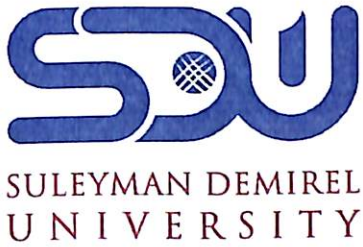
Ministry of Science and Higher Education of the Republic of Kazakhstan

Suleyman Demirel University



Arailym Talasbek

The Automation capabilities in the field of Software Testing

THESIS

Presented in Partial Fulfilment for the

Master of Technical Sciences Degree in Computer Science

(degree code: 7M06102)

Department of Computer Science

Faculty of Engineering and Natural Sciences

Supervisor: **PhD Birzhan Moldagaliyev**

Kaskelen 2023

Suleyman Demirel University
Faculty of Engineering and Natural Sciences
Department of Computer Science

Dean of Faculty

Associate Professor

PhD Zhamanov A.

06 06 2023

Topic of the thesis:

The Automation capabilities in the field of Software Testing

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science" SDU, 2021-2023

Head of Department Assistant Professor, PhD Mukash Zh.

Academic Supervisor PhD Birzhan Moldagaliyev

Master student Arailym Talasbek

Kaskelen 2023

Ministry of Science and Higher Education of the Republic of Kazakhstan

Suleyman Demirel University

Arailym Talasbek

The Automation capabilities in the software testing field

THESIS

Presented in Partial Fulfilment for the

Master of Technical Sciences Degree in Computer Science

(degree code: 7M06102)

Department of Computer Science

Faculty of Engineering and Natural Sciences

Supervisor: **PhD Birzhan Moldagaliyev**

Kaskelen 2023

Suleyman Demirel University
Faculty of Engineering and Natural Sciences
Department of Computer Science

Dean of Faculty

Associate Professor

PhD Zhamanov A.

_____

« _____ »_____ 2023

Topic of the thesis:

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science" SDU, 2021-2023

Head of Department _____ Assistant Professor, PhD Mukash Zh.

Academic Supervisor _____ PhD Birzhan Moldagaliyev

Master student _____ Arailym Talasbek

Kaskelen 2023

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Talasbek Arailym

2023

Acknowledgements

My sincere appreciation goes out to Birzhan Moldagaliyev, my thesis advisor, for his constant direction, inspiration, and support during the entire process. I have benefited greatly from his supervision in terms of product information, problem insights, and regular meetings. I want to express my gratitude for his constant direction and counsel, which were crucial to the completion of my thesis. His vast knowledge and eagerness to help have been instrumental in overcoming several obstacles and assuring the constant advancement of my research pursuits.

Dedication

This thesis is dedicated to:

My parents, sisters, Birzhan Moldagaliyev, master friends and many other for their support, help, sense of humour and useful comments for improving this project.

Abstract

Automated log analysis and anomaly detection play a crucial role in raising software quality.As modern systems grow in size and complexity, the number of journals grows exponentially makes manual testing impossible. Automated log analysis and anomaly detection approaches can uncover anomalies that might go undetected by manual examination by analyzing and monitoring log data produced by software programs. This preventative strategy enables prompt issue identification and resolving, improving software quality and user satisfaction. In order to find and fix problems that can have an impact on the functionality, dependability, and user experience of software systems, automated log analysis and anomaly detection play a crucial role in raising software quality. Also the system helpful for real customers experiencing problems in real time, the system provides a quick resolution of current errors and warnings on system server, by logging each client request, we detect anomalies in the product software. The system also has a real-time notification function to attract testers, developers to fix bugs in a timely manner, which leads to better software quality and better user experience.

