mode* ==‘D’ **then** ▷ duration mode
      $weights \leftarrow Edges(path)$
      R.append $(source, target, weights)$
  return R

as $\mathcal{O}\left(N \cdot \sum_{k=1}^{K}(k+1)R_k\right) < \mathcal{O}\left(N \cdot \sum_{k=1}^{K}(k+1)L\right) = \mathcal{O}(NK^2L)$. Because the number of networks to build is N in AD-HOTN, the time complexity is $\mathcal{O}(N^2K^2L)$. If there are m executors when Spark-based AD-HOTN is deployed, then the time complexity of Spark-based AD-HOTN is given as $C_{\text{ADSpark}} = \mathcal{O}(N^2K^2L/m)$. Thus, the overall time complexity of Spark-based AD-HOTN will be significantly smaller when significantly more executors, m, are deployed. The side-by-side performance comparison between prior work [75] and Spark-based AD-HOTN is presented in Section 2.2.

2.4 Evaluations and Discussions

This section discusses the experiments on two datasets. The first was a synthetic dataset used to compare the performance of our proposed methods against the baseline methods. The second dataset used in this study consisted of clinical consultation events collected from a large-scale healthcare operation.

2.4.1 Baseline Methods

For comparison, two point-based predictive methods and three Markov-based methods were selected as the baseline methods (listed below). These methods represent state-of-the-art, and their performance and effectiveness have been demonstrated [88, 87, 120, 19, 16, 77, 75, 112].

- The **SGT** [88] sequence embedding method extracts the short-term and long-term sequence features and embeds them in a finite-dimensional feature space. The SGT was leveraged as an embedding method before adapting the two point–based predictive models.

- The **KNN-based detector** [87] uses an unsupervised approach, even though KNN is a supervised machine learning algorithm. The anomaly score of each sequence equals the dissimilarity to its KNN. After setting a threshold value,

outliers will be identified. This method was used as one of the baselines because it is an efficient and effective point-based anomaly detection method.

- The **k-means-based detector** [120] collects points into a fixed number of clusters, and the distance to its closet centroid is calculated for each data point. For each cluster, the data outside the threshold will be counted as an outlier.

- **N-gram** [65, 44] is widely studied by the natural language processing community. N-gram takes a fixed historical length of an event into consideration by calculating the probabilities. The idea is that a particular event in a sequence is only influenced by its recent predecessors rather than a single previous event or the entire history. To apply the N-gram model in our setting, N represents the fixed order of history.

- **PST** [98], or a probabilistic suffix tree, is a compact tree representation of a variable Markov chain that uses classical suffix trees as its index structure. Variable Markov techniques advance over the unreliable conditional probability of Markov techniques by allowing symbols to be conditioned on a variable-length history. While both PST and HOTN utilize variable Markov technology, PST employs a hard threshold for pruning the tree structure. The PST-based technique computes the likelihood for the test sequence, $\mathbf{s}$, with respect to the PST. Given a sequence, $\mathbf{s} = [s_1, ..., s_n]$, and a PST, T, the odds of $\mathbf{s}$ with respect to T are denoted by SIM_O and defined as

$$SIM_O(\mathbf{s}, T) = \frac{\prod_{i=1}^{l_s} P^T(s_i|s_1...s_{i-1})}{\prod_{i=1}^{l_s} P^T(s_i)}, \tag{2.8}$$

where $P^T(s_i)$ is the empirical probability of event s_i in $\mathbf{s}$, and l_s is the number of events in sequence $\mathbf{s}$. Thus, SIM_O represents the likelihood that $\mathbf{s}$ belongs to T (as opposed to a random sequence). If the value of $SIM_O(\mathbf{s}, T)$ is less than 1, then it indicates that $\mathbf{s}$ is more likely to be a random sequence (i.e., outlier) than part of T.

- The **HON-based detector** [77, 75, 112] uses a network representation that accurately and flexibly embeds variable orders of dependencies and is proposed to overcome the information loss of conventional network representations. HON-based anomaly detection methods use graph distance metrics to surveil the network change with and without a test sequence. If the distance is larger than a given threshold, then it is identified as an anomaly because it appears to have unique data patterns.

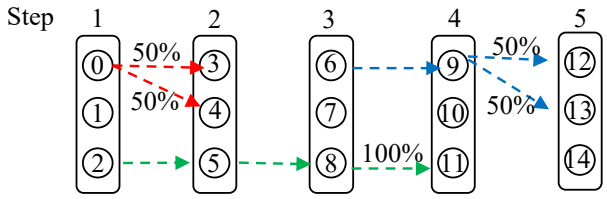
2.4.2 Synthetic EHR Dataset

Dataset Description

The primary advantage of using synthetic data is that the ground truth is known, which provides the what, when, and where for anomalies in the data. Figure 2.5 illustrates the process of building the synthetic EHR data.

The goal is to synthesize sequences with variable orders of EHRs. At the beginning, a set of 15 events is assumed and organized into a 3×5 grid, numbered from 0 to 14. Figure 2.5a illustrates the process of constructing the synthetic normal data. Each sequence consists of five steps, where one event is selected from each column with equal probability, except for three predefined constraints. **First-order constraint** (depicted by a red-dashed arrow): if the previous event is 0, then the next event must be 3 or 4 with equal transition probability. **Second-order constraint** (depicted by blue-dashed arrow): if the current event is 9, and the previous event is 6, then the event has a 50% chance of moving to event 12 and a 50% chance of moving to event 13. **Third-order constraint** (depicted by green-dashed arrow): if the previous movements are $2 \rightarrow 5 \rightarrow 8$, then the next event will be event 11. In addition, it is assumed that each transition in the process has a duration of 1 minute. One thousand sequences are generated as normal data to simulate 5,000 events.

Figure 2.5b shows the five synthetic anomalous sequences with their abnormal transitions labeled with colors that correspond to the rules they violated. Sequences



(a) The process of building synthetic EHR data

Synthetic outliers: (If not otherwise indicated, each transition last one minute)

1001: 0 5 6 9 12

1002: 0 3 6 9 14

1003: 2 5 8 10 14

1004: 0 3 6 9 12 (10 minutes between 9 to 12)

1005: 1 4 7 10 13 (10 minutes between 10 to 13)

(b) Synthetic anomalies

Figure 2.5: Building synthetic EHR data. (a) Process of building normal data in which one of the three events is selected in each step and five events in each sequence data. Three rules are specified using red-, blue-, and green-dashed arrows. (b) Five synthetic anomalies with abnormal transitions are also labeled using their corresponding colors.

1001, 1002, and 1003 did not follow the first-order, second-order, and third-order constraints, respectively. For sequences 1004 and 1005, abnormal durations were injected, specifically in the transitions 9 → 12 and 10 → 13, where the transition duration was set to 10 minutes instead of the normal 1 minute.

Experimental Settings

In the experiment, SGT was used as the embedding method for the three point-based anomaly detection methods. All methods were implemented in Python 3.6 and tested on a machine with 16 GB of memory. For KNN, the nearest 1% neighbor points were chosen for the $n_{neighbors}$ parameter by using Euclidean distance as a distance metric. For k-means, the original data were clustered to 100 clusters. The event length of N-gram was set to 5, and the maximum depth of PST was set to 5.

Results

Table 2.1 presents the detection results for synthetic data using five baseline methods and the proposed AD-HOTN approach, where m means miss (i.e., not detected), and h means hit (i.e., detected). The first three point-based methods are applied using SGT as the embedding method. As shown in the table KNN detected the first three anomalies, whereas k-means, and PST detected two anomalies each. ADHON and N-gram both detected three anomalies. None of the baseline methods detected anomalies with abnormal duration. By comparison, AD-HOTN detected anomalies with unique transition patterns and those with abnormal durations.

CSAD-HOTN detects additional anomalous subsequences and recommends plausible transitions for each anomaly. Table 2.2 shows these findings in more detail with different colors highlighting the transitions. For sequence 1001, the first-order transition 0→5 is identified, and 0→4 and 0→3 are given as plausible transitions with each near a 50% transition probability. For sequence 1002, synthetic abnormal transition 9|6→14 is identified, and 9|6→12 and 9|6→13 are given as

Table 2.1: Detection results for synthetic dataset: m means miss (i.e., not detected), and h means hit (i.e., detected).

ID	KNN	k-Means	N-gram	PST	ADHON	AD-HOTN
1001	h	m	h	h	h	h
1002	h	h	h	h	h	h
1003	h	h	h	m	h	h
1004	m	m	m	m	m	h
1005	m	m	m	m	m	h

Table 2.2: Abnormal transitions and recommended transitions for synthetic EHR dataset by CSAD-HOTN.

ID	Abnormal transitions (source→target, tp, duration)	Recommend transitions (source→target, tp, duration)
1001	0→5, 0.0028, 0:01:00	0→4, 0.527, 0:01:00 0→3, 0.473, 0:01:00
1002	9|6→14, 0.009, 0:01:00	9|6→12, 0.5, 0:01:10 9|6→13 ,0.5, 0:01:00
1003	8|5.2→10, 0.038, 0:01:00	8|5.2→11, 1.0, 0:01:00
1004	9→12, 0.337, 0:10:00	9→12, 0.337, 0:01:00
1005	7→10, 0.302, 0:10:00	7→10, 0.302, 0:01:00

plausible transitions with 50% transition probability. As for the third-order abnormal transition, 8|5.2→10 is also flagged correctly with 8|5.2→11 given as the plausible transition with 1.0 probability. For the last two outliers, 10 minute abnormal duration is captured for transition 9→12 and 7→10. The plausible transition duration is reported as 1 minute. In summary, CSAD-HOTN accurately detected all synthetic abnormal subsequences, and the recommended transitions were correctly identified.

Performance Comparison on Larger Synthetic Datasets

To perform a thorough performance comparison, the algorithms were evaluated on five synthetic datasets, each containing 100,000 normal sequences and 500 anomalous sequences. This enabled an extensive assessment of the algorithm's ability to detect anomalies across different dataset sizes and scenarios comparing with baseline methods. Specifically, for each dataset, the normal data are generated by using the same method shown in Fig. 2.5a. For the synthetic anomalies in each dataset, 400 outliers were generated that contained abnormal transitions and 100 outliers that contained abnormal duration. When generating the outlier that contains abnormal transitions, one of the three transitions rules in Fig. 2.5a (depicted as red-, blue-, and green-dashed arrows) was violated randomly. Then, the five datasets were tested with AD-HOTN and five baseline methods.

Performance Metrics. The evaluation of the anomaly detection methods is based on three commonly used metrics: Precision, Recall, and F-score. These metrics are computed as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.10}$$

$$\text{F} - \text{score} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}}, \tag{2.11}$$

where TP means true positive, FP means false positive, and FN means false negative. Precision shows the percentage of true anomalies in all detected anomalies. Recall

depicts the percentage of real anomalies detected. F-score is the harmonic mean of Precision and Recall.

To show the robustness and eliminate random system errors, we conducted five runs for each method in which synthetic data is generated randomly for each run. The results show that no FPs appeared for any methods in any run, which means their Precisions are 1. In contrast, they exhibited different performance for FNs. Table 2.3 presents the recall values of each run for the proposed method vs. baseline methods. According to the results, the only method that significantly improves detection accuracy is the AD-HOTN model, which has the best performance in each run. The next-best method detected only 80% of the outliers; specifically, it detected those with abnormal transitions. Interestingly, the performance of the KNN-based approach and of ADHON is almost the same with 79.6% average recall value and a 0.8% STD. Among the three point-based approaches, KNN performs the best and is the most stable. For the three Markov-based baseline methods, the average recall of ADHON is 79.6%, which means it performs better than N-gram and PST. AD-HOTN detected abnormal transitions and abnormal durations. Figure 2.6 compares the performance of AD-HOTN with five baselines. Each method's precision is not presented because they are all 1. It is worth noting that AD-HOTN achieves the highest F-score among the evaluated methods. This is attributed to its ability to successfully detect anomalies that specifically involve abnormal durations. KNN, N-gram, and AD-HON have similar performance for anomalies with normal transition durations.

2.4.3 Real-World EHR Dataset

Experimental Setup and Dataset

The EHR data collected in this work were obtained from a large-scale healthcare operation. The data were mapped into eight unique events: (1) `Created`, (2) `Ready`, (3) `Reserved`, (4) `InProgress`, (5) `Completed`, (6) `Failed`, (7) `Exited`, and (8) `END`.

Table 2.3: Test recall for each method on synthetic data.

Run	KNN	k-means	N-gram	PST	ADHON	AD-HOTN
1	78%	48%	76%	50%	80%	**100%**
2	80%	44%	80%	54%	80%	**100%**
3	80%	50%	80%	48%	78%	**98%**
4	80%	66%	80%	58%	80%	**100%**
5	80%	74%	78%	58%	80%	**100%**
Average	79.6%	56.4%	78.8%	53.6%	79.6%	**99.6%**
STD	0.8%	11.5%	1.6%	4.1%	0.8%	**0.8%**

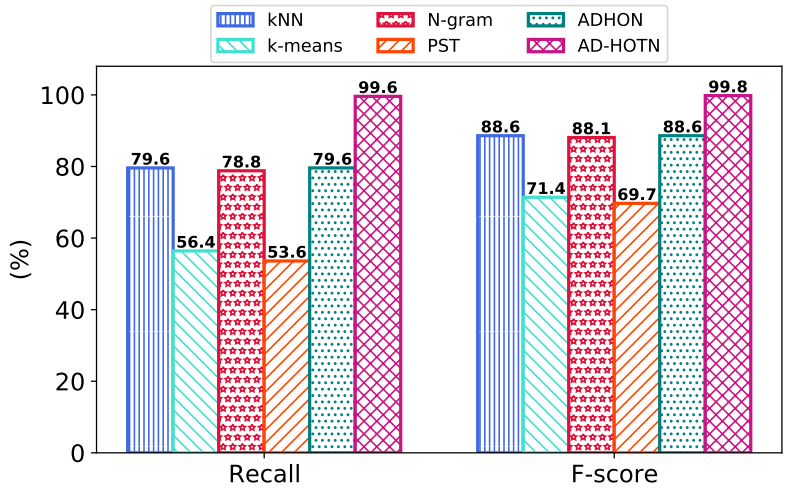


Figure 2.6: Performance comparison between AD-HOTN and baselines for large synthetic datasets. The Precision for each method is 1.

The dataset is collected between January 1, 2017 and December 31, 2019. 200,000 sequences were extracted in which the number of events per sequence was between 3 and 18 to test the proposed methods.

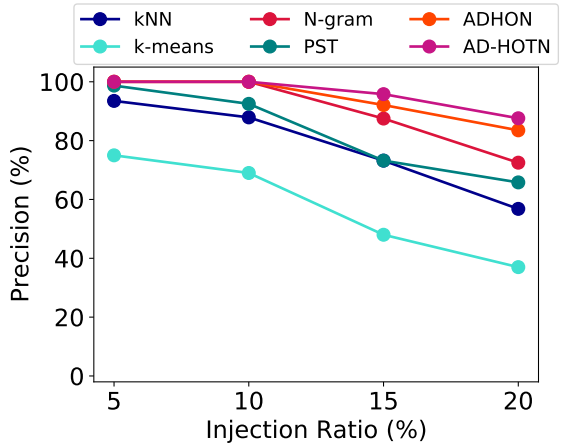
Data Preprocessing and Parameter Settings

The frequency of sequences can be as high as 20,000 and as low as 1. First, to drop random noise, the data that has a frequency of less than 30 were remove from the dataset. After preprocessing, 120,000 normal sequences are extracted. Next, varying ratios of real-world anomalies (e.g., 5%, 10%, 15%, 20%) are injected into the normal sequences for testing purposes. In the injected anomalies, 90% of them contain real-world abnormal workflows only, and 10% of them contain abnormal duration in normal workflows where the duration of a transition is substantially larger or smaller than those of normal sequences. A script is utilized to automatically select real-world abnormal workflows from the raw data. To create anomalous durations, a random modification is applied to a duration in a normal sequence. The experiment employs identical parameters used on the synthetic data.

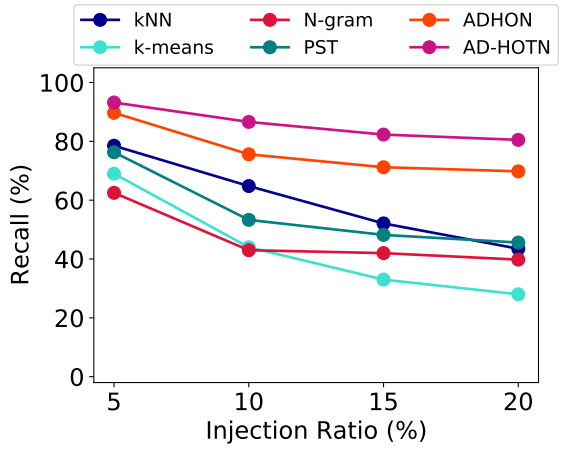
Results and Discussions

The anomaly detection performance of AD-HOTN is compared with five baseline approaches on the real-world EHR dataset, considering four different anomaly injection ratios. The results of the experiment are depicted in Fig. 2.7.

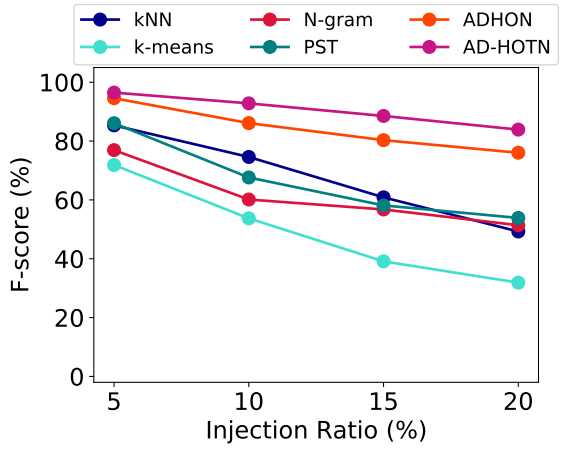
Observed from the Fig. 2.7, AD-HOTN performs much better than other approaches. With the increasing injection ratio of anomalies, the performance of the related approaches has declined in different degrees. In contrast, AD-HOTN still maintains a high accuracy even under a high injection ratio. For example, AD-HOTN can still achieve an F-score of 0.83 when the injection ratio is 20% (i.e., when 20% of the overall sequences are intentional injected anomalies). It confirms that



(a) Precision



(b) Recall

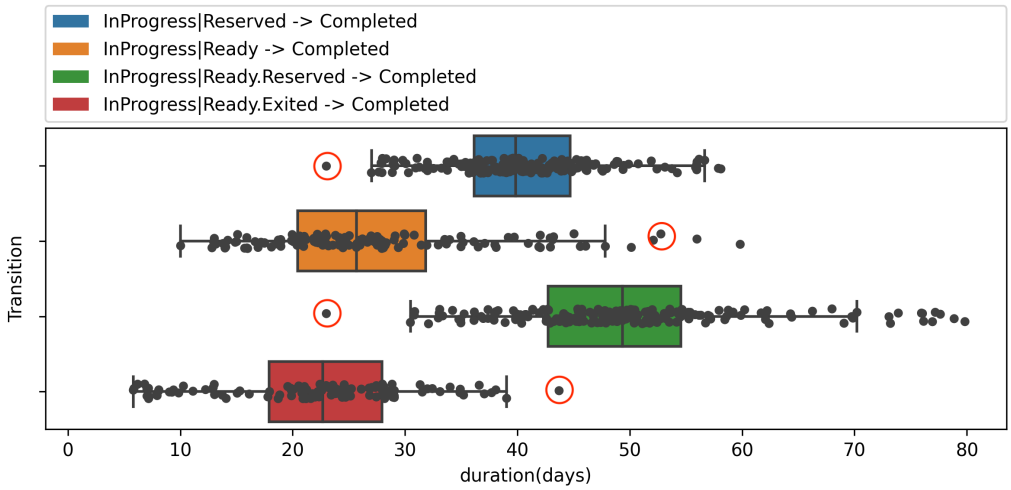


(c) F-score

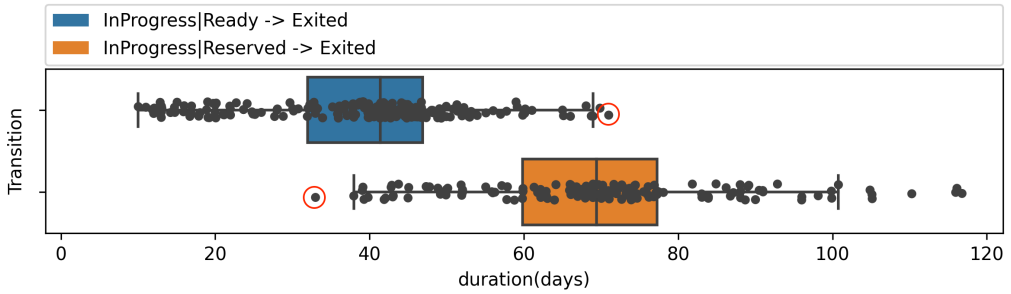
Figure 2.7: Comparison of anomaly detection accuracy between AD-HOTN and the baselines with four different anomaly injection ratios and a real-world dataset.

our approach is robust enough for real-world scenarios. The reason is that AD-HOTN can capture higher-order dependencies and their durations through HOTN representations, so it can identify hard-to-detect anomalies. Traditional approaches cannot support flexible higher-order growth when needed and are thus unable to achieve satisfactory results.

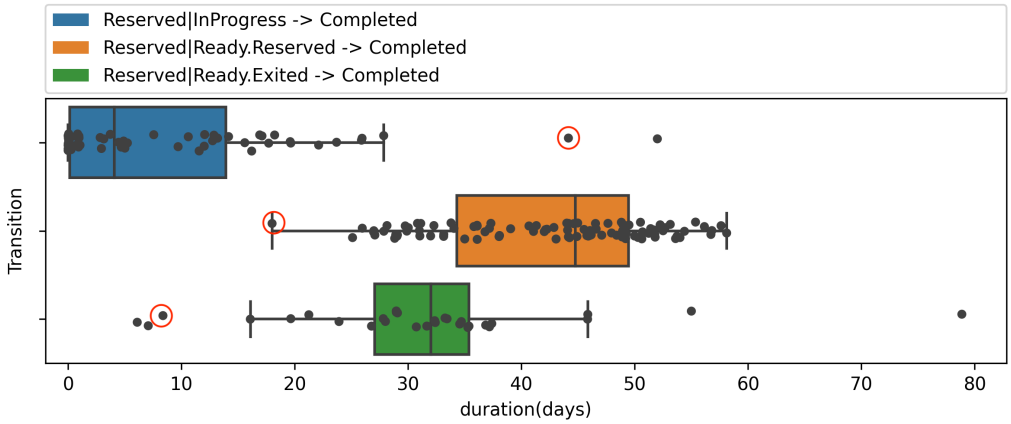
Second, three case studies were conducted to analyze the duration detection using AD-HOTN on real-world data, considering a 10% injection ratio. The details are shown in Fig. 2.8. In each case study, the transitions have the same start and end events but different dependencies. By considering the previous start events, AD-HOTN can identify those anomalies that were normal with different dependencies (some of them are labeled using red circles). Figure 2.8a shows four transitions that were generated by AD-HOTN from event `InProgress` to event `Completed`, but their dependencies are not the same. The durations of the four transitions have different distributions. From top to bottom, the four dots labeled with red circles are 22.3, 53.7, 23.1, and 43.2. The dots can be normal if the higher-order dependencies are not considered because they will hide in the middle of the data after stacking all the data points shown in Fig. 2.8a. Specifically, for the circled durations in the first (blue) and third (green), they can be normal if their transition dependencies are `Ready` (orange). Similarly, for the circled durations in the second (orange) and fourth transitions (red), they can be normal if their transition dependencies are `Ready.Reserved` (green). The same scenarios are also found in Figs. 2.8b and 2.8c. With different order dependencies built by AD-HOTN as needed, the salient information about different distributions is captured that helps identify anomalies that might be ignored by traditional approaches.



(a) Case study 1



(b) Case study 2



(c) Case study 3

Figure 2.8: Three case studies using AD-HOTN on real-world data with a 10% injection ratio. In each case study, the transitions have the same start and end events but different dependencies. By considering the previous start events, AD-HOTN can identify those anomalies that were normal with different dependencies (some of them are labeled using red circles).

2.4.4 Performance Scalability

The scalability advantage of Spark-based AD-HOTN was further demonstrated by utilizing datasets of varying sizes from real-world EHR data extracted from a large-scale HIT system. For comparison, AD-HOTN with a different number of executors and ADHON were deployed on identical hardware (Intel i5 quad-core CPU at 2.3 GHz, 16 GB of RAM, and a solid-state storage drive).

Figure 2.9 shows how these algorithms scale across a range of sequences for a different number of executors. For AD-HOTN, more nodes means less run time and better performance, especially as the data size increases above 60,000 cases. For comparison, running ADHON on a single node for the same dataset is also presented. Running Spark-based AD-HOTN on any data size is considerably faster than ADHON with more than one executor. The run-time speedup between AD-HOTN with eight nodes and ADHON is also presented in Fig. 2.9, which demonstrates the scalability of the Spark-based AD-HOTN algorithm.

2.4.5 Importance of Anomaly Detection for HIT Systems

The widespread adoption of information technology in health care has made creating an ecosystem of safer health care one of its goals. Because of this regulation, the use of HIT in US health care facilities has increased from 12.2% in 2009 to around 91% by 2017 [43]. This rapid HIT adoption has facilitated efforts to increase the quality and efficiency of health care services and decrease health care overhead while simultaneously generating massive amounts of EHRs [52]. Although HIT presents many new opportunities for improving patient care and safety, it can also create new challenges and vulnerabilities [13]. Recent studies that investigated the vulnerabilities in HIT [43, 52, 13] describe how data transmitted through the HIT systems could alter clinical decisions, interrupt the sequence of care, and/or alter care delivery processes. In a 10-year study of HIT related incidents, Martin et al. [68] found that 82% of incidents caused no harm to patients, 13% caused low harm, 4% caused

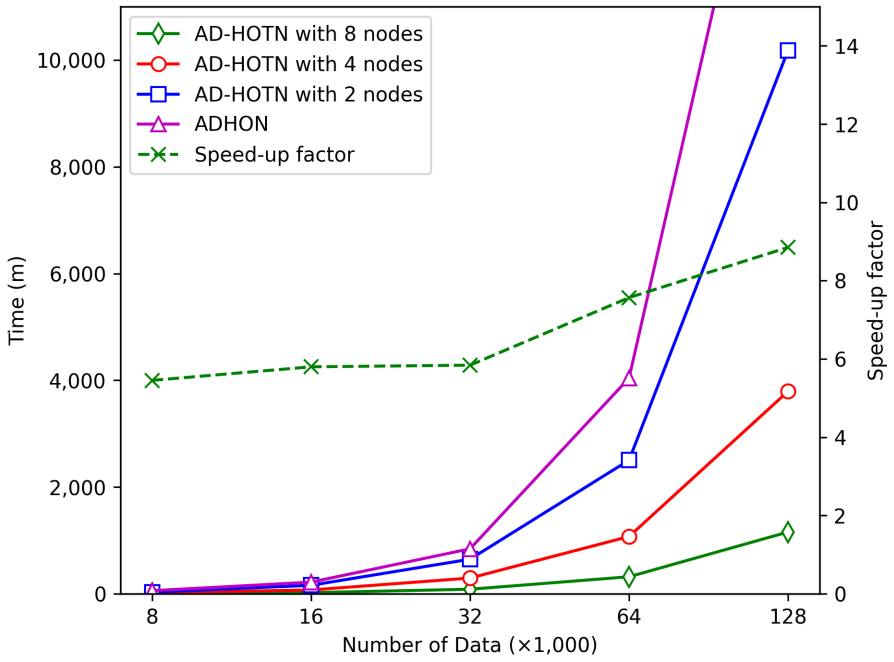


Figure 2.9: Performance comparison across different data sizes and approaches.

moderate harm, 1% caused severe harm and less than 1% caused death. In a survey of 229 incidents (from 249 reports) from the U.S. Food and Drug Administration Manufacturer and User Facility Device Experience database, Marcilly et al. [67] determined that 45.5% of the incidents resulted in no patient injury. Still, 12.7% (29 incidents) resulted in harm to the patient including 3 deaths (1.3%). The percentages of severe harm and death appear low but given the high number of HIT related incidents (e.g., 2,627 in England and Wales over the study's 10-year period [68]), the amount of harm is significant.

Anomaly detection techniques for healthcare sequential data can mitigate any potential patient safety hazards by detecting irregular patterns in the HIT data. This will help support improved decisions by clinicians, informaticists, IT specialists, engineers (e.g., software, systems), data scientists, and social scientists. Improvements in HIT hazard anomaly detection and mitigation have the potential to alleviate enormous suffering. This chapter presents new event sequence (and subsequence) anomaly detection algorithms that account for variable higher-order dependencies in the HIT data.

This study focuses on identifying infrequent patterns in EHR data that may indicate potential hazards, with a specific focus on clinical orders that have incomplete outcomes, such as the Error and Failed states. For example, proactively predicting the final state of clinical orders while they are still ongoing is one use case of interest. There are hundreds of thousands of orders per clinical domain (e.g., radiology, laboratory, etc.) generated daily in a large-scale HIT system. The objective is to identify possible Error or Failed clinical orders beforehand so that appropriate interventions can probably be performed to avoid a potential unsuccessful outcome. For instance, assume that an EHR system software glitch prevented some clinical orders from going through their usual route. This may create different event sequences that are rare, and they have never been observed in the HIT system. These sequences will be detected using the proposed approach and tagged for further analysis.

2.5 Conclusion

This chapter proposes higher-order tensor network-based event sequence, AD-HOTN, and continuous subsequence anomaly detection algorithms, CSAD-HOTN. The algorithms handle high-dimensional event sequence data by embedding event transitions and associated information about the events into variable orders of network representation. Specifically, anomalies in this study are identified as sequences that, when removed from the test dataset, result in a significant network change; the network change is then quantified using graph distance metrics. The algorithms were tested using synthetic and real-world EHR data, and the performance of AD-HOTN was compared with that of five baseline methods. The AD-HOTN algorithm outperformed the baseline methods, and the CSAD-HOTN algorithm correctly identified the plausible transitions for the detected anomalous subsequences in the abnormal sequences.

The work reported here has potential applications well beyond electronic health records. In the life sciences alone, anomaly detection can help highlight crucial risks and identify potential problems in patient care. As an illustration, an anomalous sequence of clinical actions may pinpoint a worrisome aberration from standard treatment regimens. Numerous examples have been presented in the literature. Anomalies in transcriptomic data may elucidate previously-unknown phenomena that warrant further study [63]. Anomalous metabolomic signatures can aid in the diagnosis of Central Nervous System diseases [104]. Anomalies in medication adherence patterns might also indicate behavioral instability [55].

Methods such as AD-HOTN and CSAD-HOTN that rely on network change to detect anomalies should perform well in these and related applications. Implications for advances in human health are manifest. Future work will involve conducting experiments on the data with more salient information between graph edges and performing experiments on larger real-world datasets.

Chapter 3

EHR-BERT: A BERT-based Model for Effective Anomaly Detection in Electronic Health Records *

3.1 Introduction

The increasing complexity of modern large-scale online computer systems makes them more susceptible to unexpected software malfunctions and malicious attacks. Anomaly detection in discrete event sequences is crucial for maintaining the reliability, security, and performance of large-scale computing systems. Fortunately, these systems typically log significant events and vital states, generating extensive log files that aid in monitoring the system and identifying the root cause of issues. HIT is one such system that offers EHRs as a valuable resource to enhance clinical decision-making, patient care processes, and system surveillance. As medical systems advance and the use of HIT systems becomes more widespread, detecting anomalies in EHRs is increasingly important for safeguarding patient safety and the assets of medical facilities.

*This chapter is a part of a manuscript of the same title authored by Haoran Niu, Olufemi A. Omitaomu, and Michael A. Langston, et al., which is currently under review.

EHRs, much like other system logs, document significant events and system states at various stages of operation, as well as valuable information regarding patient care and clinical processes. This makes EHRs indispensable components of HIT systems, and the effective management of EHRs is crucial for ensuring the quality and efficiency of healthcare services. Despite the importance of identifying anomalies and risks in HIT systems, the task remains complex due to the unstructured nature of the logs, which often consist of text messages that are challenging to analyze automatically. Additionally, the high number of events and their interactions create another challenge in anomaly detection. There is also considerable variability in events across different systems, and the temporal nature of sequential events means that the order of events is crucial. The sheer volume of data generated daily from these complex systems also poses a challenge for real-time detection. Consequently, the complexity of EHRs has led to various challenges and vulnerabilities, necessitating advanced and effective anomaly detection techniques.

In response to these challenges, numerous anomaly detection methods have been developed for discrete sequences. Initially, log-based anomaly detection techniques were based on rule-based approaches [26, 80, 114]. These methods relied on keywords or regular expressions to identify anomalies, requiring expert knowledge to create the rules, making it challenging to detect new or sophisticated anomalies. Currently, there are three main categories of sequence anomaly detection techniques: statistic-based methods (e.g., PCA [113], Invariant Mining [61], LogCluster [58]), Markov-based methods (e.g., n-gram [66], PST [98], HOTN [78]), and deep learning-based models (e.g., DeepLog [30], LogAnomaly [70], RobustLog [118], LogBERT [37]). Statistical-based methods analyze the event count vector or mine invariants to detect anomalies. For example, Xu et al. [113] applied PCA to the event count vectors to identify outliers. Lou et al . [58] used an unsupervised clustering method on the event count vectors to differentiate anomalies from normal sequences. These methods overlook the temporal information inherent in the sequence, thereby limiting the models' ability to generalize. Markov-based methods, on the other hand, have

demonstrated good performance in capturing the interactions between events and identifying anomalies in system logs. For instance, PST [98] uses a probabilistic suffix tree to compactly represent the dependencies between events of varying lengths. Anomalies can be identified by comparing the similarity between the test sequence and the tree models and flagging those with low scores. Niu et al. [78] proposed a HOTN to capture not only event dependencies but also their relevant information. Although these approaches are interpretable, they have limitations when dealing with long dependencies.

Recently, there has been a growing trend to use deep learning models for sequence anomaly detection by treating discrete event sequences as if they were natural language sentences. Recurrent neural network (RNN)-based models have demonstrated excellent performance and scalability in this task. DeepLog [30] was the first to propose an RNN-based model, which was later improved upon by LogAnomaly [70] and RobustLog [118] by adding attention layers and semantic information, resulting in improved detection accuracy. Although these models have demonstrated promising results, they come with some limitations. For example, recurrent neural networks are known to be notoriously hard to train due to their exploding and vanishing gradient problems. To address these limitations, Guo et al. [37] proposed LogBERT, which is inspired by the success of Bidirectional Encoder Representations from Transformers (BERT) in natural language processing tasks. LogBERT aims to capture the contextual information of sequences, yet it has been observed that using the Random Masked Token Prediction (RMTP) approach leads to low accuracy and high false negatives for EHR sequence data, due to the sparsity of abnormal events in anomalous sequences. Furthermore, the feasibility of the BERT model for healthcare application was successfully showcased through the development of the Med-BERT model for disease prediction [89].

Motivated by these developments and challenges, the study introduces EHR-BERT, a novel BERT-based model specifically designed for effective anomaly detection in EHR sequences. Our model advances the state-of-the-art by leveraging

the power of BERT and a newly proposed Sequential Masked Token Prediction (SMTP) strategy. The introduction of SMTP enables EHR-BERT to outperform existing models in the domain of anomaly detection. By treating log sequences as natural language sentences, EHR-BERT effectively captures contextual information and event correlations, improving the accuracy of anomaly detection in EHR sequences. The input events in this model are masked incrementally to minimize false negatives. In particular, the BERT model is utilized to capture the contextual information between the input events. This gives it the ability to find the relationships among events which goes beyond temporal/sequence adjacency. The model is trained on sequences of normal events, allowing it to automatically learn the patterns in the sequences from both directions of each event through the SMTP strategy. As a result, the model can detect anomalous sequences that violate the correlations it has learned. To mitigate the issue of false negatives, the study introduces the SMTP strategy, which is applied during both the training and prediction stages. By iteratively checking subsequent events until each event has been evaluated at least once, our model will ensure that no abnormal events are overlooked.

The primary contributions of our work are the introduction and evaluation of EHR-BERT, which demonstrates significant advancements in the field of anomaly detection in EHRs. We show that the SMTP strategy is highly effective in enhancing the model's ability to learn patterns from normal log sequences and improves accuracy in predicting masked tokens. Our extensive evaluation on large HIT system logs and diverse clinical domains highlights EHR-BERT's superior performance compared to the state-of-the-art baseline methods. This demonstrates the potential of our model to enhance patient safety, reduce medical errors, and support healthcare professionals in decision-making. Furthermore, the method introduced in this study can have broader implications for sequence anomaly detection tasks, as it could enhance the performance of BERT-based models in various large-scale computing systems beyond healthcare.

In summary, this chapter makes the following contributions:

- The problem of high false negatives in sequence anomaly detection tasks using BERT-based models with the existing RMTP method is highlighted.
- EHR-BERT is proposed as a novel self-supervised sequence anomaly detection model that incorporates the SMTP strategy to enhance prediction accuracy and reduce false negatives.
- The performance of EHR-BERT is evaluated using three large-scale EHR sequence datasets in different settings, and the approach is shown to outperform the current state-of-the-art baseline methods.

The remainder of this chapter is organized as follows: Section 3.2 reviews related work. Section 3.3 presents our proposed self-supervised EHR-BERT model and the Sequential Masked Token Prediction strategy. The evaluation of the proposed model against five baseline methods using EHR sequence datasets from three medical domains is presented in Section 3.4. Finally, Section 3.5 summarizes the key findings of the study.

3.2 Related Work

In this section, a brief overview of the most commonly used methods in anomaly detection for event sequences is presented as this has been an active area of research over the last decade.

Initially, early techniques for detecting anomalies in discrete sequences were rule-based, relying on regular expressions or keywords such as "`fail`" or "`exception`" to identify unusual patterns [26, 114, 80, 92, 93]. Constructing rules for detecting anomalies in sequence data requires expertise and can limit the detection of new patterns, making it challenging to achieve the desired flexibility. As a result, manual inspection of sequences is often necessary to detect anomalies. For instance, Yen et al. [114] proposed an unsupervised clustering method for log features that could potentially detect anomalies, but it still requires manual labeling of such anomalies.

In order to achieve automatic anomaly detection, some researchers have employed statistical-based learning approaches for classifying sequences [113, 61, 58, 57]. Liang et al. [57] used Support Vector Machines (SVM) to detect failures, where event count vectors are used in the training stage and a sequence is flagged if it is located above the hyperplane during prediction. Xu et al. [113] applied PCA to log outlier detection, where each log sequence is vectorized as an event count vector, and PCA is used to find patterns between the event count vectors. Lou et al. [61] discovered log invariants related to system execution and proposed Invariant Mining (IM) to identify anomalies by automatically mining violations of invariants. LogCluster [58] is an unsupervised approach that clusters logs into normal and abnormal sets by vectorizing log sequences as event count vectors.