Аңдатпа

Автоматты журналды талдау және аномалияны анықтау бағдарламалық құрал сапасын жақсартуда маңызды рөл атқарады. Заманауи жүйелердің көлемі мен күрделілігі өскен сайын, журналдар саны экспоненциалды түрде өседі, бұл қолмен тестілеуді мүмкін емес етеді. Автоматтандырылған журналды талдау және аномалияларды анықтау тәсілдері бағдарламалар арқылы жасалған журнал деректерін талдау және бақылау арқылы қолмен тексеру арқылы анықталмауы мүмкін ауытқуларды анықтай алады. Бұл проактивті стратегия ақауларды жылдам анықтауға және түзетуге, бағдарламалық құралдың сапасын және пайдаланушының қанағаттануын жақсартуға мүмкіндік береді. Автоматты журналды талдау және аномалияларды анықтау бағдарламалық жасақтама жүйесінің функционалдығына, сенімділігіне және қолайлылығына әсер етуі мүмкін мәселелерді табу және түзету үшін бағдарламалық құрал сапасын жақсартуда маңызды рөл атқарады. Сондай-ақ, жүйе нақты уақыт режимінде проблемаларды бастан кешіретін нақты тұтынушылар үшін пайдалы, жүйе ағымдағы қателер мен ескертулердің жүйелік серверде тез шешілуін қамтамасыз етеді, әрбір клиент сұранысын тіркейді, біз өнімнің бағдарламалық жасақтамасындағы ауытқуларды анықтаймыз. Сондай-ақ жүйеде тестерлерді, әзірлеушілерді қателерді дер кезінде түзетуге тарту үшін нақты уақыттағы хабарландыру функциясы бар, бұл бағдарламалық жасақтама сапасының жақсаруына және пайдаланушы тәжірибесінің жақсаруына әкеледі.

Аннотация

Автоматический анализ журналов и обнаружение аномалий играют решающую роль в повышении качества программного обеспечения. По мере роста размеров и сложности современных систем количество журналов растет в геометрической прогрессии, что делает ручное тестирование невозможным. Автоматизированный анализ журналов и подходы к обнаружению аномалий могут обнаруживать аномалии, которые могут остаться незамеченными при ручном изучении, путем анализа и мониторинга данных журналов, создаваемых программами. Эта превентивная стратегия позволяет быстро выявлять и устранять проблемы, повышая качество программного обеспечения и удовлетворенность пользователей. Автоматический анализ журналов и обнаружение аномалий играют решающую роль в повышении качества программного обеспечения, чтобы находить и устранять проблемы, которые могут повлиять на функциональность, надежность и удобство использования программных систем. Также система полезна для реальных клиентов, испытывающих проблемы в режиме реального времени, система обеспечивает быстрое устранение текущих ошибок и предупреждений на сервере системы, регистрируя каждый запрос клиента, мы обнаруживаем аномалии в программном обеспечении продукта. В системе также есть функция уведомления в режиме реального времени для привлечения тестировщиков, разработчиков к своевременному исправлению ошибок, что приводит к повышению качества программного обеспечения и улучшению пользовательского опыта.

Abbreviations

ML - Machine Learning

NLP- Natural Language Proccesing

HDFS - Hadoop File System

SLCT - Sequence Log Compression and Transformation

IPLoM - Iterative Partitioning Log Mining

LKE - Log Key Extraction

SVM - Support Vector Machine

SMTP - Simple Mail Transfer Protocol

Table of Contents

Chapter 1

Background and motivations

1.1 Introduction

Automated log analysis and anomaly detection are essential for improving software quality because they help identify and fix issues that may affect the functionality, dependability, and user experience of software systems. By analyzing and monitoring log data generated by software systems, automated log analysis and anomaly detection approaches can identify possible problems or anomalies that might go unnoticed by manual review. This proactive approach provides quick issue detection and resolution, enhancing the caliber of software and user pleasure. The following are the research goals for this dissertation:
- To investigate the drawbacks of traditional manual log analysis techniques as well as the advantages of automation.
- To create and put into use a revolutionary automated log analysis framework that uses anomaly detection methods.
- To assess the potency and efficiency of the suggested framework in raising software quality through the identification and correction of anomalies.
- To evaluate the effects of automated log analysis and anomaly detection on the debugging, troubleshooting, and maintenance procedures involved in software development.