Markov-based methods are another popular category of sequence anomaly detection approaches that use the higher order dependencies in sequences by building a probabilistic model from the conditional probability for events. These models effectively learn the distribution of event dependencies during the training stage, providing a reliable estimate of conditional probabilities. Basic fixed Markovian techniques can accomplish this task [66, 10, 72]. Boldt et al.[10] applied fixed-order Markov models to streaming sequence data, considering different temporal resolutions. While fixed Markov methods have been useful in building probabilistic models of sequence data, they have inherent limitations due to the storage requirements for capturing all transition dependencies [20]. To address these limitations, variable Markov-based methods have been proposed [98, 77, 78] that allow symbols to be conditioned on a variable-length history. dependencies [20]. Variable Markov-based methods [98, 77, 78] are proposed to address these limitations by allowing symbols to be conditioned on a variable-length history. Sun et al.[98] proposed the Probabilistic Suffix Tree (PST) to represent a variable-length history of a sequence in a compact tree representation. Niu et al.[78] considered salient information using variable-order tensor network representations and identified anomalies by monitoring the network changes.

To address the challenge of detecting anomalies in complex modern systems, researchers have increasingly turned to deep learning models for their powerful learning capabilities. RNNs have been particularly successful in natural language learning tasks, leading to their natural application in sequence anomaly detection by treating discrete event sequences as natural sentences. Du et al. [30] proposed DeepLog, a sequence anomaly detection approach that employs a LSTM model to learn correlations between events from normal sequences. The model then predicts the next events based on a fixed history of log sequences as input. If the actual log event is not in the prediction list, the test sequence is flagged as an anomaly. To enhance the model's capability in dealing with unknown events, researchers have proposed LogAnomaly [70] and RobustLog [118], both of which incorporate semantic information into RNN models using word embedding techniques. In the meanwhile, they also utilizing attention schema [7] to capture the contextual information in the log sequences and automatically learn the importance of different log events. With the self-attention design of Transformer [105], compared with RNN models, it can reduce training time since it processed a sequence as a whole rather recursively, and also reduce information loss in long dependencies. Guo et al. [37] proposed a BERT-based model to learn the patterns of normal sequences by utilizing a RMTP method. Although it shows promising results, it is observed that the random masked method can produce high false negatives in EHR sequences. Motivated by it, the study proposes a sequential masked token prediction task, which aims to improve the prediction accuracy of BERT-based anomaly detection models.

3.3 Design of EHR-BERT

3.3.1 Overview

To achieve accurate anomaly detection, a novel sequence anomaly detection model called EHR-BERT is proposed in this chapter. The model leverages the BERT

language model to learn patterns in EHR sequences in both forward and backward directions, allowing it to detect anomalies when the EHR patterns deviate from those learned from normal EHR sequences. Guo et al. [37] presented the RMTP task, demonstrating the potential of using the BERT model for detecting anomalies in system logs with promising results. Despite that, it is found that applying the RMTP task to EHR sequence data can lead to a high rate of false negatives. This is due to the fact that when testing abnormal sequences with a low mask ratio, most of the masked events are normal, and the model can still make correct predictions based on the unmasked events, even if there are sparse abnormal events present. This can deceive the model into falsely classifying abnormal sequences as normal, resulting in low performance and a high number of false negatives. In contrast to the RMTP task used in LogBERT [37], this study introduces a self-supervised SMTP strategy that masks events in an input sequence iteratively, resulting in a significant improvement in detection accuracy.

3.3.2 Framework

Figure 3.1 illustrates the design of our proposed EHR-BERT framework, which consists of three main steps: EHR parsing, EHR sequencing, and BERT-based classification. Initially, the EHR data is extracted from HIT systems and parsed, converting free-text messages into structured EHR log data that includes information about patients, events, timestamps, and other relevant details. These preprocessing steps are crucial for preparing the EHR data for subsequent analysis with the EHR-BERT model. After parsing the EHR data, sequences of events are constructed in chronological order by organizing them according to patient records. To extend the applicability of the proposed approach beyond patient records, the sequence construction process can be modified to organize events by different categories, such as departments or employees, depending on the requirements of the application. For instance, the approach can be applied to log data generated by a software company to

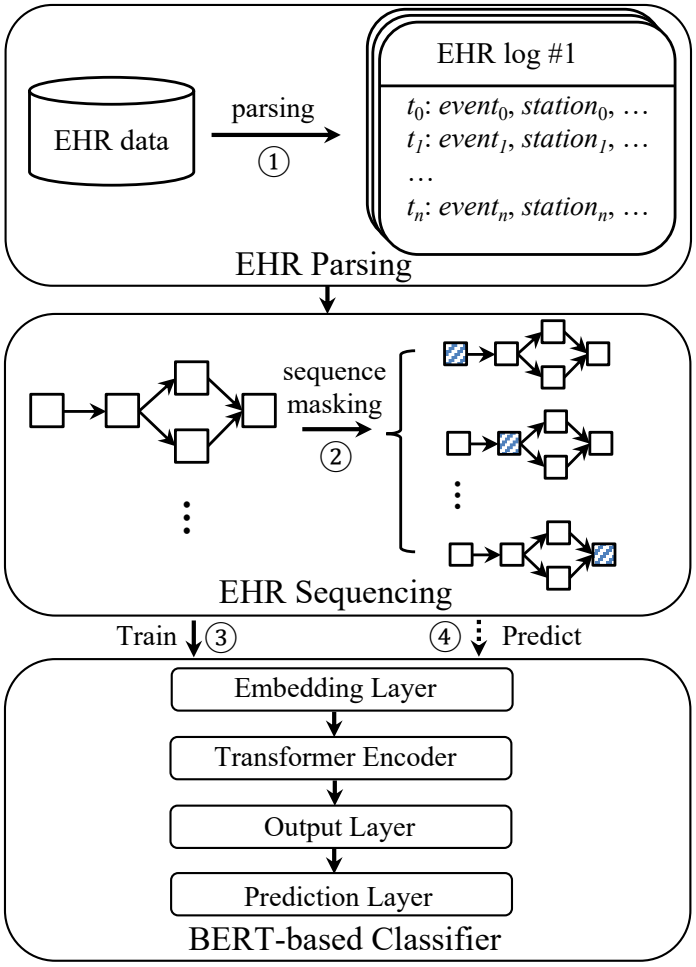


Figure 3.1: The Overview of EHR-BERT.

detect anomalies in software development processes. Existing BERT models [27, 37] typically take sequences with random masks as inputs, which can lead to a high rate of false negatives and low performance in anomaly detection tasks. To address this issue, EHR-BERT replaces events in a sequence with a MASK token successively, generating training and prediction data that allows for more accurate detection of anomalies in EHR sequences. Once the EHR sequences have been generated, BERT [27] is leveraged to detect anomalies. BERT can capture bidirectional context information from sequences, making it well-suited for anomaly detection tasks. During training, the model is trained to capture normal sequence patterns by encoding the correlation among events and accurately predicting masked events in normal sequences. This process enables the trained model to discriminate anomalies when predictions cannot be matched to masked events, improving the accuracy of anomaly detection in EHR sequences.

EHR Preprocessing

Raw EHR log data is of paramount importance for successful automated EHR analysis. Nevertheless, raw EHR data is often composed primarily of free-text data and contains a wide range of specific information, including demographics, medical history, medication, laboratory test results, and more. To streamline this process, the study incorporates EHR parsing as the initial step (①) in Figure 3.1. EHR parsing helps to convert the raw data into structured EHR log data, making it easier to analyze and work with. Our model extracts events from each EHR log data within a massive volume database. Typically, relevant information is dispersed across multiple tables in the database, and it may require denormalization and careful examination to identify the events. To achieve this, the method first identifies all timestamp records that represent an event on a log. During the process, non-character tokens such as delimiters and punctuation marks are removed from the log data. Additionally, stop words (e.g., a, an, the) are eliminated. To facilitate event extraction and reduce redundancy, the WordPiece model tokenization technique [109] is employed. For

example, the parsing process maps phrases like `"EXAM IN PROGRESS"` and `"BEING EXAMINED"` to the same event `"in exam"`. The etymologically related events are also combined, for example, `"ADDENDUM ADDED"` and `"Add Comments"`.

Data Sequencing. After cleaning the data, EHR-BERT sorts the events for each patient in chronological order based on their timestamps. Each event sequence is then treated as a sentence in natural language, denoted as $S^j = [t_1^j, ..., t_i^j, ..., t_n^j]$, where t_i^j ($i \in [1, n]$) represents the i-th event/token and n is the length of the sequence j. The model then applies SMTP, a Sequential masking method (step ② in Figure 3.1) to each sequence, masking a certain number of events at a time, and iteratively masking all events to generate more sequences. This approach increases the amount of data and helps the model capture event correlations in detail, while also reducing false negatives in the prediction stage (as discussed in Section 3.3.3).

BERT-based Classifier

The BERT-based classifier shown in Figure 3.2 is used to train and predict the preprocessed EHR sequences. The model includes two embedding layers, a Transformer Encoder, and a prediction layer. The training objective is to capture the correlation among sequence events in normal sequences by predicting the masked events.

Input Embedding. The input embedding of EHR-BERT includes adding a special token t_{cls} at the beginning of each sequence as the first token, which is used as the aggregate sequence representation for the whole sequence. In addition, masked tokens in the data are replaced with a special token denoted as t_{mask}. These modifications allow the model to understand the relationship between different tokens and to capture the masked tokens in the sequence. The embedding layer consists of token embedding and position embedding. The first embedding layer, the token/event embedding, represents individual tokens as real-valued vectors in a predefined vector space, defined as a matrix $\mathbf{E}$ with dimensions $\mathbb{R}^{n*d}$, where n is the length of the input sequence and d is the dimension of token embedding.

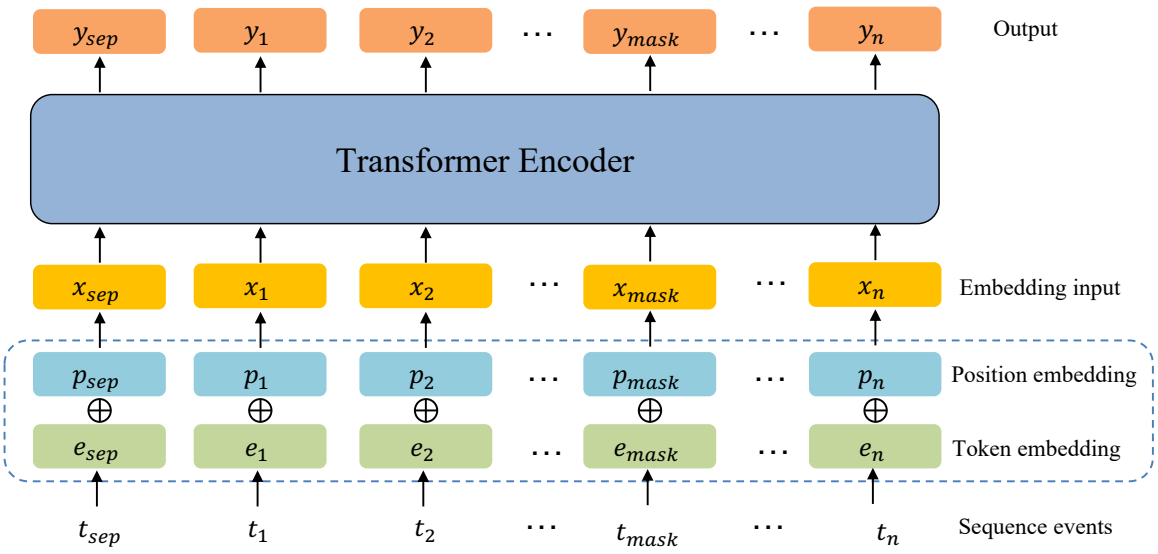


Figure 3.2: BERT-based classifier.

The Transformer encoder in BERT does not have a recurrent structure like RNNs, so a position embedding layer is added to capture temporal information of a sequence. The position embedding encodes the position information of sequence tokens using sine and cosine functions. Specifically, for the i-th token in a sequence and the k-th dimension of the d-dimensional embedding, the position embedding is defined as $\mathbf{P}(i, 2k) = \sin(i/10000^{2k/d})$ and $\mathbf{P}(i, 2k+1) = \cos(i/10000^{2k/d})$. This sinusoid position embedding allows the model to encode relative positions, with $\mathbf{P}_{i+l}$ represented as a linear function of $\mathbf{P_i}$ where l is a fixed offset. The input embedding of token t_i is obtained by adding its token embedding e_i^j and position embedding p_i^j, which can be expressed as $x_i^j = e_i^j + p_i^j$.

Transformer. The EHR-BERT model is based on a multi-layer bidirectional Transformer encoder architecture proposed by Vaswani et al. [105] in 2017. Each transformer layer consists of two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. A residual connection is employed around each sub-layer, followed by layer normalization [42, 6]. In a multi-head self-attention mechanism, the attention function maps a query and a set of key-value pairs to an output, which is computed as a weighted sum of the values. The weight assigned to each value is computed by a compatibility function of the query with the corresponding key. The input to the attention function consists of queries and keys of dimension d_k, and values of dimension d_v. The output of the scaled dot-product attention mechanism is calculated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3.1}$$

where Q, K, V represent the queries, keys, and values, respectively. The softmax function calculates a weighted sum of the values, where the weight assigned to each value is determined by the compatibility function of the query with the corresponding key. The attention mechanism allows each key to attend to all the input tokens by computing a sum attention score distribution over the input sequence.

Rather than performing a single attention function, EHR-BERT uses multiple attention functions by projecting the queries, keys, and values h times with different learned linear projections. This approach is called multi-head attention, where the model employs h parallel self-attentions to attend jointly to information from different representation subspaces at different positions. Formally, the multi-head attention concatenates h parallel heads together as:

$$\text{MultiHead}(X^j) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{3.2}$$

where

$$\text{head}_i(Q, K, V) = \text{Attention}\big(X^j W_i^Q, X^j W_i^K, X^j W_i^V\big) \tag{3.3}$$

where $X^j \in \mathbb{R}^{n*d}$ is the input representation of sequence j; W_i^Q, W_i^K, W_i^V, W^O are parameter matrices of the projections with dimension $\mathbb{R}^{d*d_k}, \mathbb{R}^{d*d_k}, \mathbb{R}^{d*d_v}, \mathbb{R}^{h*d_v*d}$ respectively.

In addition to multi-head attention layers, each layer of the EHR-BERT model also includes a fully connected feed-forward network. This network is applied to the hidden representation of each event separately and identically. Thus, a transformer layer with a ReLU activation is defined as:

$$\text{FFN}(X^j) = ReLU\bigg(\Big(\text{Attention}\big(X^j\big)W_1\Big)W_2\bigg) \tag{3.4}$$

where W_1 and W_2 are two linear transformations.

A transformer encoder typically consists of multiple transformer layers to generate accurate sequence embeddings. The output of the transformer encoder for a given input sequence is computed as:

$$y_i^j = \text{Transformer}_L\big(x_i^j\big) \tag{3.5}$$

where Transformer_L represents the transformer encoder with L layers and y_i^j represents the output embedding vector of the input token t_i^j.

3.3.3 Sequential Masked Token Prediction

In order to reduce false negatives introduced by RMTP and improve prediction performance, the study proposes a self-supervised task to capture dependencies among events in normal sequences during model training, and to identify anomalous patterns in the prediction phase. This method, which is described in this section, is called sequential masked token prediction.

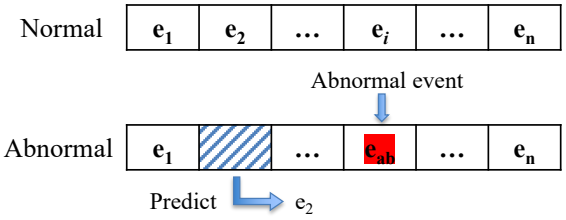
Traditional language models can only be trained in a left-to-right or right-to-left direction. To train a deep bidirectional representation, Devlin et al. proposed random masked language model in their work [27]. This method involves masking a certain percentage of the input tokens at random and then predicting those masked tokens. The approach has been highly successful in various language generation and understanding tasks. LogBERT [37] advanced the field of anomaly detection by utilizing RMTP to train BERT models for system log sequence anomaly detection tasks. In this method, a small portion of input tokens is randomly masked, and the model generates a set of candidates from the top-g output based on its training on only normal sequences. If the real tokens are in the candidate set, the test sequence is treated as normal.

While these methods have shown promising results in some applications, they can suffer from high false negatives when applied to EHR sequence anomaly detection tasks, where anomalous events are sparse. An example of this is shown in Figure 3.3a. There is a normal sequence pattern $S_n = [e_1, e_2, ..., e_i, ..., e_n]$ used to train the model using MLKP. When testing an abnormal sequence $S_a = [e_1, e_2, ..., e_{ab}, ..., e_n]$, which contains the same events with the same order in S_n, except that it replaces the normal event e_i in S_n with an abnormal event e_{ab} in S_a, the MLKP approach randomly masks events. When using a low masking ratio (such as the commonly used 15%), it

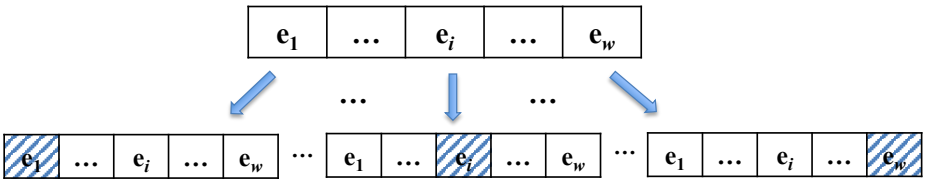
is very common for the randomly masked events to all be normal events, as abnormal events are sparse in anomalies. In other words, in abnormal sequences, most events are normal. Here, an example is demonstrated where the normal event e_2 is masked, and the BERT model predicts it based on the unmasked events. Our observation shows that the BERT model can still make correct predictions for the masked token even with sparse abnormal tokens in the input. Therefore, even with an abnormal event present in the test sequence, the model may still predict the masked event as e_2, leading it to believe that the sequence is normal.

To address this issue, the study proposes a self-supervised training and anomaly detection strategy called SMTP. This method involves iteratively masking a certain number of events in a sequence and training the model to predict the masked events at each step. By masking a small number of events at a time and iteratively masking all events, the dataset is augmented, resulting in an increased number of training examples. This approach helps the model capture event correlations in greater detail. Throughout the process, to enhance stability and robustness in training, SMTP employs a masking strategy that includes masking the token with a [MASK] token 80% of the time, substituting the token with a random token 10% of the time, and retaining the token unchanged 10% of the time. Moreover, by focusing on predicting multiple masked events in the sequence, the risk of false negatives is reduced in the prediction stage. The successive masking process is illustrated in Figure 3.3b. To ensure that the EHR-BERT model receives inputs of a fixed length, a sliding window of size w is applied to the sequences before masking is performed. The SMTP process involves masking k events at each iteration until all events have been masked at least once. For an input sequence $S_n = [e_1, ..., e_i, ..., e_w]$, a sliding window of size w is applied before masking. Setting $k = 1$ results in w sequences, each containing one masked token. The masked sequences are then fed into the BERT model, which outputs a probability distribution over the masked token e_{mask} for each sequence j:

$$\hat{y}^j_{mask} = \text{Softmax}\big(\text{Transformer}(x^j_{mask})\big), \tag{3.6}$$



(a) An example where RMTP causes false negative



(b) SMTP process

Figure 3.3: Subfigure (a) illustrates how RMTP can introduce false negatives. On the other hand, subfigure (b) shows the SMTP process.

where x^j_{mask} is the embedding vector of the masked token e^j_{mask}. Then, $\hat{y}^j_{mask}$ will be used to predict the original token with cross entropy loss, where the objective function is defined as:

$$\mathscr{L}_{\text{SMTP}} = -\frac{1}{B}\sum_{j=1}^{B}\sum_{i=1}^{K} y_i^j \log \hat{y}_i^j, \tag{3.7}$$

where probability distributions $y^j_{mask_i}$ indicates the real i-th masked token, K is the total number of masked tokens in the sequence j, B is the batch size.

Once trained with SMTP (step ③ in Figure 3.1) using normal sequences, EHR-BERT becomes capable of capturing the dependencies among normal events. Consequently, EHR-BERT can distinguish between normal and abnormal sequences by detecting the abnormal correlations among events (step ④ in Figure 3.1). Specifically, the test sequences are masked using the same procedure as the one applied during training, and EHR-BERT predicts the masked tokens based on the probability distribution of the model's output (Equation 3.6). A candidate set is generated, consisting of g normal tokens. The g tokens with the highest likelihood values are selected by computing $\hat{y}^j_{mask}$. If the real masked token is present in the candidate set, we classify the sequence as normal. On the other hand, if the masked token is not found in the candidate set, the sequence is labeled as abnormal. This is because the low likelihood of the masked token suggests that it does not preserve normal dependencies with other tokens. A threshold, denoted by r, is set for the undetected token ratio. If the undetected token ratio exceeds r, the sequence is classified as abnormal. Specifically, the undetected token ratio is calculated as the ratio of the number of undetected tokens to the total number of masked tokens, i.e., $R = \frac{\#\text{ undetected tokens}}{\#\text{ masked tokens}}$.

3.4 Evaluations and Discussions

In this section, the experimental dataset extracted from the database of a large-scale healthcare operation is described. The baseline methods are outlined, the

experimental setup is explained, and the evaluation metrics are specified. Finally, the performance of EHR-BERT on the extracted EHR data is evaluated and compared to five baseline methods.

3.4.1 Datasets

In this study, EHR data generated from a large-scale healthcare operation was obtained. The dataset consists of records collected between January 1, 2017, and January 1, 2019. Three EHR datasets were extracted from the healthcare domains of Consults, Laboratory, and Pharmacy. After preprocessing, over 16 million EHR sequences were obtained from these three domains. The three datasets are summarized in Table 3.1.

(1) **Consults** dataset includes 2,204,523 sequences, 48 unique EHR events, and 1,048,575 unique sequence patterns. Anomalies are randomly selected from rare sequences (sequence pattern frequencies equal to one) for testing purposes, with 21,399 anomalies selected in total.

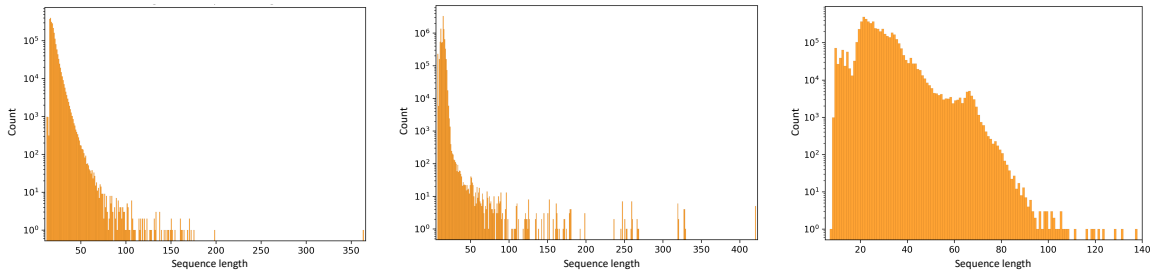
(2) **Laboratory** dataset consists of 5,406,801 sequences, 37 unique EHR events, and 158,391 unique sequence patterns. Anomalies are selected in the same manner as in the Consults dataset.

(3) **Pharmacy** dataset comprises 9,138,291 sequences extracted from patients, with 58 EHR events and 1,269,780 sequence patterns.

In Figure 3.4, we present the distribution of sequence length and the distribution of sequence pattern frequency in the three datasets. Figures 3.4a, 3.4b, and 3.4c show that the majority of sequences have less than 100 events, with over half of them having less than 50 events. These distributions help to determine the sliding window size for the model. Typically, the sliding window should be set to be no less than half the length of the majority of sequences for computational efficiency, but not too large, as short sequence correlations may be obscured by excessive padding tokens.

Table 3.1: Detail of EHR datasets.

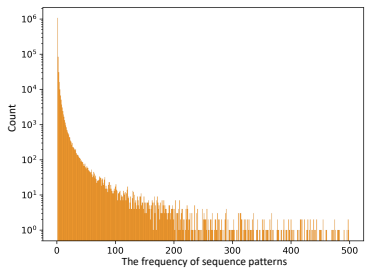
Datasets	**# of sequences**	**# of unique events**	**# of patterns**	**# of anomalies**
Consults	2,204,523	48	1,048,575	21,399
Laboratory	5,406,801	37	158,391	15,270
Pharmacy	9,138,291	58	1,269,780	19,450



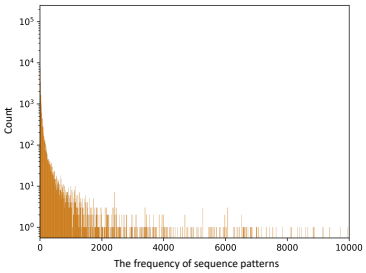
(a) Consults: histogram of sequence length

(b) Lab: histogram of sequence length

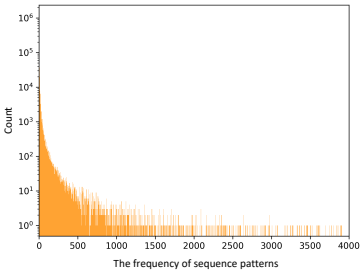
(c) Pharmacy: histogram of sequence length



(d) Consults: histogram of sequence pattern frequencies



(e) Lab: histogram of sequence pattern frequencies



(f) Pharmacy: histogram of sequence pattern frequencies

Figure 3.4: Figure a-c show the histogram of sequence length for three datasets. Figure d-f present the histogram of sequence pattern frequencies for three data sets.

In Figures 3.4d, 3.4e, and 3.4f, the x-axis shows the quantity of sequences that share the same sequence pattern, while the y-axis displays the count of unique sequence patterns with a specific frequency. For instance, in Figure 3.4d, the highest frequency of unique sequence patterns is 498, and there are two unique sequences with that frequency. As shown, the number of unique sequential patterns decreases rapidly with increasing frequency. As ground truth is not available in the data, the data is labeled based on their frequencies for model evaluation. Normal data is labeled using predefined thresholds on the frequency of sequence patterns, while anomalous data is randomly selected from sequences that occur only once. This is based on the assumption that the majority of sequences in the dataset are normal, while anomalies are rare and often not identical. In the experiments, the model is tested using thresholds of 10, 20, 50, 100, and 200 to label the data and differentiate between normal and anomalous sequences. If the frequency of a sequence pattern is higher than the threshold, it is labeled as normal. For each threshold, a random selection of anomalies was made by taking five percent of the data with a frequency of 1. Table 3.2 displays the number of randomly selected anomalies under each threshold in the "# of anomaly" column for each medical domain. In addition, 10 tests are performed for each setting, and the average accuracy is reported.

3.4.2 Baseline Methods

For comparison purposes, five state-of-the-art sequence anomaly detection methods were selected as our baseline methods (as listed below). These include two statistical methods and three deep learning-based methods. Although they were originally proposed for syslog anomaly detection, they can still serve as valuable benchmarks for our task of identifying sequence anomalies in modern software systems.

- **PCA** [113] - This method creates an event count matrix based on the frequency of sequence events and maps it into a lower dimensional space using Principal

Table 3.2: Detail of EHR datasets in evaluations. The columns "# of anomaly" show the number of randomly selected anomalies under each threshold.

Threshold of normal data	Consults (# of)			Laboratory (# of)			Pharmacy (# of)		
	sequences	normal seq	anomaly	sequences	normal seq	anomaly	sequences	normal seq	anomaly
200	130,620	109,221		8,137,638	8,122,368		2,152,809	2,133,359	
100	223,310	201,911		8,356,685	8,341,415		2,529,810	2,510,360	
50	322,008	300,609	21,399	8,536,381	8,521,111	15,270	2,883,680	2,864,230	19,450
20	478,938	457,539		8,724,931	8,709,661		3,305,895	3,286,445	
10	625,023	603,624		8,841,132	8,825,862		3,594,847	3,575,397	

Component Analysis. If a log count vector is significantly different from the normal vector space, it is deemed an anomaly.

- **LogCluster** [58] - This clustering-based method utilizes a knowledge base to eliminate redundancy from previously analyzed log sequences. Log sequences are transformed into event count vectors and represented using Inverse Document Frequency (IDF). Hierarchical clustering is then used to generate two sets of vector clusters (normal and abnormal). The state of a test sequence is determined by calculating its distance to the cluster centroid. If the distance exceeds a threshold, the log sequence is flagged as an anomaly.

- **DeepLog** [30] - DeepLog is a deep neural network model that uses Long Short-Term Memory (LSTM) to model a sequence as a natural language sequence. It predicts the next log events based on the input log sequence. An anomaly is identified if the actual log event or parameter is unexpected compared to the predictions.

- **LogAnomaly** [70] - LogAnomaly is a deep learning method based on the LSTM model and is capable of capturing the semantic information hidden within sequences by incorporating a word embedding layer.

- **LogBERT** [37] - BERT models have demonstrated impressive natural language learning capabilities in numerous applications. Inspired by this, LogBERT learns the inter-event dependencies in normal sequences using a random masked event prediction method.

3.4.3 Experimental Settings

Implementation and Environment. During our evaluations, the hyperparameters for the EHR-BERT model during evaluations were set as follows: a sliding window size of $w = 64$, a hidden vector size of $d = 256$, $L = 4$ transformer layers, each containing $h = 4$ multi-head attention layers, and a mini-batch size of 64. The training process

was stopped after 10 consecutive epochs without any loss improvement. EHR-BERT was implemented using PyTorch V1.11 and tested on NVIDIA Tesla V100 GPUs. Baseline methods were implemented using the open-source toolkit Loglizer [†] for PCA and LogCluster, and logdeep [‡] for DeepLog and LogAnomaly.

Evaluation Designs. The EHR-BERT model is compared with five baseline methods on all three datasets using five different thresholds (10, 20, 50, 100, and 200). Sequences with a frequency larger than the threshold are considered normal data, and 20% of the normal data is used for training. 5% of the sequences with a frequency of 1 are randomly selected as anomalies for testing. Each setting is tested 10 times, and the average accuracy is reported. Table 3.2 provides further details on the EHR data under the different thresholds.

Evaluation Metrics. To assess the performance of the anomaly detection methods, three standard evaluation metrics are used: Precision, Recall, and F-score. These metrics are defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{3.8}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3.9}$$

$$\text{F} - \text{score} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}}, \tag{3.10}$$

where TP, FP, and FN refer to true positive, false positive, and false negative, respectively. Precision represents the percentage of true anomalies among all detected anomalies. Recall shows the percentage of actual anomalies that have been detected. F-score is the harmonic mean of Precision and Recall.

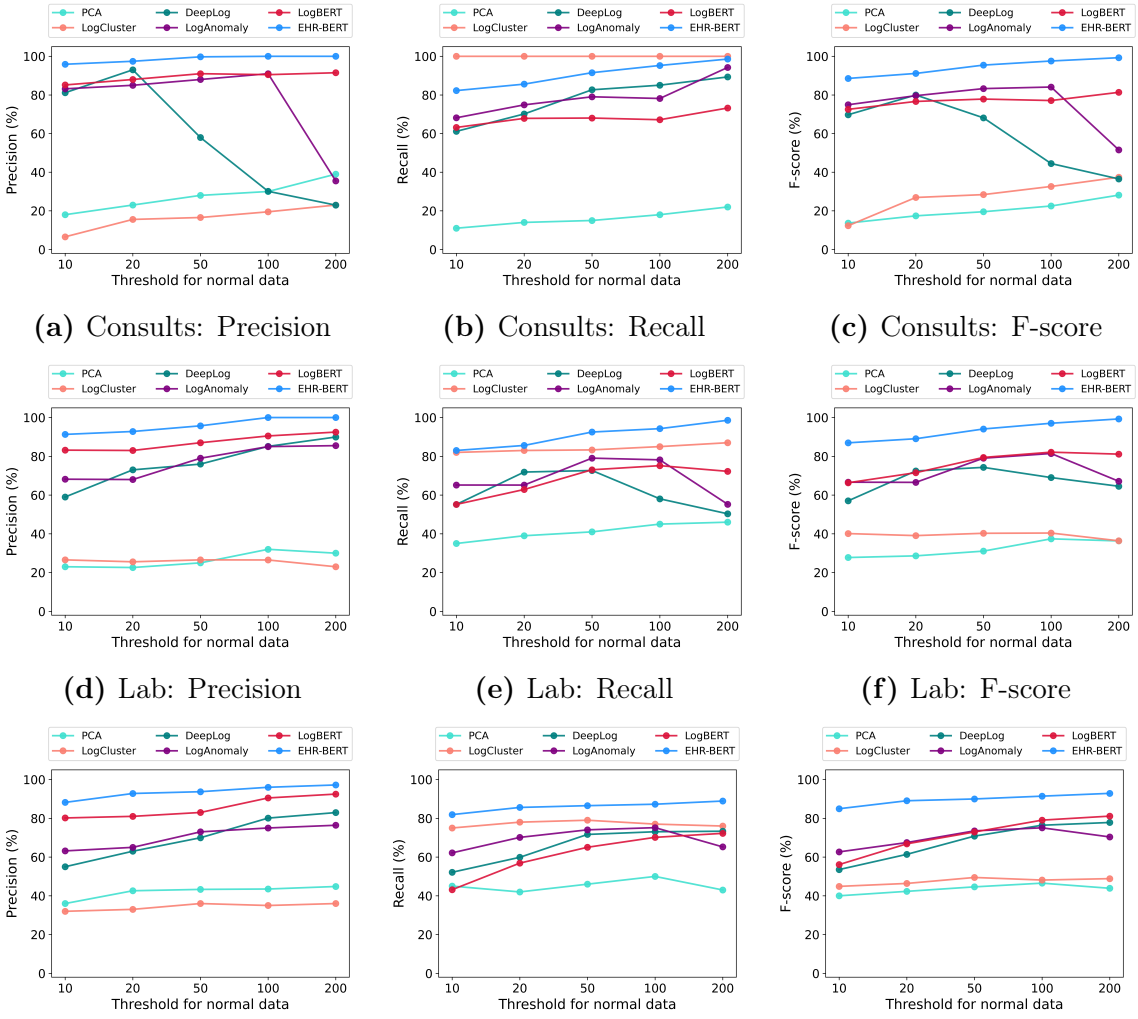
[†] https://github.com/logpai/loglizer
[‡] https://github.com/donglee-afar/logdeep

3.4.4 Experimental Results

In this section, the efficacy of EHR-BERT in real-world EHR data is demonstrated by comparing it with five baseline methods on EHR sequences from three medical domains: Consults, Laboratory, and Pharmacy. Figure 3.5 shows the comparison results of EHR-BERT as well as the five baseline methods on the three datasets with five different thresholds for selecting normal data.

Figures 3.5c, 3.5f and 3.5i display the comparison results between EHR-BERT and five baseline methods across the three datasets with five different thresholds for normal data selection. The results indicate that EHR-BERT outperforms all of the baseline methods, with F-scores ranging from 80% to nearly 99% for the five thresholds in the three datasets. When evaluating the four deep learning-based methods (DeepLog, LogAnomaly, LogBERT, and EHR-BERT), it can be seen that they all perform better than the two traditional statistical methods (PCA and LogCluster). This is because traditional statistical methods tend to lose temporal information, while deep learning-based methods are better equipped to capture the correlations between tokens in sequences, which can be treated as natural language. Among the deep learning-based methods, DeepLog, LogAnomaly, and LogBERT have similar F-scores, while PCA and LogCluster perform poorly, with F-scores ranging from 0.2 to 0.4, much lower than that of EHR-BERT. This highlights the limitations of traditional statistical methods in handling EHR sequence data generated from modern HIT systems, as the temporal information in sequences is lost when converting them into counting vectors. Between PCA and LogCluster, they have similar F-scores, but LogCluster performs better in terms of recall and false negatives. Figure 3.5b demonstrates the recall performance of LogCluster, which almost achieves perfect recall.

When comparing EHR-BERT with LogBERT, similar precision performance is observed. But, EHR-BERT exhibits a 20% to 30% increase in recall compared to



(a) Consults: Precision **(b)** Consults: Recall **(c)** Consults: F-score

(d) Lab: Precision **(e)** Lab: Recall **(f)** Lab: F-score

(g) Pharmacy: Precision **(h)** Pharmacy: Recall **(i)** Pharmacy: F-score

Figure 3.5: Comparison evaluation results on EHR-BERT with five baselines on EHR dataset from three medical domains.

LogBERT. This indicates that LogBERT predicts more false negatives than EHR-BERT during evaluation. This shows that our SMTP strategy greatly improves the performance of anomaly detection tasks when using BERT models.

EHR-BERT outperforms RNN-based models, such as DeepLog and LogAnomaly, in all test cases. EHR-BERT requires less training time than RNN models and achieves the best overall performance on the task without needing a semantic embedding layer like LogAnomaly. This suggests that the Transformer encoder in EHR-BERT is more effective in capturing the patterns in log sequences than RNNs.

As the threshold increases, the F-score of EHR-BERT also increases, which is in line with our expectations. With a larger threshold, anomalies are more distinct from normal data, making the task less complex. On the other hand, in this process, the F-score of DeepLog and LogAnomaly first increases and then decreases. This reduction could be because the models struggle to distinguish between normality and anomalies when there are too few normal sequential patterns for training.

3.4.5 Discussions

Anomaly detection in discrete event sequences is critical for identifying potential issues such as software failures before they escalate. In large-scale computing systems, including HIT, anomaly detection is indispensable for maintaining system reliability, security, and performance. As HIT systems become more extensive and intricate, advanced and effective anomaly detection techniques are necessary to ensure their continued success in various applications. The adoption of HIT in healthcare has increased the quality and efficiency of healthcare services, but it has also created new challenges and vulnerabilities. HIT can impact clinical decisions, disrupt the sequence of care, and change care delivery processes, resulting in patient harm. Though the percentage of severe harm and death resulting from HIT-related incidents seems low, given the high number of incidents, the amount of harm is significant. Recent studies have analyzed the vulnerabilities in HIT and the potential risks associated with its

use. For instance, a decade-long study of HIT-related incidents found that 1% of the incidents caused severe harm, and less than 1% caused death [68]. In another study [67], almost 13% of incidents resulted in patient harm, including 3 deaths.

While there have been some recent progress (for example, [118, 78, 37]), there are still some challenges with anomaly detection in EHR. The EHR-BERT model presents in this chapter is expected to bridge this gap. The performance of EHR-BERT is discussed based on three features: (i) effects of SMTP to the performance of EHR-BERT; (ii) the robustness of EHR-BERT in terms of the diversity of the datasets used for evaluation; and (iii) the generalization of EHR-BERT in terms of its effectiveness in other applications; and (iv) the potential impact of EHR-BERT on clinical practice.

Effects of SMTP: The performance of EHR-BERT model can be attributed to the utilization of the SMTP task to enhance the accuracy of anomaly detection in EHR sequences. Compared to the RMTP task in the traditional BERT model, the SMTP task has proven to be more effective in the EHR anomaly detection task. This indicates that by repeatedly masking input tokens during training and prediction, the SMTP approach can enhance the model's ability to learn normal EHR sequence patterns and reduce false negatives in anomaly detection. This finding has broader implications for sequence anomaly detection tasks since the SMTP method could enhance the performance of BERT-based models in these tasks as well.