Automated log analysis and anomaly detection are essential for improving software quality because they help identify and fix issues that may affect the functionality, dependability, and user experience of software systems. By analyzing and monitoring log data generated by software systems, automated log analysis and anomaly detection approaches can identify possible problems or anomalies that might go unnoticed by manual review. This proactive approach provides quick issue detection and resolution, enhancing the caliber of software and user pleasure. The following are the research goals for this dissertation: To investigate the drawbacks of traditional manual log analysis techniques as well as the advantages of automation.To create and put into use a revolutionary automated log analysis framework that uses anomaly detection methods.To assess the potency and efficiency of the suggested framework in raising software quality through the identification and correction of anomalies.To evaluate the effects of automated log analysis and anomaly detection on the debugging, troubleshooting, and maintenance procedures involved in software development.

Traditional manual log analysis methods[1] have a number of shortcomings that make it difficult for them to effectively analyze log data and ensure the quality of software. Because testers or developers may overlook significant log entries or mistakenly identify trends and anomalies, human error may happen during manual log analysis. This could lead to unknown issues and potential software failures. The manual analysis of large log data volumes takes time. It becomes increasingly impractical for testers and developers to manually go through logs as systems get more complex and log files grow. They work less efficiently, and it takes longer to identify and fix software issues. The number of generated log entries increases exponentially as software system complexity increases. Scaling up manual log analysis methods to handle this growing amount of data is challenging. It becomes difficult to manually check and analyze logs in a fair amount of time, making it difficult to accurately discover abnormalities and patterns. Manual log analysis techniques occasionally make use of keyword searches or pre-established procedures to find specific occurrences or issues. This approach might miss subtle patterns or abnormalities that are outside the bounds of the predetermined constraints, perhaps omitting issues with software quality.

The existing system will be beneficial because On the other side, automating log analysis has the following advantages: Automated log analysis algorithms are capable of processing massive amounts of log data rapidly and effectively. Algorithms and systems that can parse, filter, and analyze logs at scale make it possible to find abnormalities and trends more quickly. This effectiveness enables for more thorough coverage while cutting down on the time and effort needed for log analysis. Automation makes it possible to analyze log data in real-time or very close to real-time, which enables proactive issue detection. The early diagnosis of potential software issues is made possible by anomaly detection techniques, which can spot departures from expected behavior. This proactive strategy lessens the impact of problems on software quality by preventing them from getting worse. Automated log analysis uses cutting-edge data analytics methods like machine learning to increase the precision of anomaly detection. These algorithms can spot tiny irregularities and complicated patterns that would be hard for people to notice. Automation increases software quality and log analysis accuracy by lowering false positives and false negatives. Automated log analysis solutions are able to manage growing log volumes, ensuring thorough analysis without being constrained by the availability of human resources.Automated log analysis makes it possible to monitor log data continuously, allowing abnormalities and problems to be found as they arise. Continual software quality monitoring and rapid problem-solving are made possible by this proactive monitoring, which enhances program dependability and user happiness.

Automated notifications are important for alerting system administrators or developers to abnormalities since they guarantee timely awareness and rapid response. System or application anomalies may have serious repercussions, including service interruptions, security lapses, or performance deterioration. System administrators and developers can get instant alerts about these abnormalities thanks to automated notifications, which enables them to take quick action and reduce risks. System administrators or developers can quickly find and fix anomalies before they become serious issues by receiving automatic notifications [2]. With rapid problem-solving, issues are reduced in impact on system performance, user experience, and overall operations. Particularly in large-scale or compli-