Diversity of Datasets: The EHR-BERT model also demonstrated remarkable capability in performing effectively across diverse clinical domains. The model's performance was evaluated on three distinct EHR datasets derived from various clinical domains, including Consults, Laboratory, and Pharmacy, and it surpassed the baseline methods on all three datasets. This indicates that the EHR-BERT model is both robust and generalizable across different clinical domains, positioning it as a promising approach for real-world EHR anomaly detection applications.

Generalization of Applications: Based on the two aforementioned features, it is evident that the approach of the EHR-BERT model for sequence anomaly detection

can be extended to other large-scale computing systems beyond HIT. Specifically, the model treats log sequences as natural language sentences, enabling it to capture contextual information and event correlations. This suggests that the EHR-BERT model has the potential to be effective in other large-scale computing systems. By utilizing the power of machine learning and deep learning algorithms, researchers and system practitioners can fully leverage these critical technology infrastructures.

Potential Impact on Clinical Practice: The EHR-BERT model has the potential to significantly impact clinical practice by improving the detection of anomalies in EHRs. Early identification of anomalies can lead to reduced medical errors and improved patient safety. Moreover, by automating the anomaly detection process, EHR-BERT can save valuable time for healthcare professionals, allowing them to focus on patient care rather than manually inspecting EHRs for potential issues. Furthermore, EHR-BERT could aid in identifying systemic issues in the HIT infrastructure and help prevent data breaches or other security vulnerabilities that may arise from undetected anomalies. By ensuring the integrity of EHRs, EHR-BERT can contribute to enhancing the quality of clinical decision-making and ultimately improve patient outcomes.

The generalizability of EHR-BERT across various healthcare settings is an essential factor in determining its potential impact on clinical practice. Our evaluation on diverse clinical domains demonstrates that EHR-BERT performs effectively across different types of EHRs, making it a promising tool for widespread adoption in the healthcare industry.

In summary, EHR-BERT has the potential to greatly impact clinical practice by enhancing anomaly detection in EHRs, reducing medical errors, and supporting healthcare professionals in their decision-making. The model's superior performance, generalizability, and potential for integration into various healthcare settings demonstrate its promise as a valuable tool for improving patient safety and the overall quality of healthcare services.

3.5 Conclusion

System log anomaly detection plays a crucial role in maintaining and safeguarding modern large-scale software systems, including HIT systems. Deep learning-based methods have demonstrated impressive performance in this task; on the other hand, existing state-of-the-art BERT-based models with RMTP suffer from low accuracy and high false negatives. To address this issue, this work proposed EHR-BERT, a novel self-supervised sequence anomaly detection model based on BERT and a SMTP strategy. Our model treats log sequences as natural language sentences, capturing contextual information and correlations between events. After training on sequences from normal executions, EHR-BERT identifies anomalies when the masked tokens deviate from the model's predictions. Extensive evaluation on large HIT system logs, EHRs, demonstrates the superior effectiveness of EHR-BERT as compared to state-of-the-art baseline methods.

Future work includes evaluating EHR-BERT on different types of large-scale system logs, such as EHRs from various HIT systems and software logs from different applications, to test their generality and scalability. Moreover, there are plans to conduct studies to assess the real-world applicability of EHR-BERT in diverse healthcare settings, further validating its generalizability and clinical impact. By doing so, the aim is to contribute to the adoption and integration of EHR-BERT into healthcare systems worldwide, ultimately improving patient safety and the overall quality of healthcare services.

Chapter 4

System-Level Anomaly Detection in HIT Systems Using High-Dimensional Volume: A Framework Based on Machine Learning and Graph Algorithms *

4.1 Introduction

The detection of traffic anomalies is a crucial issue for contemporary large-scale computer systems, with volume anomalies being a specific and significant example. Anomalies are unusual and significant changes, such as uncommon small or large traffic volumes, that occur within a predetermined period. These anomalies can indicate a broad spectrum of operational issues, including malicious activities like denial-of-service attacks and worm spreads, unusual events, misconfiguration, equipment failures that lead to traffic shifts, or abnormal user or customer behaviors.

*This chapter is from a manuscript in preparation of the same title by Haoran Niu, Olufemi A. Omitaomu, Michael A. Langston, et al.

HIT systems are critical components of modern healthcare facilities. These systems, which store and manage electronic health records, have revolutionized the way patient information is stored, managed, and accessed. On the other hand, the large-scale and complex nature of HIT systems make detecting volume anomalies a challenging task. The volume of data flowing through these systems can vary significantly, making it difficult to define a fixed threshold for what constitutes normal traffic. Additionally, the presence of numerous sub-systems and processes can result in a high degree of variability in traffic patterns. Finally, the system can be affected by various types of noise, such as network congestion or system glitches, further complicating the detection of volume anomalies.

Time-series traffic anomaly detection has been extensively studied using statistical methods such as Wavelet analysis [62, 41, 17] and PCA [117, 82, 18], as well as machine learning methods such as clustering [8, 34, 29], classification[12, 103], and neural networks [51, 74, 53, 108]. Nevertheless, these methods can be challenging to interpret and analyze the root cause of anomalies in high-dimensional data, leading to suboptimal performance in different scenarios. To overcome these limitations, a framework is proposed that combines machine learning models and graph algorithms to detect volume anomalies in HIT systems, enabling system-level hazard detection and fast response. Our framework aims to improve the accuracy and efficiency of system-level volume anomaly detection, while providing meaningful insights to support decision-making for HIT systems.

In this chapter, we present a comprehensive framework for detecting network-wide volume anomalies using machine learning models and graph algorithms. Our proposed framework consists of three main steps, which are data preprocessing, volume outlier detection using a voting strategy on multiple machine learning methods, and system-level anomaly detection based on graph algorithms. The key contributions of this chapter are as follows:

- An approach for volume anomaly detection is designed, employing a voting machine method that combines five different machine learning methods. The evaluation results demonstrated that the proposed method can effectively reduce false positives, avoid the poor performance of a single method, and achieve overall better results in detecting volume anomalies.

- A novel method is proposed to construct weighted graphs that represent the correlation of co-occurring outliers using simultaneous anomaly similarity as edge weight. This approach enables us to capture the complex relationships among volume anomalies and identify potential clusters of anomalies, which can be further investigated to uncover the underlying causes of the anomalies.

- The proposed framework utilizes advanced graph algorithms, including spectral graph theory for threshold selection and paraclique algorithm for cluster identification. By leveraging these techniques, our approach can effectively capture the correlation and structure of co-occurring volume anomalies, leading to more accurate and reliable detection of system-level anomalies.

- A case study is performed using real-world data from a large-scale healthcare operation to validate the effectiveness of the proposed framework. The evaluation results showed that the framework was able to successfully identify anomalous events and provide meaningful insights.

Overall, the proposed framework provides a practical solution for network-wide volume anomaly detection and has the potential to be applied in a wide range of applications, including healthcare, cybersecurity, and financial fraud detection.

The remaining part of this chapter is organized as follows. Section 4.2 provides an introduction to related works in the field of network-wide volume anomaly detection in large-scale computer systems. Section 4.3 details the proposed framework and methods used in this study, including data preprocessing, voting machine, and system-level anomaly detection based on graph algorithms. Section 4.4 presents the

evaluations conducted to test the effectiveness of the proposed framework, including an evaluation of the voting machine method and a case study using real-world data. Finally, section 4.5 summarizes the key findings of the chapter.

4.2 Related Work

Over the past decade, there has been significant research on network-wide volume anomaly detection. Despite this, the problem of detecting anomalies in modern systems remains a major challenge due to their increasing complexity and the potential for advanced attacks against distributed deployments.

4.2.1 Statistical Methods

Statistical-based anomaly detection methods are commonly employed to identify network-wide volume abnormalities. These techniques typically involve the use of probabilistic models and training data to accomplish the task. Wavelet analysis and PCA are popular methods used in this area.

Wavelet analysis is a commonly used method in signal analysis to handle signals that vary in amplitude and frequency over extended periods of time. Studies such as [62, 41, 17, 97] have employed wavelet analysis to detect volume anomalies in network traffic. Soule et al. [97] proposed a method that uses the output signals of a wavelet transform to detect volume anomalies and compares the local to the global variance on the residual signal. Lu and Ghorbani [62] proposed a real-time method for anomaly detection using wavelets combined with system identification theory. Nonetheless, using wavelet-based methods for network-wide anomaly detection poses several challenges, including sensitivity to the choice of wavelets and decomposition levels, high computational cost, and difficulties in interpretation [119].

PCA is another commonly used statistical technique for detecting volume anomalies [56, 90, 117, 60, 82, 18]. It is a dimensionality reduction approach that

transforms high-dimensional volume data into a lower-dimensional space. Lakhina et al. [56] first addressed the problem of anomaly diagnosis in network-wide traffic using PCA to efficiently separate traffic measurements into normal and anomalous subspaces. Pascoal et al. [82] employed a robust PCA detector combined with a robust feature selection algorithm to adapt to distinct network contexts and circumstances. Camacho et al. [18] utilized a PCA-based multivariate statistical process control (MSPC) approach that monitors both the Q-statistic and D-statistic. Nevertheless, PCA-based methods may not be effective for detecting nonlinear anomalies since they assume a linear relationship between the variables.

4.2.2 Machine Learning-Based Approaches

In addition to statistical-based methods, machine learning techniques have also been applied to volume anomaly detection in recent years. These approaches aim to capture complex patterns and relationships in the data that may be difficult to model using traditional statistical methods alone. This subsection briefly reviews some of the recent literature on machine learning-based volume anomaly detection, including clustering, classification, and neural network methods.

Clustering techniques are frequently utilized to group objects into classes of similar objects and can be adapted to identify outliers by identifying values that are dissimilar to any cluster. Numerous studies, such as [86, 8, 34, 29], have utilized clustering methods to detect volume anomalies in network. For instance, Rajasegarar et al. [86] used the average inter-cluster distance of KNN clusters to identify anomalous clusters, while Flanagan et al. [34] proposed a modified version of the MCOD algorithm using both distance-based outlier detection and cluster density analysis. Still, clustering methods may suffer from the “curse of dimensionality", making them ineffective when applied to high-dimensional data, leading to poor performance and difficulty in identifying meaningful clusters [2, 103].

Classification techniques are extensively used in the field of anomaly detection [12, 103]. These methods involve the construction of a classifier during the training phase using labeled training data, followed by the classification of instances as normal or anomalous during the testing phase. Bayesian methods are a common approach for classification in this field, as they provide a probabilistic framework that allows for uncertainty and prior knowledge to be taken into account [54, 99, 46]. Additionally, SVMs have been shown to be effective in detecting anomalies in network traffic [69, 50, 106]. SVMs use a nonlinear transformation to map the data into a high-dimensional feature space, where a hyperplane is used to separate normal and anomalous instances. Ensemble approaches, which involve combining the responses of multiple classifiers into a single one, have been utilized in various studies to improve the performance of anomaly detection [76, 1, 96, 35]. These methods have been shown to achieve better results than using individual classifiers alone.

In recent years, artificial neural networks have been increasingly used for volume anomaly detection, with various studies exploring their capabilities [51, 74, 53, 108]. These works have investigated the potential of neural networks in detecting anomalies, such as outliers and unusual patterns of traffic data in different applications. For example, Kieu et al. [51] utilized autoencoders based on convolutional neural networks and long short-term memory neural networks to detect outliers in time-series data. Similarly, Naseer et al. [74] explored the suitability of different deep neural network structures, including convolutional neural networks, autoencoders, and recurrent neural networks, for anomaly-based intrusion detection systems. Kim and Cho [53] proposed a C-LSTM neural network that combines a convolutional neural network, long short-term memory, and deep neural network to effectively model the spatial and temporal information contained in one-dimensional traffic data. Recently, Wu et al. [108] demonstrated the effectiveness of graph neural networks for anomaly detection in IoT-enabled smart transportation, smart energy, and smart factory applications. These techniques can be difficult to interpret, however, and analyzing the root cause of anomalies can be challenging in the context of high-dimensional data.

4.3 Framework Design

In this section, the proposed framework is presented for system-level anomaly detection in HIT systems using high-dimensional volume data. The framework consists of three main components: data preprocessing, time-series anomaly detection based on voting machine and graph-based anomaly correlation analysis.

The time-series anomaly detection component leverages state-of-the-art machine learning algorithms to detect anomalies in the time-series volume data collected from each station in the HIT network. By using an ensemble of multiple algorithms, our framework can effectively capture different types of anomalies and achieve over all better results.

The graph-based anomaly correlation analysis component uses advanced graph algorithms to identify clusters where stations have high correlations between simultaneous anomalies detected at different volume. By constructing a graph where each vertex represents a station and edge weights represent the similarity of concurred anomalies between two stations, clusters of stations with similar patterns of anomalous behavior can be identified . This enables the detection of system-level anomalies that may be difficult to detect using only station-level data.

Overall, the framework integrates state-of-the-art techniques from both time-series anomaly detection and graph analysis to provide a robust and effective solution for spatial-temporal volume anomaly detection in network-wide HIT systems.

4.3.1 Overview

Figure 4.1 presents the design of the proposed framework, which consists of three main steps: data processing, anomaly detection, and graph analysis.

The data processing step involves preprocessing the raw data collected from each station in the HIT network to generate time-series volume data that can be used in the subsequent steps of the framework. Specifically, the raw data is aggregated into volume data under different categories in each station, such as nursing, emergency,

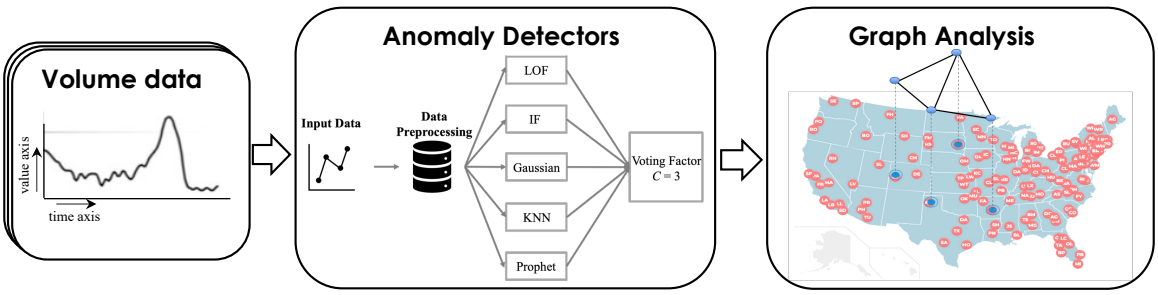


Figure 4.1: Framework of volume anomaly detection.

outpatient, inpatient, etc. This preserves the characteristic information of each category in the data and enables us to identify anomalies at a more granular level. Then, a series of preprocessing steps is applied to the volume data, including filtering, normalization, etc. These steps help to remove noise and ensure that the data is consistent and of high quality. The data processing step is a critical part of our framework for spatial-temporal volume anomaly detection in network-wide HIT systems. By aggregating the raw data into volume data and applying preprocessing steps, high-quality time-series data is generated that can be used to effectively detect and analyze volume anomalies.

The anomaly detection step involves applying five machine learning methods to the preprocessed time-series data to detect anomalies at each station under each category. Each method is designed to capture different types of anomalies, and the choice of method depends on the characteristics of the volume data and the specific scenario. To ensure robust detection performance, we use an ensemble method that combines the results of the five individual methods. Specifically, we use a voting scheme where each method is given equal weight and the final anomaly decision is based on a majority vote. This approach helps to avoid false positives and increases overall detection accuracy. In addition, the ensemble method allows us to handle scenarios where one or more individual methods fail to detect anomalies. In such cases, the voting mechanism can still identify anomalies by relying on the output of other methods. Overall, the anomaly detection step is a critical component of our framework for spatial-temporal volume anomaly detection in network-wide HIT systems. By employing a combination of machine learning methods and an ensemble approach, the detection of anomalies at each station is enhanced, resulting in improved overall detection accuracy and reduced false positives.

The graph analysis step involves constructing a graph where each vertex represents a station in the network and edge weights represent the deviation between correlated anomalies and their mean value of the lookback period between two stations. The lookback period is a user-defined parameter that specifies the time range over

which the mean value is calculated. Then, The edge weights are utilized to quantify the degree of similarity between two stations in terms of their anomalous behavior. Subsequently, a graph thresholding method based on spectral graph theory is employed to remove edges with low weights and construct a graph where each vertex represents a station and the edges connect pairs of highly correlated stations. Advanced graph algorithms then are leveraged, such as paraclique detection, to identify clusters of highly correlated stations with similar anomalous behavior. Analyzing these clusters provides insights into the root causes of anomalies and helps identify system-level anomalies that impact multiple stations.

Together, these three steps form a comprehensive framework for spatial-temporal volume anomaly detection in network-wide HIT systems. Our framework combines state-of-the-art techniques from both time-series anomaly detection and graph analysis to provide a robust and effective solution for detecting and analyzing volume anomalies in complex HIT systems.

4.3.2 Data Preprocessing

In HIT systems, individual EHR data is stored in the system and there is typically no volume data for EHR available. The volume data is generated by aggregating and counting the EHR data entered into the system across various categories. The data is then sorted temporally to create time-series volume data for each category at each station in the network. This preprocessing step is necessary to enable the subsequent analysis of volume anomalies in the HIT system.

Formally, the HIT network is represented as a set of stations denoted by $\mathbf{S} = \{s_1, s_2, ..., s_n\}$, where n is the total number of stations. At each station, there are different types of volumes or traffics that are collected over time representing as $\mathbf{V}^k = \{V_1^k, V_2^k, ..., V_m^k\}$. Here, k denotes the station, and m represents the total number of unique type of volumes collected at station k. By processing the raw

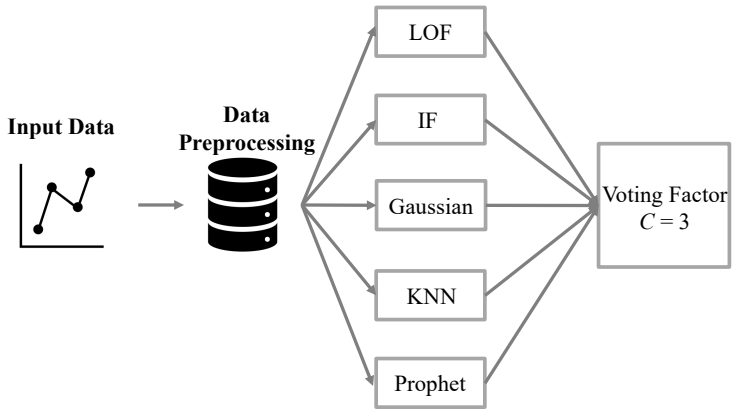
data in this way, time-series volume data is generated that captures the characteristic information of each category of traffic at each station.

In order to reduce noise in the data, we filtered out volumes that contained more than 40% missing values, as these volumes are less reliable for further analysis. Following this step, we normalized all volumes to the same scale using min-max scaling. This not only facilitates comparative analysis across stations and categories, but also enables robust detection without using different sets of parameters in different situations.

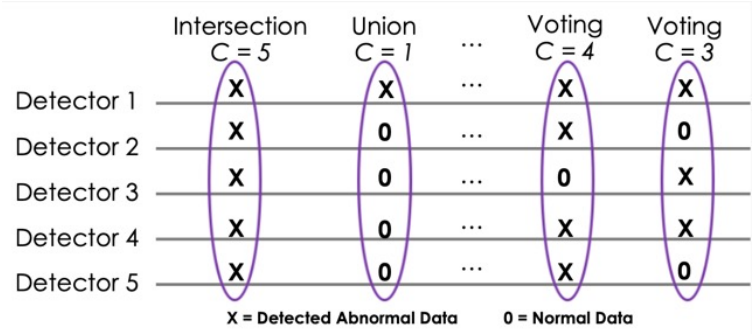
4.3.3 Voting Machine

Figure 4.2 shows the two main components of the anomaly detection step in our proposed framework. Figure 4.2a illustrates the diagram of the “voting machine" approach, where a computational task is solved by using different methods and then combining the detection results of these computations. Specifically, preprocessed time-series volume data is fed into five machine learning models: Local Outlier Factor (LOF), Isolation Forest (IF), Gaussian Mixture Model (Gaussian), K-Nearest Neighbors (KNN), and Facebook Prophet (Prophet). The idea behind the voting machines is that it generates an aggregated view over a decision of multiple detectors which potentially have different weaknesses and advantages. Due to the aggregated vote, these weaknesses are minimized, and the majority vote leads to better results [33].

Figure 4.2b illustrates the voting strategy used in our framework under different voting factors. For a certain time period, a data point classified as an anomaly is represented with an X, while normal data are represented with a 0. Different voting factors can be chosen to determine the threshold for identifying anomalies, where a voting/control factor C represents the minimum number of voters that need to identify a data point as an anomaly for it to be considered as such. In the example shown in the figure, when $C = 3$, at least three of the detectors need to predict the



(a) Diagram of voting machine.



(b) Voting strategies.

Figure 4.2: Diagram showing the voting machine algorithm for combining results from multiple anomaly detection models.

data point as abnormal to be considered an anomaly. By adjusting the voting factor, the trade-off between detection sensitivity and specificity can be controlled.

Formally, a preprocessed time-series volume data is given as $V = \{v_1, v_2, ..., v_n\}$, where n is the number of input data and timestamps over a time window, and v_i represent volume under timestamp i. The data will be fed into the five anomaly detectors, which they produce detection results individually. By given the parameter p and input data V, prediction results can be given by detector j as $P_j(p, F) = [r_1, r_2, ..., r_n]$, where where $r = 0$ if the data is normal, or $r = 1$ if the data is an outlier under each timestamp. Thus, the voting results over base detectors are calculated as $y = \mathrm{VT}(P_1, P_2, ..., P_n)$, where the function VT could be defined as:

$$\mathrm{VT}(P_1, P_2, ..., P_n) = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} P_j > C \\ 0 & \text{otherwise} \end{cases}$$

This function outputs 1 if the number of models predicted a positive outcome is larger than control factor C, and 0 otherwise. Together, Figure 4.2 highlights the voting machine approach for combining the results of multiple machine learning methods to improve overall anomaly detection accuracy.

The five base anomaly detectors used in the framework are discussed as following:

- **Local Outlier Factor (LOF).** The local outlier factor [14], which is based on the local density, represents the degree of being outlier in this approach. It is an unsupervised, density-based method for detecting outliers that differ significantly from their local neighborhoods. It defines the local density of a data point as the number of other points within a certain radius, and then compares this density to the densities of its k-nearest neighbors. If the density of the point is much lower than the densities of its neighbors, it is considered to be an outlier. One advantage of using the LOF method is its ability to capture outliers that are missed by traditional methods. Specifically, the LOF method can identify outliers that have a short distance with their neighbors but a lower local density compared to their neighbors. This is beneficial for detecting

anomalies that may be more subtle or unique compared to other methods. On the other hand, LOF can be sensitive to the choice of the parameter k, which is the number of neighbors used in the distance calculation. In addition, it may not be effective in detecting rare or novel anomalies that are significantly different from the normal data distribution.

- **Isolation Forest (IF) Detector.** The Isolation Forest (IF) [59] is a popular unsupervised anomaly detection algorithm that leverages the concept of decision trees. An IF model consists of an ensemble of binary decision trees, where each tree in the model is called an "isolated tree". The algorithm works by randomly selecting a sub-sample of the data and assigning it to a binary tree, with the branching of the tree performed on a random threshold. The data is then split into left and right branches by the selected threshold. This process is then recursively repeated until all data points are isolated or the maximum depth is reached. Anomaly scores are assigned to each data point based on the depth of the tree required to arrive at that point. The score is an aggregation of the depth obtained from each of the trees. The advantage of the IF approach is that it can efficiently isolate anomalies by taking advantage of their unique characteristics, such as having fewer instances and being located in low-density regions of the feature space. But, the method requires fine-tuned parameters to ensure that it can perform well in detecting anomalies.

- **Gaussian Anomaly Detection.** Gaussian anomaly detection is a common approach for detecting anomalies [9]. In this method, data are modeled as a Gaussian distribution, and the cumulative distribution probabilities of each data point are given by a Gaussian distribution function. If the probability of a data point is below or above a particular threshold, then the data will be detected as an anomaly. The method is easy to interpret, has low computation time, and can achieve fair performance. Despite that, it lacks consideration for the temporal order of data, which could lead to information loss.

- **K-Nearest Neighbor (KNN).** The K-Nearest Neighbor (KNN) approach is based on a similarity measure that calculates the distance of the k-th nearest neighbor from the data point [31]. The distance reflects the sparseness of the neighborhoods of a data point, with larger nearest neighbor distances typically indicating more sparse neighborhoods and a higher likelihood of outliers. In this approach, the number of nearest neighbors is chosen by using a validation dataset. A threshold is then used to determine whether a data point is an anomaly or not. One advantage of the KNN method is its simplicity and interpretability. Yet, its performance may be affected by the choice of k, and it does not consider the temporal order of the data as well.

- **FBProphet.** While the above methods have their advantages, they are all lack of consider the time series factor, which may contains periodic changes and trends. Prophet is a time-series forecasting model developed by Facebook's data science team [101]. It utilizes an additive regression model with a piecewise linear trend, annual and weekly seasonality, and holiday effects to forecast future values of a time-series. They are given as:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t, \tag{4.1}$$

where $g(t)$ is trend function for non-periodic changes in the time series, $s(t)$ models weekly and yearly periodic changes in the data, $h(t)$ is the holiday function which models the irregular changes in the data, and ϵ_t is a normal distributed error function representing the changes that can't be modeled by previous functions. In the context of anomaly detection, Prophet can also be used to identify anomalies as data points that deviate from the predicted values. The advantage of Prophet is its ability to handle missing values and outliers in the time-series data. Nonetheless, Prophet can be computationally intensive and requires careful tuning of its hyperparameters.

In time-series anomaly detection, it is important to consider the temporal order of the data as anomalies could manifest themselves over time. But, some of the base anomaly detectors used in our framework, such as LOF, IF, Gaussian, and KNN, do not inherently incorporate temporal information. This can lead to information loss, especially when dealing with large datasets over a long period of time, where local outliers may be overlooked. To address this issue, it is common to apply a sliding window technique to these models and incorporate time-series information accordingly. In our work, a similar strategy is implemented with a sliding window approach to capture temporal patterns and ensure accurate anomaly detection results. By doing so, it is able to effectively overcome the limitations of the individual anomaly detection methods and improve the overall performance of the framework.

In the framework, each method has its own strengths and weaknesses. By combining them in an ensemble approach, the limitations of individual methods can be overcome, leading to more accurate and robust results. LOF is effective at detecting local anomalies, but it may miss global anomalies. IF is good at detecting anomalies in high-dimensional data, but it may not perform well when the data distribution is irregular. Gaussian can detect anomalies with a simple interpretation, but it may not perform well with data that does not follow a Gaussian distribution. KNN is good at detecting outliers with sparse neighborhoods, but it may not be effective with dense data. Prophet can capture seasonality and trends, but it may not be suitable for detecting abrupt changes in the data.

By combining these methods, the limitations of each individual method can be offset, resulting in a more accurate and robust anomaly detection system. The ensemble approach allows us to take advantage of the strengths of each method and mitigate their weaknesses. For example, LOF and KNN are effective at detecting local anomalies, while Gaussian and Prophet are better at capturing global patterns. IF is effective at handling high-dimensional data, which is often encountered in HIT systems. Moreover, the ensemble approach allows for a reduction in the number of false positives and an increase in overall detection accuracy. This is achieved by

setting a threshold for the number of detectors that need to identify a data point as an anomaly, which can be adjusted based on the specific application and system requirements. In summary, the combination of these five anomaly detection methods in our ensemble approach allows us to achieve a more accurate, robust, and flexible volume anomaly detection system for HIT systems.

4.3.4 Graph Anomaly Detection

The Voting machine can detect outliers in each volume for all stations, but not all detected outliers are caused by system hazards; they may result from regular changes in stations. Regular movements are assumed to be random and are likely to be standalone noises, which are not significant in identifying system-level anomalies. Moreover, the correlations of outliers between stations have not been considered yet, which is crucial in identifying system-level anomalies.

Therefore, graph algorithms are utilized to identify correlations among outliers across stations and eliminate isolated noise. This section focuses on the development and implementation of graph algorithms for detecting network-wide anomalies in the volume data. There are three steps to our approach:

1. **Graph construction.** Firstly, the volumes between stations are compared based on the outliers detected by the voting machine, and each station is represented as a node in the graph. The edge weight is determined using the simultaneous outlier similarity, which is the sum of the deviation similarities between outliers occurring around the same time or within a certain time interval for the same types of volumes. This way, a graph is constructed where each node represents a station in the network, and edges represent the similarity of co-occurred outliers between two stations.

2. **Threshold selection.** After building the graph, only the most significant relationships are retained by setting a proper threshold. In this way, relatively low edge weights edges are removed and it would help reduce noises in further

steps. Nonetheless, this threshold application step is a major juncture in which errors can be introduced in the form of both false negatives and false positives. By setting this threshold too high or too low, important relationships can be lost or buried. Therefore, careful consideration is necessary to strike a balance between preserving important relationships while minimizing false alarms. Here, the spectral graph theory method is applied due to its effectiveness and it is a data-driven approach that does not rely on any pre-defined parameters.

3. **Graph Decomposition.** After applying the thresholding method on the graph, a graph is obtained without relatively low edge weights. The clusters in the graph can represent the stations that are highly correlated with each other in terms of simultaneous anomalies between their volumes. Here, paraclique algorithm [40] is leveraged to find those clusters. A paraclique is a near clique that is not strict to complete subgraphs, so it can identify more nodes with high connectivity to cliques but with missing edges.

Graph Construction

After the outliers for the volumes of each station are detected using the Voting machine, the next step is to identify correlations between them in order to detect system-level anomalies. Graph algorithms provide a natural way to represent and analyze the relationships between stations. The steps are presented to build the graph based on the co-occurrence of outliers between stations.

To represent a set of stations as a graph, the notion of simultaneous anomalies or co-occurring outliers is defined. Two outliers from the same type of volume data of two stations are treated as simultaneous anomalies if their time difference is less than Max Defer Time (MDT). MDT is a hyperparameter that determines the maximum time difference between two simultaneous anomalies that can be considered for calculating their similarity.

Next, the concept of aggregated simultaneous anomaly similarity (SAS) is defined as the edge weight in the graph. The edge weight reflects the sum of the deviation similarities between any pairs of simultaneous anomalies between two stations. SAS measures the similarity of two co-occurring outliers from the same type of volume data of two stations, and is calculated as the product of the logarithm of the absolute difference between the outlier and the mean value of its lookback period volume for each station. Specifically, it is given as:

$$\mathrm{SAS}(o_i, o_j) = \log\left(\left|\mathbf{o}_i - \overline{\mathbf{v}_i^{\mathrm{lbp}}}\right| + 1\right) \cdot \log\left(\left|\mathbf{o}_j - \overline{\mathbf{v}_j^{\mathrm{lbp}}}\right| + 1\right), \tag{4.2}$$

where o_i and o_j represent the value of a pair of simultaneous anomalies for two stations, and $\overline{\mathbf{v}_i^{\mathrm{lbp}}}$ is the mean value of the lookback period volume of outlier i. The lookback period is a historical period preceding the occurrence of an outlier that is used to obtain the normal value of the volume data in a specific time period. To obtain an accurate historical mean value without being biased by outliers, any identified outliers in the period are excluded from calculating the mean value. The lookback period is another hyperparameter that determines how far back in historical data the outlier is compared with. The SAS increases as the outliers deviate further from their respective average values.

The edge weight between two stations is calculated by considering all common volumes between them. For each common volume, the sum of the SAS values is computed between all pairs of co-occurring outliers from the same type of volume data in the two stations. The resulting sum represents the total SAS between the two stations, which is given as:

$$\mathit{weight}(S_i, S_j) = \sum\nolimits_{v \in V_{ij}} \sum\nolimits_{(o_i, o_j) \in \mathrm{zip}(O_i^v, O_j^v)} \mathrm{SAS}(o_i, o_j), \tag{4.3}$$

where O_i^v and O_j^v represent the set of simultaneous anomalies between two stations i and j for a specific volume type v. The scaled total SAS, obtained after min-max

scaling, is used as the edge weight in the graph to reflect the similarity of co-occurrence of outliers between two stations.

Algorithm 3 presents the pseudocode for building such a graph. An empty graph G is initialized and iterated through all pairs of stations i and j in the input set. For each pair, it finds the common volumes V_{ij} between the two stations and initialize the edge weight w_{ij} to zero. Then, it iterates through each common volume v_{ij} and finds the outliers O_i and O_j for that volume in station i and station j, respectively. To reduce the computation time, it sorts the outliers in each station by date, then use a sliding window of size $maxDeferTime$ to find co-occurred outliers between the two stations. For each pair of outliers, their simultaneous outlier similarities are computed using Equation (4.2) and add the value to the edge weight w_{ij}. After iterating through all common volumes and outlier pairs, it adds the edge (i, j) with the updated weight w_{ij} to the graph G. Finally, it returns the fully constructed graph.

Algorithm 3 Building Graph

```
1:  procedure BuildGraph(stations, maxDeferTime)
2:      Initialize empty graph G
3:      for all pairs of stations i, j ∈ stations do
4:          V_i ←VType(station_i)          ▷ a list of volume types in station i
5:          V_j ←VType(station_j)          ▷ a list of volume types in station j
6:          V_ij ←set(V_i + V_j)           ▷ common volumes in the station pair
7:          w_ij ← 0                       ▷ initialize edge weight
8:          for all common volumes v_ij ∈ V_ij do
9:              O_i ← outliers of v_ij in station_i
10:             O_j ← outliers of v_ij in station_j
11:             for all pairs of outliers (o_i, o_j) ∈ zip(O_i, O_j) do
12:                 if |o_i.date − o_j.date| < maxDeferTime then
13:                     w_ij+ = SAS(o_i, o_j)   ▷ calculate simultaneous outlier similarity
    using Equation (4.2)
14:         Add edge (i, j) with weight w_ij to graph G
15:     return G
```

Threshold Selection

The next step after building a graph for the outliers detected by the voting machine is to select a threshold using spectral graph theory [83]. This method is systematic and does not rely on any pre-defined parameters. Spectral graph theory computes the eigenvalues and eigenvectors of adjacency matrices of graphs [24] constructed at various threshold values, which can provide information about graph connectivity. Specifically, the smallest nonzero eigenvalue shows the algebraic connectivity of the largest connected component of a graph. Lower connectivity values indicate the presence of nearly-disconnected components [28]. Therefore, the threshold value that minimizes the number of edges connecting dissimilar clusters is chosen as the cutoff. The idea behind this is to find the threshold that makes the clusters in the graph more distinct.

Graph Decomposition

In this step, the paraclique algorithm [22] is applied to decompose the graph after thresholding into a set of clusters, which those clusters reveal the stations that have high simultaneous outlier similarities. In an undirected graph G, a clique is a subgraph of G in which every pair of vertices is connected by an edge. A paraclique is an expanded version of a maximum clique [11], which is a clique that includes the largest possible number of vertices in a graph. Specifically, a paraclique is formed by adding non-clique vertices that are adjacent to most, but not all, nodes of a maximum clique. The motivation for using paracliques is to identify those vertices that are highly correlated with the dense graph but have missing edges and are unchecked by the maximum clique. The missing edges are often lost due to noise, improper thresholding, poor experimental processes, and other factors [40, 39].

Formally, a glom number g is used to control the adding non-clique vertices in to a paraclique, P. A non-clique vertex was chosen if and only if it was adjacent all but at most g vertices in P [40].

In summary, Graph Anomaly Detection is a three-step process for detecting anomalies in volume data. The first step involves building a graph to represent the relationships between different stations based on the similarity of their co-occurring outliers. The second step is to select a threshold for the graph using spectral graph theory to identify the most anomalous clusters. Finally, the third step involves using paraclique analysis to identify highly correlated vertices that may have been missed by the previous steps. The resulting clusters and vertices can then be further analyzed to gain insights into potential anomalies in the volume data.

4.4 Evaluations

This section showcases two evaluations of the proposed framework using data sourced from a HIT system implemented in a large-scale healthcare operation. The first experimental evaluation assesses the performance of the Voting Machine method, where four types of common anomalies are randomly injected into the extracted EHR data under multiple settings.

To evaluate the framework's overall ability in detecting system-level anomalies, a case study is performed where the operation performed a network-wide HIT software system upgrade in mid-2018. The event is treated as a system hazard and use it as ground truth for our evaluation, assuming that the system has changed without our knowledge. Our results demonstrate promising performance in both evaluations.

4.4.1 Datasets

The study extracted volume data from the HIT system of a large-scale healthcare operation. The data consisted of the number of daily orders entered into HIT systems for different service sections across 130 medical facilities, from 2017-2019. Each order represented a record for each patient, resulting in a total of roughly 90 million data points across 4,270 different service sections.

After combining the variances of services with the same name, 2,615 unique service section types were left. The most frequent service sections included primary care, surgery, medical, medicine, nursing, mental health service, ambulatory service, pharmacy, and more.

The data was converted into daily time-series volume data for each station under each service section for the three-year period. Holidays and weekends were removed from the data to reduce noise. Next, a filter was applied to drop the volume data with high sparsity, using 60% as a cutoff, and removing volumes where the sparsity exceeded 60%. Overall, 100,054 time-series volume data was processed from 130 medical facilities for our evaluation.

4.4.2 Evaluation on Voting Machine

Data Injection

The proposed Voting Machine was evaluated on the extracted time-series datasets by injecting anomalies into the data. The four typical types of volume deviations, which represent the most common abnormal observations in HIT systems [79], are shown in Figure 4.3. The deviation may normalize after a short time without any intervention (reoccurring), occur abruptly (sudden) or incrementally (incremental), with oscillation gradually leading to constant abnormalities (gradual), or a combination of these occurrences.

There were 20 data points generated for each type of anomaly to simulate four weeks of working days in the time-series datasets. High values in the data indicate abnormal behaviors, and these anomalies were randomly generated between 0.1 and 0.2, while the low values ranged from 0.8 to 0.9. The `sudden` type shows an abrupt increase in the middle of the data without going back to normal, while the `reoccurring` type shows an abrupt increase at the start of the second quarter of the data and returns to normal in the fourth quarter. The `gradual` type has an oscillation period before stabilizing at an abnormal volume level, and the `incremental` type

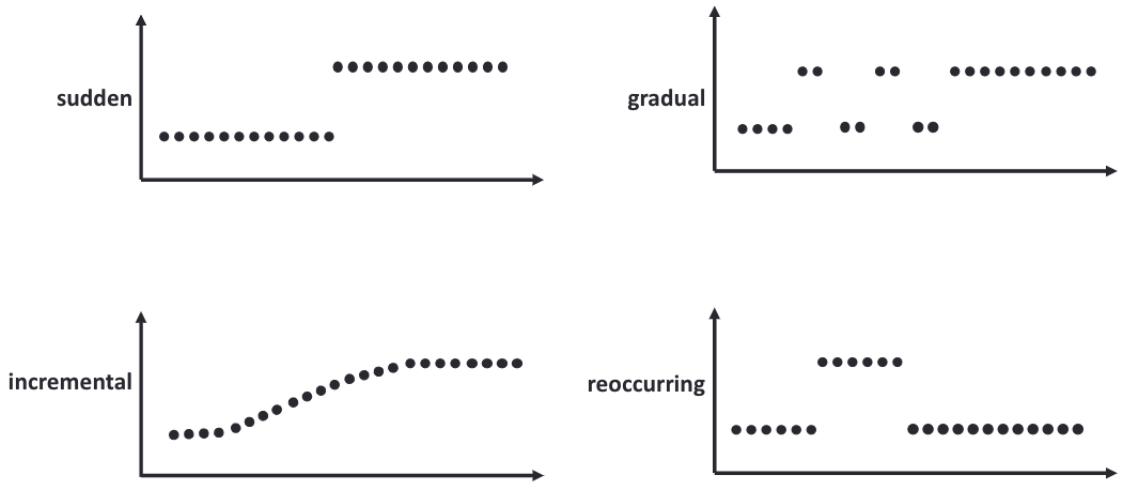


Figure 4.3: Depictions of four common volume deviations in HIT systems. Each dot denotes a data point.

shows a gradual transition from low to high values in the second and third quarters of the data. To simulate real-world situations, a 20-day period in VA's volume data were randomly replaced with the scaled anomalies, ranging from [mean, mean+k*std]. 1,000 volumes were randomly selected from the time-series dataset and each of the four types of anomalies was injected for three different values of k (1, 2, and 3).