cated systems, manual monitoring and anomaly detection in logs can be resource- and time-intensive. By automatically identifying irregularities and alerting the appropriate employees, automated notifications streamline the process. As a result, administrators and developers may concentrate on fixing problems rather than spending valuable time and resources manually looking for them.Developers or system administrators can get thorough notifications with pertinent context, including problem codes, log excerpts, timestamps, and other important data. They can then respond appropriately to investigate and resolve the issue after rapidly understanding its nature. Automated notifications give system administrators or developers immediate information about anomalies that can be used to take appropriate action, reduce risks, and guarantee the proper operation of systems and applications. They improve resource utilization, cooperation, incident management, and compliance, which eventually results in increased system dependability, security, and user happiness.

1.2 Literature Review

By quickly locating and fixing problems in software systems, automated log analysis and anomaly detection are essential for improving software quality. An overview of current studies on automated log analysis, anomaly detection, and their effects on software quality improvement is given in this review of the literature. It examines the many strategies, techniques, and algorithms used in this area and points out areas that still need to be studied as well as problems and opportunities.

Automated log analysis is the process of mining software system log data for useful insights and information that can be put to use. The significance of automated log analysis in raising software quality has been underlined in a number of studies. To handle the ever-increasing volume and complexity of log data, for instance, A. Wadekar and A. Patil (2019) [3] highlighted the necessity for automated log analysis tools. They suggested log mining methods for identifying and treating software errors.

Finding patterns or events in log data that dramatically depart from expected

behavior is the goal of anomaly detection. In the literature, a number of anomaly detection methods have been put forth, including statistical techniques, machine learning techniques, and data mining techniques.Using machine learning approaches, Lakshmi Geethanjali Mandagondi (2007) [4] presented a framework for anomaly identification in system logs. They gave examples of how their method worked to find aberrant log entries and enhance the quality of software.

Automated log analysis and anomaly detection integration significantly raises the caliber of software. These methods enable early detection and resolution of software faults by automatically examining log data, spotting anomalies, and delivering insights. Domenico Cotroneo and Roberto Natella(2014) [5] created an automated log analysis system that successfully identified and treated software flaws, enhancing software dependability and customer satisfaction. Automated log analysis and anomaly detection have made use of a variety of methodologies and algorithms. These consist of deep learning methods, support vector machines (SVM), hidden Markov models (HMM), clustering algorithms, and rule-based strategies. The method chosen is determined by variables like the features of the log data, the complexity of the system, and the level of accuracy that is required. For efficient log analysis and anomaly identification, Z. Zhao (2018) [6] presented a hybrid approach integrating rule-based and machine learning techniques.Although automated log analysis and anomaly detection have shown tremendous promise in raising software quality, there are still a number of difficulties and areas that might use more study. The scalability of methods to handle massive amounts of log data produced by contemporary software systems is a significant obstacle. A important research focus continues to be assuring the precision and dependability of anomaly detection algorithms. Additionally, there are additional prospects for research due to the interpretability of automated log analysis data and the incorporation of domain expertise. Configuring Automatic Email Notifications of Events and Reports [7] explains states that setting up automatic email notifications for events and reports is a crucial part of efficient system communication and monitoring. For such notification systems, the available literature offers insights on the design, implementation, triggering methods, content, integration, and security issues. Organizations can create strong and effective email notifi-

cation systems to notify stakeholders and enable prompt actions by utilizing the knowledge and best practices from these studies.

The literature study emphasizes the value of automated anomaly detection and log analysis in raising software quality. It gives a summary of the present research, investigates various methods and algorithms, and points out obstacles and chances for future study. Organizations can strengthen their software development processes, increase software stability, and guarantee greater customer pleasure by utilizing these automated solutions. Future studies in this field should concentrate on overcoming scaling difficulties, enhancing the precision of anomaly detection systems, and successfully incorporating domain knowledge.

1.3 List of Publications

One paper that was published in the publication "SDU Bulletin" contained the main concepts and conclusions of the dissertation. In order to thoroughly analyze all of the methodologies under consideration, the research undertaken for this dissertation integrates some of the findings and experiments described in those works. Here is a list of the publications produced for this master's thesis. The following articles, listed in order of appearance in the text, serve as the foundation for the thesis:

1. Suleyman Demirel University Bulletin: Natural and Technical Sciences, v. 62, April 2023. Talasbek A.; Birzhan Moldagaliyev "The Automation capabilities in the field of software testing" [8]

Chapter 2

Methodology

The purpose of this study is to determine how automated log analysis and anomaly detection improve the quality of software. A combination of quantitative and qualitative research techniques will be used to accomplish this. The automated log analysis and anomaly detection framework will be the focus of data collection, analysis, and assessment for the research methodology. The research technique is thoroughly explained in the following parts.

Finding pertinent log sources connected to the software system under study is the first step in the research approach. Server logs, application logs, error logs, performance logs, and other pertinent sources are examples of possible log sources. These logs are crucial for comprehending system activity, spotting anomalies, and enhancing the caliber of software. Choosing the data gathering method is a step in the data collection process. Log file monitoring, log streaming, or log aggregation are examples of data collection systems. To collect the log data, a suitable technique will be chosen based on the unique needs of the study. This procedure might entail putting in place the required mechanisms or using the log collection technologies that already exist.

After the log data has been gathered, a framework for automated log analysis and anomaly detection will be created and put into place. To evaluate the log data and find abnormalities, this framework will employ statistical techniques, machine learning algorithms, and natural language processing (NLP) [5]. Based

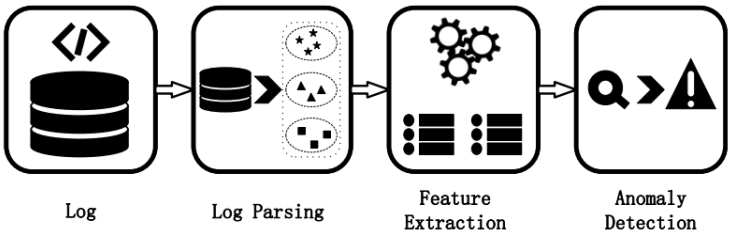


Figure 2.1: The process of log analysis and anomaly detection

on the features of the log data and the research goals, suitable algorithms and methodologies will be chosen as part of the framework design. To extract useful information from the log data, techniques like text mining, pattern recognition, clustering, and classification may be used. The framework will be put into action by creating software modules or making use of already-existing tools and libraries to carry out automated log analysis and anomaly identification Figure 2.1. Depending on the needs and level of expertise of the researcher, programming languages like Python or R may be utilized for implementation.The log data will be subjected to preprocessing procedures and data cleaning processes in order to guarantee the quality and accuracy of the analysis. Data normalization, the elimination of unimportant details, the treatment of missing values, and the handling of noisy or inconsistent data are some examples of preprocessing techniques. These preprocessing methods will aid in raising the caliber and dependability of the outcomes of automated log analysis.

Methods for cleaning data may include approaches for error correction, outlier detection, and elimination of duplicate log entries. These techniques are designed to guarantee that the log data is precise, consistent, and analysis-ready. The log data will be ready for the automated log analysis and anomaly detection process by applying the proper preprocessing procedures[9] and data cleaning processes, assuring trustworthy and useful results.The methodology for this study includes gathering log data from pertinent sources, designing and implementing an automated framework for log analysis and anomaly identification, and using preprocessing and data cleaning procedures. This methodology will make it possible to examine how automated log analysis and anomaly detection improve the quality of software. The outputs of the framework will be analyzed and evaluated in

the following stage in order to determine how well it has improved the quality of software.