Performance Metrics. To evaluate the performance of the proposed Voting Machine method and individual detectors, we calculated Precision, Recall, and F-score under each setting:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4.4}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4.5}$$

$$\text{F} - \text{score} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}}, \tag{4.6}$$

where TP stands for true positive, FP for false positive, and FN for false negative. Precision measures the percentage of true anomalies in all detected anomalies. Recall measures the percentage of real anomalies detected. F-score is the harmonic mean of Precision and Recall.

Experimental Settings

As previously mentioned, experiments were conducted using three values for k: 1, 2, and 3. For each value, 1,000 time-series volumes were randomly selected and each of the four types of anomalies was individually injected, resulting in a total of 4,000 volumes for testing. Under each setting, 10 runs were performed. The sliding window size was set as 1 month for LOF, IF, Gaussian, and KNN, and the voting factor C was set to 3 during the process. The hyperparameters of each machine learning detector were tuned based on a validation set.

Experimental Results

The evaluation results presented in Table 4.1 demonstrate the performance of the Voting Machine and five individual detectors (LOF, IF, Gaussian, KNN, and Prophet) in detecting anomalies injected into the time-series volume data. The results are reported using precision (P), recall (R), and F-score (F) for each of the four types of injected anomalies under three settings (k=1, 2, and 3).

The evaluation results presented in Table 4.1 demonstrate that the proposed Voting Machine is a highly effective method for detecting anomalies in time-series volume data. The Voting Machine outperforms all individual detectors, achieving the best F-score for all types of anomalies across all three settings. Specifically, it achieves an F-score of 92.4%, 94.8%, and 96.0% for k=1, 2, and 3, respectively, which shows that it reduces false positives in the four types of anomaly detection. This performance improvement is likely due to the ability of the Voting Machine to combine the strengths of multiple individual detectors and to mitigate their weaknesses.

While the individual detectors perform well in some cases, none of them are as effective as the Voting Machine in detecting all types of anomalies under all three settings. The best-performing individual detector, Prophet, achieves an F-score of 91.0%, 89.1%, 89.3%, and 89.8% for k=1, 2, 3, and average respectively. This indicates that, while individual detectors can be effective in certain situations, they may not be suitable for detecting anomalies in complex, real-world datasets.

In terms of Recall, the Voting Machine achieves high values, with 86.6%, 92.4%, and 97.3% for k=1, 2, and 3, respectively. On the other hand, among the individual detectors, IF, KNN, and Prophet achieve the best Recall for k=1, 2, and 3, of 90.5%, 96.6%, and 99.5% respectively. This indicates that individual detectors may be more effective in identifying certain types of anomalies, while the Voting Machine excels in achieving overall performance improvements by leveraging the strengths of multiple detectors.

Table 4.1: Evaluation Results for Voting Machine Using Injected Anomalies. Precision (P), Recall (R), and F-score (F) are reported. The best performing method for each metric (column) is highlighted.

Settings	$k = 1$ (%)			$k = 2$ (%)			$k = 3$ (%)			**Average** (%)		
	P	R	F	P	R	F	P	R	F	P	R	F
LOF	98.1	54.4	70.0	96.8	65.1	77.8	85.2	97.8	91.1	91.5	72.5	80.9
IF	84.2	**90.5**	87.22	79.9	94.6	86.6	76.2	97.4	85.5	79.8	**94.2**	86.4
Gaussian	90.5	89.7	90.1	82.2	85.8	84.0	79.5	89.4	84.1	83.9	88.3	86.0
KNN	87.0	78.0	82.3	83.9	**96.6**	89.8	80.0	83.8	81.9	83.5	86.1	84.8
Prophet	95.8	86.7	91.0	84.6	94.0	89.1	81.1	**99.5**	89.3	86.4	93.4	89.8
Voting	**99.1**	86.6	**92.4**	**97.3**	92.4	**94.8**	**94.8**	97.3	**96.0**	**96.9**	92.1	**94.5**

Among the individual detectors, Prophet achieved the best performance for all types of anomalies and settings, with IF achieving the highest recall among the anomalies and LOF achieving the highest recall. These results indicate that different detectors may be more suitable for detecting different types of anomalies, and that selecting the appropriate detector(s) for a particular dataset can significantly impact the accuracy of anomaly detection.

Overall, the proposed Voting Machine is an effective method for detecting anomalies in time-series volume data, outperforming individual detectors and achieving high accuracy for various types of anomalies under different settings. The results demonstrate the effectiveness of combining multiple detection methods to improve anomaly detection accuracy in complex datasets, and highlight the need for further steps to identify system-level anomalies.

4.4.3 Evaluation on Graph Anomaly Detection

Data

The HIT system underwent several updates, including design changes to the patient record system. These updates have the potential to significantly affect the calculation of certain volume data, leading to changes in system-wide traffic patterns. These changes affect but are not limited to the following volume types: Surgery Volume, Dental Volume, Medicine Volume, and so on.

In order to evaluate the effectiveness of our framework in detecting system-level changes, a case study was conducted using the software system upgrades as a basis, assuming no prior knowledge of the changes. Specifically, there are 6 dates between May and June 2018 during which 36 facilities performed system maintenance related to the updates. Table 4.2 provides a list of the facilities and corresponding dates on which the updates were performed. 36 facilities conducted the changes on each date. The aim of the study was to evaluate whether our framework could effectively

Table 4.2: List of medical facilities and corresponding dates for system updates conducted during 2018.

Date Index	Facility Count	Facility ID
5/9/18	8	03, 29, 40, 42, 60, 62, 95, 96
5/18/18	6	16, 46, 48, 72, 73, 75
5/20/18	6	21, 37, 44, 56, 58, 65
5/30/18	5	54, 59, 62, 70, 93
6/10/18	6	02, 08, 31, 39, 50, 55
6/23/18	5	01, 04, 19, 44, 57

detect the events and flag them as anomalies by using graph algorithms to detect the resulting clusters from the system maintenance performed by the facilities.

Experimental Settings

The evaluation utilized all 10,054 preprocessed time-series volumes from 130 medical facilities between April 1, 2018, and June 31, 2018. The tests were conducted on 48 different settings, varying the MaxDeferTime from 0 days to 5 days, the Lookback Period from 1 week to 4 weeks, and the glom factor between 1 and 2. The same setting was used for the Voting Machine model as in the previous evaluation.

Detection Rate was used as our performance metric, which is equivalent to Recall (Eq. 4.5), since it only had access to information about the incidents, and hence only had true positive and false negative results. Once paracliques were detected, the intersection of each paraclique with the ground truth clusters listed in Table 4.2 was cross-checked. They were considered true positives only if the intersection contained three or more facilities.

Experimental Results

Figure 4.4 depicts the detection rate of our framework on 48 different settings, with the y-axis representing Detection Rate and the x-axis representing Max Defer Time. The results are shown using 8 different lines, each representing a different combination of lookback period and glom factor. The highest detection rate of 1 was achieved when the lookback period was 2 weeks and the glom factor was 2, indicating that all 36 facilities were successfully detected in this setting.

Lookback Period. Overall, our results showed that the best performance was generally achieved when the lookback period was set to 2 weeks. Interestingly, it was observed that the detection rate initially increased and then decreased after 2 with MaxDeferTime ranging from 0 to 5, when the lookback period was set to 2. For other values of lookback period, the best performance was generally achieved with a

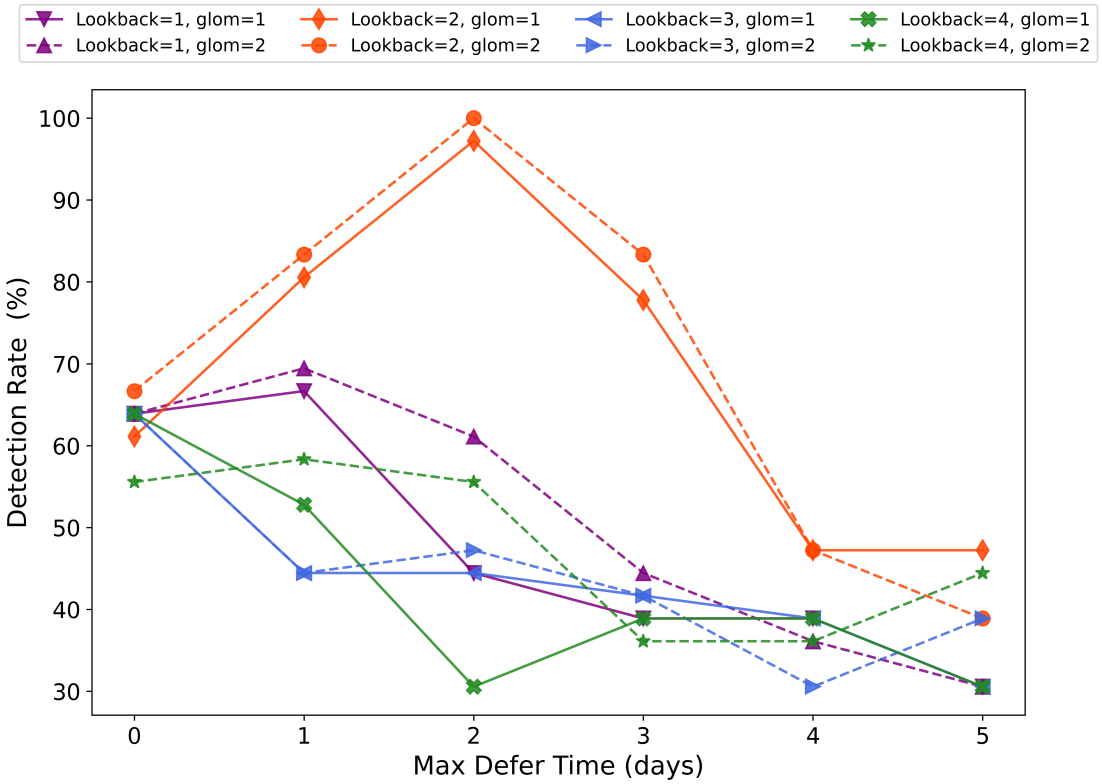


Figure 4.4: The Detection Rate of the proposed framework on different settings of MaxDeferTime, Lookback Period, and glom factor.

MaxDeferTime of 0 or 1, and the performance was similar across different values of MaxDeferTime. Therefore, the selection of the lookback period can have a significant impact on the performance of the method, and in this case, a 2-week lookback period was the most effective for separating the outlier from the mean value.

MaxDeferTime. In terms of the MaxDeferTime, the best performance varied depending on the specific settings. Despite that, in general, the results indicated that the best performance was achieved when MaxDeferTime was set to 0-2 days. This can be attributed to the fact that a large MaxDeferTime has a looser rule for including simultaneous outliers, which can introduce more noise into the graph and make it harder to distinguish true clusters from noise. As a result, it shows the volume changes for the updates generally deferred from 0 to 2 days.

Glom Factor. It is also observed that increasing the glom factor from 1 to 2 generally resulted in better detection rates. This could be due to the fact that with a higher glom number, more non-clique vertices are added into the cliques, which helps capture the underlying structure of the data better and reduce false negatives. In other words, a higher glom factor can improve the accuracy of the graph representation of the data and increase the likelihood of detecting true clusters. On the other hand, it is worth noting that a higher glom factor can also increase the number of false positives, as more non-clique vertices are included in the cliques. Therefore, selecting an appropriate glom factor is crucial in balancing the trade-off between detection rates and false positives.

Overall, the results of our evaluation indicate that our framework is effective in detecting system-level changes in time-series volume data. By selecting appropriate combinations of parameters, high detection rates can be achieved, in this case when using a lookback period of 2 weeks and a glom factor of 2. These results have important implications for the detection of anomalies in complex datasets, particularly in the healthcare domain.

Paracliques

Next, the detected paracliques were analyzed shown in Table 4.3, using a lookback period of 2 weeks, a MaxDeferTime of 2 days, and a glom factor of 1 and 2 to gain further insights. Interestingly, when using a glom factor of 2, five more facilities were added to the paracliques, marked in red in the table. In contrast, these facilities were not included when using a glom factor of 1. Specifically, station 544 in cluster 1 was missed by glom factor 1, and the other four facilities were not listed as having performed maintenance in Table 4.2.

To validate our results, the identified clusters were cross-checked with the ground truth in Table 4.2. Our analysis revealed that paraclique 2 corresponds to the cluster that performed system updates on 5/9/2018. Paracliques 1 and 7 combined contain the cluster that performed system updates on 5/18/2018 and 5/20/2018. These clusters were merged because the defer time of the volumes varied from 0 to 2 days in different facilities. Thus, these facilities are connected when building the graph for outliers with MaxDeferTime set to 2. Although facilities 16, 46, and 72 performed system maintenance on 5/18/2018, they were included in paraclique 7 because their volume changed with no strong connection to other facilities 2 days later.

It was also identified that the facilities that performed updates on 5/30/2018 are located in paraclique 5, while those that performed updates on 6/10/2018 are in paraclique 3. Additionally, the facilities that performed updates on 6/23/2018 are located in paraclique 4. Overall, our analysis demonstrated that our framework could successfully identify clusters resulting from system maintenance across 130 facilities, highlighting its potential for detecting system-level changes.

Volume Changes

To elaborate further, simultaneous outliers refer to the volumes that experience a significant change in their number of transactions across the facilities at the same time. In this case study, the top five volumes of simultaneous outliers accounted for

Table 4.3: Paracliques detected using lookback period of 2 weeks, MaxDeferTime of 2 days, and a glom factor of 1 and 2. Differences are marked in red.

Paraclique	glom = 1	glom = 2
1	21 37 48 56 58 65 73 75	21 37 48 56 58 65 73 75 **44**
2	03 29 40 42 60 62 95 96	03 29 40 42 60 62 95 96
3	02 08 31 39 50 55	02 08 31 39 50 55
4	01 04 19 44 57	01 04 19 44 57
5	54 59 62 70 93	54 59 62 70 93 **63 91**
6	20 52 59	20 52 59
7	16 46 72	16 46 72 **36 64**

91.9% of all simultaneous outliers detected. The volume with the highest percentage of simultaneous outliers was Medicine, which accounted for 43.2% of all simultaneous outliers. The second and third volumes with the highest percentage of simultaneous outliers were Dental and Surgery. These two volumes together accounted for 41.7% of all simultaneous outliers. The remaining two volumes in the top five were Primary Care and Nursery, accounting for 5.2% and 1.8% of all simultaneous outliers, respectively.

Therefore, examining the distribution of simultaneous outliers can gain insights into the specific volumes that are driving the paraclique results. In this case study, the top three volume types with the highest percentage of simultaneous outliers were all closely related to updates of the updates.

4.5 Summary

In this chapter, a spatial-temporal anomaly detection framework was proposed for system-level HIT systems. The framework consists of three main steps: data preprocessing, graph construction, and anomaly detection. In the data preprocessing step, the EHR data of HIT systems was collected and preprocessed. In the graph construction step, a weighted graph was constructed based on the correlation of simultaneous outliers between system volumes. In the anomaly detection step, graph algorithms were leveraged to identify clusters of stations that have high correlation of simultaneous outliers, which could indicate potential anomalies in the system.

Two evaluations were performed to test the effectiveness of our framework. The first evaluation focused on the anomaly detection performance of a voting machine model with four types of injected anomalies. The results showed that our framework was able to detect all injected anomalies with a low false positive rate.

The second evaluation focused on the detection of real-world anomalies in a large-scale HIT system. The results showed that our framework was able to successfully identify clusters of simultaneous outliers that corresponded to known system design

changes. It was also observed that the performance of the framework was sensitive to the choice of lookback period and MaxDeferTime, and that a higher glom factor generally resulted in better detection rates.

Overall, the proposed framework provides a promising approach for detecting spatial-temporal anomalies in system-level HIT systems. It has the potential to assist system administrators in detecting and addressing potential system issues in a timely manner, which can ultimately improve the overall system performance and patient care.

Chapter 5

Conclusions

Anomaly detection is a critical problem for the creation and maintenance of a reliable and modern healthcare system. This dissertation proposed three methods for detecting anomalies in two important aspects of HIT systems: discrete sequence and volume data. Although the designed and tested methods were initially developed for HIT systems, they have the potential to be applied to other large-scale computer systems as well.

Summary of Contributions

Chapter 2 focused on EHR sequence anomaly detection. A network representation method, HOTN, was proposed that efficiently captures high-dimensional event sequence data and represents it in a network. Then, a graph-based anomaly detection framework, AD-HOTN, was designed for explicitly identifying anomalies in data sequences, and a subsequence anomaly detection algorithm, CSAD-HOTN, was constructed for accurately locating anomalous segments. The methods outperform state-of-the-art baseline methods using synthetic sequence data and real-world EHRs, enabling more effective anomaly detection in high-dimensional event sequence data.

Chapter 3 developed EHR-BERT, a cutting-edge BERT-based model for detecting anomalies in EHR sequences. By leveraging the powerful BERT architecture

and incorporating the SMTP strategy, the model effectively captures contextual information and event correlations within EHRs, significantly improving anomaly detection accuracy. Furthermore, the SMTP strategy addresses false negatives, ensuring that no abnormal events are missed. The EHR-BERT model demonstrates the potential to enhance the quality and efficiency of healthcare services, as well as patient safety, across various healthcare settings. This work contributes an essential component to the broader research on improving clinical practice and patient care through advanced EHR anomaly detection techniques.

In chapter 4, the problem of system-level anomaly detection in HIT systems was addressed by proposing a comprehensive framework that combines machine learning models and graph algorithms to analyze high-dimensional volume anomalies. It demonstrated the effectiveness of our approach in detecting network-wide volume anomalies using real-world HIT datasets. The framework aimed to provide meaningful insights to support decision-making and enable fast response in the face of system-level hazards. The work contributed to the development of reliable and modern healthcare systems.