Chapter 3

Log Mining

3.1 Log Datasets

In automated log analysis and anomaly detection research, log files are essential. These datasets offer the essential data for creating and analyzing algorithms, models, and methodologies. They enable academics to examine and comprehend system behavior, events, and anomalies since they contain real-world log data produced by diverse software systems. Depending on the precise research aims and the characteristics of the targeted software systems, log collections might differ in size, complexity, and content. They may include logs from various systems, including operating systems, databases, network hardware, application servers, and web servers. The log datasets may contain records of user interactions, server activity, error messages, performance indicators, or security incidents, among other elements of system operation.

We have employed five big log datasets, totaling 16,000,000+ lines of log messages, from a variety of sources, including supercomputers such as BGL, HPC, distributed systems such as HDFS, Zookeeper, and standalone applications such as Proxifier, to enable systematic evaluations on the most cutting-edge log parsing approaches. The dataset HDFS ia briefly summarized in Table 3.1. Due to corporations' frequent reluctance to provide their production logs due to confidentiality concerns, logs are a rare source of data for research. With the kind

System	Logs	Length	Events
HDFC	11,000+	8∼25+	29

Table 3.1: Data of HDFC System

assistance of their authors, we were able to get three log datasets. BGL, which has 131,000+ processors and 32,000GB of memory, is an open dataset of logs gathered from a BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL). A high performance cluster with 49 nodes and 6,100+ cores and 128GB of memory at Los Alamos National Laboratory, known as HPC, is also an open dataset with logs. A big data analytics framework [10] uses a 203-node cluster on the Amazon EC2 platform to collect HDFS logs. We also gathered two datasets: one from the desktop application Proxifier and the other from a Zookeeper installation on a 32-node cluster in our lab, in order to augment the log data for evaluation purposes. Particularly, the HDFS logs from [10] have established anomaly labels that each indicate whether or not a request for a data block operation is an abnormality. The labels are created based on domain expertise and are suitable for our studies of anomaly detection with various log parsers. The dataset, which contains over 11 million log messages, specifically records 575,061 operation requests with a total of 29 event kinds. 16,838 of the 575,061 requests totaled are flagged as abnormalities, which we utilize as the basis for our evaluation.

3.2 Log Parsing

There are numerous log parser techniques that we can use to handle a large number of log messages. Describe them briefly and determine which method can process data more quickly [9].

1. Sequence Log Compression and Transformation (SLCT): SLCT is a log parsing technique that concentrates on obtaining structured data from logs with sequential patterns. Sequence log transformation and compression are used in a two-step method. SLCT recognizes frequent subsequences in log lines and displays them with placeholders during the compression stage. SLCT increases parsing

effectiveness and cuts down on redundancy by compressing related log lines.

By taking into account the placements and values of placeholders, SLCT translates the compressed log lines to a predetermined event template in the transformation stage. The mapping procedure aids in the extraction of structured data from the log entries.

SLCT can accurately parse logs with a variety of message content and is good in managing logs with repetitive sequences.

2. IPLoM (Iterative Partitioning Log Mining): IPLoM is a method for processing log files that automatically identifies log patterns using an iterative process. It runs through several cycles to gradually improve the log parsing results. IPLoM divides log lines into segments based on tokens or delimiters that are used frequently in each iteration. It groups log lines with comparable structures together using a clustering method.

IPLoM then derives log patterns from each cluster by taking into account the similarities between log lines. To find patterns, it makes use of methods like longest common subsequence and frequent itemset mining.

In the next iterations, IPLoM repeats the partitioning and pattern extraction steps on the remaining logs after removing the parsed log lines from the original log file. Until all log lines have been parsed, this iterative procedure keeps going.

IPLoM can handle a variety of log formats and adapt to various log structures by iteratively improving the parser outcomes.

3. LKE (Log Key Extraction): LKE is an unstructured log parsing technique that was created exclusively for key extraction from log data. It focuses on locating and extracting crucial properties or fields from log messages that contain significant data. To train models using labeled log data, LKE uses machine learning methods, particularly CRF (Conditional Random Fields) models. Based on the context and patterns found in the training data, the models are trained to recognize important fields in log messages.