Bibliography

[1] A. A. Aburomman and M. B. I. Reaz. A novel svm-knn-pso ensemble method for intrusion detection system. *Applied Soft Computing*, 38:360–372, 2016. 79

[2] O. O. Aremu, D. Hyland-Wood, and P. R. McAree. A machine learning approach to circumventing the curse of dimensionality in discontinuous time series machine data. *Reliability Engineering & System Safety*, 195:106706, 2020. 78

[3] Associated Press. German hospital hacked, patient taken to another city dies. https://apnews.com/article/technology-hacking-europe-cf8f8eee1adcec69bcc864f2c4308c94, 2021. [Accessed: May 1, 2023]. 2

[4] R. Athilakshmi, S. G. Jacob, and R. Rajavel. Protein sequence based anomaly detection for neuro-degenerative disorders through deep learning techniques. In *Advances in Big Data and Cloud Computing*, pages 547–554. Springer, 2019. 9

[5] C. Aytekin, X. Ni, F. Cricri, and E. Aksu. Clustering and unsupervised anomaly detection with l 2 normalized deep auto-encoder representations. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2018. 10, 13

[6] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 56

[7] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 50

[8] E. Bigdeli, M. Mohammadi, B. Raahemi, and S. Matwin. Incremental anomaly detection using two-layer cluster-based structure. *Information Sciences*, 429:315–331, 2018. 75, 78

[9] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021. 87

[10] M. Boldt, A. Borg, S. Ickin, and J. Gustafsson. Anomaly detection of event sequences using multiple temporal resolutions and markov chains. *Knowledge and Information Systems*, 62(2):669–686, 2020. 14, 49

[11] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. *Handbook of Combinatorial Optimization: Supplement Volume A*, pages 1–74, 1999. 94

[12] A. Boukerche, L. Zheng, and O. Alfandi. Outlier detection: Methods, models, and classification. *ACM Computing Surveys (CSUR)*, 53(3):1–37, 2020. 75, 79

[13] S. K. Brenner, R. Kaushal, Z. Grinspan, C. Joyce, I. Kim, R. J. Allard, D. Delgado, and E. L. Abramson. Effects of health information technology on patient outcomes: a systematic review. *Journal of the American Medical Informatics Association*, 23(5):1016–1036, 2016. 40

[14] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000. 86

[15] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In

Proceedings of the First Workshop on Machine Learning for Computing Systems, pages 1–8, 2018. 10

[16] S. Budalakoti, A. N. Srivastava, and M. E. Otey. Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):101–113, 2008. 14, 27

[17] C. Callegari, M. Pagano, S. Giordano, and T. Pepe. Combining wavelet analysis and information theory for network anomaly detection. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, pages 1–5, 2011. 75, 77

[18] J. Camacho, A. Pérez-Villegas, P. García-Teodoro, and G. Maciá-Fernández. Pca-based multivariate statistical network monitoring for anomaly detection. *Computers & Security*, 59:118–137, 2016. 75, 77, 78

[19] M. Çelik, F. Dadaşer-Çelik, and A. Ş. Dokuz. Anomaly detection in temperature data using dbscan algorithm. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 91–95. IEEE, 2011. 14, 27

[20] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2010. 6, 9, 49

[21] V. Chandola, V. Mithal, and V. Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *2008 Eighth IEEE International Conference on Data Mining*, pages 743–748. IEEE, 2008. 10, 13, 14

[22] E. J. Chesler and M. A. Langston. Combinatorial genetic regulatory network analysis tools for high throughput transcriptomic data. In *Systems Biology and Regulatory Genomics: Joint Annual RECOMB 2005 Satellite Workshops on*

Systems Biology and on Regulatory Genomics, San Diego, CA, USA; December 2-4, 2005, Revised Selected Papers, pages 150–165. Springer, 2006. 94

[23] S.-B. Cho and H.-J. Park. Efficient anomaly detection by modeling privilege flows using hidden markov model. *Computers & Security*, 22(1):45–55, 2003. 10

[24] F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997. 94

[25] K. W. Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017. 10, 14

[26] M. Cinque, D. Cotroneo, and A. Pecchia. Event logs for the analysis of software failures: A rule-based approach. *IEEE Transactions on Software Engineering*, 39(6):806–821, 2012. 45, 48

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 53, 58

[28] C. H. Ding, X. He, and H. Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 275–280, 2001. 94

[29] J. Dromard, G. Roudiere, and P. Owezarski. Online and scalable unsupervised network anomaly detection method. *IEEE Transactions on Network and Service Management*, 14(1):34–47, 2016. 75, 78

[30] M. Du, F. Li, G. Zheng, and V. Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298, 2017. 10, 45, 46, 50, 66

[31] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 88

[32] H. Estiri, J. G. Klann, and S. N. Murphy. A clustering approach for detecting implausible observation values in electronic health records data. *BMC Medical Informatics and Decision Making*, 19(1):1–16, 2019. 9, 10

[33] T. Fernando, H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Deep learning for medical anomaly detection–a survey. *ACM Computing Surveys (CSUR)*, 54(7):1–37, 2021. 84

[34] K. Flanagan, E. Fallon, P. Connolly, and A. Awad. Network anomaly detection in time series using distance based outlier detection with cluster density analysis. In *2017 Internet Technologies and Applications (ITA)*, pages 116–121. IEEE, 2017. 75, 78

[35] G. Folino and P. Sabatino. Ensemble based collaborative and distributed intrusion detection systems: A survey. *Journal of Network and Computer Applications*, 66:1–16, 2016. 79

[36] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 10, 14

[37] H. Guo, S. Yuan, and X. Wu. LogBERT: Log anomaly detection via BERT. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021. 45, 46, 50, 51, 53, 58, 66, 71

[38] D. Gusfield. Algorithms on stings, trees, and sequences: Computer science and computational biology. *ACM Sigact News*, 28(4):41–60, 1997. 6

[39] T. Ha, D. Swanson, M. Larouche, R. Glenn, D. Weeden, P. Zhang, K. Hamre, M. Langston, C. Phillips, M. Song, et al. Cbgrits: cerebellar gene regulation in time and space. *Developmental biology*, 397(1):18–30, 2015. 94

[40] R. D. Hagan, M. A. Langston, and K. Wang. Lower bounds on paraclique density. *Discrete Applied Mathematics*, 204:208–212, 2016. 91, 94

[41] M. Hamdi and N. Boudriga. Detecting denial-of-service attacks using the wavelet transform. *Computer Communications*, 30(16):3203–3213, 2007. 75, 77

[42] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 56

[43] T. Heart, O. Ben-Assuli, and I. Shabtai. A review of phr, emr and ehr integration: A more personalized healthcare and public health policy. *Health Policy and Technology*, 6(1):20–25, 2017. 40

[44] N. Hubballi, S. Biswas, and S. Nandi. Sequencegram: n-gram modeling of system calls for program based anomaly detection. In *2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011)*, pages 1–10. IEEE, 2011. 28

[45] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 387–395, 2018. 10

[46] M. Injadat, F. Salo, A. B. Nassif, A. Essex, and A. Shami. Bayesian optimization with machine learning algorithms towards anomaly detection. In *2018 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE, 2018. 79

[47] Jercich, Kat. The biggest healthcare data breaches of 2021. https://www.healthcareitnews.com/news/biggest-healthcare-data-breaches-2021, 2021. [Accessed: May 1, 2023]. 1

[48] R. Jin, H. Hong, H. Wang, N. Ruan, and Y. Xiang. Computing label-constraint reachability in graph databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 123–134, 2010. 15

[49] S. S. Joshi and V. V. Phoha. Investigating hidden markov models capabilities in anomaly detection. In *Proceedings of the 43rd Annual Southeast Regional Conference, Volume 1*, pages 98–103, 2005. 10

[50] W. Khreich, B. Khosravifar, A. Hamou-Lhadj, and C. Talhi. An anomaly detection system based on variable n-gram features and one-class svm. *Information and Software Technology*, 91:186–197, 2017. 79

[51] T. Kieu, B. Yang, and C. S. Jensen. Outlier detection for multidimensional time series using deep neural networks. In *2018 19th IEEE international conference on mobile data management (MDM)*, pages 125–134. IEEE, 2018. 75, 79

[52] M. O. Kim, E. Coiera, and F. Magrabi. Problems with health information technology and their effects on care delivery and patient outcomes: a systematic review. *Journal of the American Medical Informatics Association*, 24(2):246–250, 2017. 40

[53] T.-Y. Kim and S.-B. Cho. Web traffic anomaly detection using c-lstm neural networks. *Expert Systems with Applications*, 106:66–76, 2018. 75, 79

[54] M. Klassen and N. Yang. Anomaly based intrusion detection in wireless networks using bayesian classifier. In *2012 IEEE fifth international conference on advanced computational intelligence (ICACI)*, pages 257–264. IEEE, 2012. 79

[55] J. Knights, Z. Heidary, J. M. Cochran, et al. Detection of behavioral anomalies in medication adherence patterns among patients with serious mental illness engaged with a digital medicine system. *JMIR Mental Health*, 7(9):e21378, 2020. 43

[56] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 61–72, 2004. 77, 78

[57] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo. Failure prediction in ibm bluegene/l event logs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 583–588. IEEE, 2007. 49

[58] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen. Log clustering based problem identification for online service systems. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 102–111, 2016. 45, 49, 66

[59] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008. 87

[60] Y. Liu, L. Zhang, and Y. Guan. A distributed data streaming algorithm for network-wide traffic anomaly detection. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):81–82, 2009. 77

[61] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li. Mining invariants from console logs for system problem detection. In *USENIX Annual Technical Conference*, pages 1–14, 2010. 45, 49

[62] W. Lu and A. A. Ghorbani. Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal processing*, 2009:1–16, 2008. 75, 77

[63] C. Ma and C. Kingsford. Detecting, categorizing, and correcting coverage anomalies of rna-seq quantification. *Cell systems*, 9(6):589–599, 2019. 43

[64] N. Malini and M. Pushpa. Analysis on credit card fraud identification techniques based on knn and outlier detection. In *2017 Third International Conference*

on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), pages 255–258. IEEE, 2017. 9

[65] C. Manning and H. Schutze. *Foundations of statistical natural language processing*. MIT press, 1999. 28

[66] C. Marceau. Characterizing the behavior of a program using multiple-length n-grams. In *Proceedings of the 2000 Workshop on New Security Paradigms*, pages 101–110, 2001. 10, 14, 17, 45, 49

[67] R. Marcilly, J. Schiro, M. C. Beuscart-Zéphir, and F. Magrabi. Building usability knowledge for health information technology: a usability-oriented analysis of incident reports. *Applied clinical informatics*, 10(03):395–408, 2019. 42, 71

[68] G. Martin, S. Ghafur, I. Cingolani, J. Symons, D. King, S. Arora, and A. Darzi. The effects and preventability of 2627 patient safety incidents related to health information technology failures: a retrospective analysis of 10 years of incident reporting in england and wales. *The Lancet Digital Health*, 1(3):e127–e135, 2019. 40, 42, 71

[69] T. Mehmood and H. B. M. Rais. Svm for network anomaly detection using aco feature subset. In *2015 International symposium on mathematical sciences and computing research (iSMSC)*, pages 121–126. IEEE, 2015. 79

[70] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, et al. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, volume 19, pages 4739–4745, 2019. 45, 46, 50, 66

[71] C. C. Michael and A. Ghosh. Two state-based approaches to program-based anomaly detection. In *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*, pages 21–30. IEEE, 2000. 10, 14, 17

[72] C. C. Michael and A. Ghosh. Simple, state-based approaches to program-based anomaly detection. *ACM Transactions on Information and System Security (TISSEC)*, 5(3):203–237, 2002. 49

[73] G. Münz, S. Li, and G. Carle. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*, pages 13–14, 2007. 10, 13

[74] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han. Enhanced network anomaly detection based on deep neural networks. *IEEE access*, 6:48231–48246, 2018. 75, 79

[75] H. Niu, O. Omitaomu, Q. Cao, O. Ozmen, H. Klasky, M. M. Olama, L. Pullum, T. Kuruganti, M. Ward, A. Laurio, et al. Anomaly detection in sequential health care data using higher-order network representation. Technical report, Oak Ridge National Laboratory, Oak Ridge, TN, United States, 2020. 9, 14, 27, 29

[76] H. Niu, O. A. Omitaomu, and Q. C. Cao. Machine committee framework for power grid disturbances analysis using synchrophasors data. *Smart Cities*, 4(1):1–16, 2020. 79

[77] H. Niu, O. A. Omitaomu, Q. C. Cao, M. Olama, O. Ozmen, H. Klasky, L. Pullum, A. T. Malviya, T. Kuruganti, J. Scott, et al. Adaptive anomaly detection for dynamic clinical event sequences. In *2020 IEEE International Conference on Big Data*, pages 4919–4928. IEEE, 2020. 10, 14, 23, 27, 29, 49

[78] H. Niu, O. A. Omitaomu, M. A. Langston, M. Olama, O. Ozmen, H. B. Klasky, A. Laurio, B. Sauer, M. Ward, and J. Nebeker. Detecting anomalous sequences in electronic health records using higher-order tensor networks. *Journal of Biomedical Informatics*, 135:104219, 2022. 3, 9, 45, 46, 49, 71

[79] O. A. Omitaomu, H. B. Klasky, M. Olama, O. Ozmen, L. Pullum, A. M. Thakur, T. Kuruganti, J. M. Scott, A. Laurio, F. Drews, et al. A new

methodological framework for hazard detection models in health information technology systems. *Journal of Biomedical Informatics*, 124:103937, 2021. 1, 3, 96

[80] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais. Detection of early-stage enterprise infection by mining large-scale log data. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 45–56. IEEE, 2015. 45, 48

[81] O. Ozmen, H. B. Klasky, O. A. Omitaomu, M. Olama, T. Kuruganti, M. Ward, J. M. Scott, A. Laurio, F. Drews, and J. R. Nebeker. Feature engineering and process mining to enable hazard detection in health information technology. *AMIA Summits on Translational Science Proceedings*, 2020:469, 2020. 3

[82] C. Pascoal, M. R. De Oliveira, R. Valadas, P. Filzmoser, P. Salvador, and A. Pacheco. Robust feature selection and robust pca for internet traffic anomaly detection. In *2012 Proceedings Ieee Infocom*, pages 1755–1763. IEEE, 2012. 75, 77, 78

[83] A. D. Perkins and M. A. Langston. Threshold selection in gene co-expression networks using spectral graph theory techniques. In *BMC bioinformatics*, volume 10, pages 1–11. BioMed Central, 2009. 94

[84] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson. Network traffic anomaly detection using recurrent neural networks. *arXiv preprint arXiv:1803.10769*, 2018. 9

[85] B. J. Radford, B. D. Richardson, and S. E. Davis. Sequence aggregation rules for anomaly detection in computer network traffic. *arXiv preprint arXiv:1805.03735*, 2018. 9

[86] S. Rajasegarar, C. Leckie, and M. Palaniswami. Hyperspherical cluster based distributed anomaly detection in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 74(1):1833–1847, 2014. 78

[87] 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*, pages 427–438, 2000. 10, 13, 27

[88] C. Ranjan, S. Ebrahimi, and K. Paynabar. Sequence graph transform (sgt): A feature embedding function for sequence data mining. *arXiv preprint arXiv:1608.03533*, 2016. 10, 14, 27

[89] L. Rasmy, Y. Xiang, Z. Xie, C. Tao, and D. Zhi. Med-BERT: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *NPJ digital medicine*, 4(1):86, 2021. 46

[90] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of pca for traffic anomaly detection. In *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 109–120, 2007. 77

[91] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2):117–149, 1996. 17

[92] J. P. Rouillard. Real-time log file analysis using the simple event correlator (sec). In *LISA*, volume 4, pages 133–150, 2004. 48

[93] S. Roy, A. C. König, I. Dvorkin, and M. Kumar. Perfaugur: Robust diagnostics for performance anomalies in cloud services. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1167–1178. IEEE, 2015. 48

[94] V. Saligrama, J. Konrad, and P.-M. Jodoin. Video anomaly identification. *IEEE Signal Processing Magazine*, 27(5):18–33, 2010. 9

[95] P. Shoubridge, M. Kraetzl, W. Wallis, and H. Bunke. Detection of abnormal change in a time series of graphs. *Journal of Interconnection Networks*, 3(01n02):85–101, 2002. 20

[96] P. Sornsuwit and S. Jaiyen. Intrusion detection model based on ensemble learning for u2r and r2l attacks. In *2015 7th international conference on information technology and electrical engineering (ICITEE)*, pages 354–359. IEEE, 2015. 79

[97] A. Soule, K. Salamatian, and N. Taft. Combining filtering and statistical methods for anomaly detection. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 31–31, 2005. 77

[98] P. Sun, S. Chawla, and B. Arunasalam. Mining for outliers in sequential databases. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 94–105. SIAM, 2006. 10, 14, 28, 45, 46, 49

[99] M. Swarnkar and N. Hubballi. Ocpad: One class naive bayes classifier for payload based anomaly detection. *Expert Systems with Applications*, 64:330–339, 2016. 79

[100] X. Tan and H. Xi. Hidden semi-markov model for anomaly detection. *Applied Mathematics and Computation*, 205(2):562–567, 2008. 10

[101] S. J. Taylor and B. Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018. 88

[102] The U.S. Department of Health and Human Services (HHS), Office of the National Coordinator (ONC). 2022 report to congress: Update on the access, exchange, and use of electronic health information. https://www.healthit.gov/sites/default/files/page/2023-02/2022_ONC_Report_to_Congress.pdf, 2022. [Accessed: May 1, 2023]. 1

[103] S. Thudumu, P. Branch, J. Jin, and J. Singh. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7:1–30, 2020. 75, 78, 79

[104] J. Troisi, A. Landolfi, L. Sarno, S. Richards, S. Symes, D. Adair, C. Ciccone, G. Scala, P. Martinelli, and M. Guida. A metabolomics-based approach for non-invasive screening of fetal central nervous system anomalies. *Metabolomics*, 14(6):1–10, 2018. 43

[105] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 50, 56

[106] H. Wang, J. Gu, and S. Wang. An effective intrusion detection framework based on svm with feature augmentation. *Knowledge-Based Systems*, 136:130–139, 2017. 79

[107] Z. Wang, Z. Chen, J. Ni, H. Liu, H. Chen, and J. Tang. Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection. *arXiv preprint arXiv:2008.13361*, 2020. 10, 14

[108] Y. Wu, H.-N. Dai, and H. Tang. Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal*, 9(12):9214–9231, 2021. 75, 79

[109] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system:

Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 53

[110] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang. Graph based tensor recovery for accurate internet anomaly detection. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1502–1510. IEEE, 2018. 15

[111] J. Xu, M. Saebi, B. Ribeiro, L. M. Kaplan, and N. V. Chawla. Detecting anomalies in sequential data with higher-order networks. *arXiv preprint arXiv:1712.09658*, 2017. 10, 14, 17, 18

[112] J. Xu, T. L. Wickramarathne, and N. V. Chawla. Representing higher-order dependencies in networks. *Science Advances*, 2(5):e1600028, 2016. 15, 16, 17, 27, 29

[113] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132, 2009. 45, 49, 64

[114] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th annual computer security applications conference*, pages 199–208, 2013. 45, 48

[115] E. N. Yolaçan. *Learning from sequential data for anomaly detection*. Northeastern University, 2014. 2

[116] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016. 25

[117] B. Zhang, J. Yang, J. Wu, D. Qin, and L. Gao. Pca-subspace method—is it good enough for network-wide anomaly detection. In *2012 IEEE Network Operations and Management Symposium*, pages 359–367. IEEE, 2012. 75, 77

[118] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, et al. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817, 2019. 45, 46, 50, 71

[119] Z. Zhang, Q. K. Telesford, C. Giusti, K. O. Lim, and D. S. Bassett. Choosing wavelet methods, filters, and lengths for functional brain network construction. *PloS one*, 11(6):e0157243, 2016. 77

[120] W. Zhong, G. Altun, R. Harrison, P. C. Tai, and Y. Pan. Improved k-means clustering algorithm for exploring local protein sequence motifs representing common structural property. *IEEE Transactions on Nanobioscience*, 4(3):255–265, 2005. 14, 27, 28

Vita

Haoran Niu was born in Changzhi, China. He earned a Master of Science degree in Electrical Engineering from Xi'an Jiaotong University in 2017. Following this, he was admitted to the Computer Engineering PhD program at the University of Tennessee, Knoxville, under the guidance of Dr. Michael A. Langston, and has been collaborating with Dr. Olufemi A. Omitaomu from Oak Ridge National Laboratory. His research interests include system security, anomaly detection, network analysis, graph algorithms, machine learning, and AI.