LKE uses the trained models to locate and extract critical fields from fresh log messages as it parses them. To accurately extract the pertinent information, it

takes into account the word order, their placement, and the previously acquired patterns.

LKE is especially helpful for extracting important data from logs where the structure or format could differ, like free-form log messages or log files with diverse templates.

The SLCT, IPLoM, and LKE log parsing methods use various strategies and techniques to overcome the difficulties of log parsing, increasing the precision and effectiveness of extracting structured data from log files.

Chapter 4

Anomaly Detection

It is possible for machines to learn on their own from experience without explicit programming thanks to the application of mathematical modeling and computer algorithms in the field of machine learning. This area of research, which is included under artificial intelligence, makes use of algorithms to find hidden patterns in data. Machine Learning permits the prediction of desired outputs by utilizing previous data associated with a particular problem. There are three different types of machine learning: Reinforcement learning, supervised learning, and unsupervised learning.

To create predictions, machine learning (ML) [11] uses statistical models. Classifying whether a specific log event, or combination of log events, is generating a genuine incident that needs attention may be a valuable prediction for log analysis. Finding the event or events that serve to explain the underlying cause of a problem could be another useful forecast.

The more data that is available, the more accurate the machine learning (ML) model will be at producing these predictions. Because of this, models usually improve in accuracy with time. However, there are two difficulties with this when it comes to logs: This results in a lengthy lead time to value, meaning that the system may need weeks or months of data to provide reliable forecasts. Even worse, when an application's behavior is constantly changing, such as when frequent updates are being deployed for each of its microservices, slow learning ML

actually isn't very helpful.ML models can be trained using supervised or unsupervised methods, respectively. A labelled data set, typically created manually by humans, is necessary for supervised training to enable the model to comprehend the source and effect of the data. The model will be able to identify that type of incident again if it observes the same log events or pattern, for instance, if we label all log events that correspond to an actual occurrence. Given the ever-growing number of potential failure modes with complex software applications, this can be quite time-consuming. Unsupervised training - In this method, the model tries to identify patterns and correlations on its own in the data set, which may subsequently be applied to make predictions.

4.1 Feature Extraction

There are frequently tens of thousands or even hundreds of thousands of logs in the parsed log template sequence. The log sequence that is produced after log parsing needs to be partitioned in order to make it easier to extract log features and carry out further processing. According to the log event occurrence time or a unique identifier, the log is divided into blocks of a certain size. It can be classified into time grouping and session grouping depending on the grouping method. Using a self-defined time span, time grouping separates logs into sequences [12]. Depending on whether the time spans overlap, it can be further divided into fixed time windows and sliding time windows.

Following log parsing, the generated log templates are transformed into numerical data that is machine-understandable. Digital Features entail the extraction of pertinent data from log fields, including timestamps, log levels (such as ERROR and INFO), source IP addresses, error codes, and other structured data that may be contained in the log. Digital characteristics give log events quantifiable data that can be utilized for additional research.

Indeed, to utilize machine learning algorithms effectively, the log statements need to be converted into numerical features. Here is a detailed explanation of the features extracted from the parsed log statements:

Blockid: Each unique blockid is assigned a numerical value ranging from 1 to

1008. A dictionary is created to map the blockid values, allowing for efficient representation and processing of this information.

Location: The location feature is already in a numerical format, so it can be directly used as is without further processing.

IP: Similar to the blockid, each unique IP address is assigned a numerical value ranging from 1 to 106. A dictionary is created to map the IP addresses to their corresponding numerical values.

Port: The port number is also represented as a numerical value, and no additional processing is required for this feature.

Time: The date and time variables in the log statements are converted into seconds. The first log statement is considered as the reference point or time zero, and subsequent log statements are represented as the time elapsed since the first log statement. This allows for the representation of time as a numerical feature.

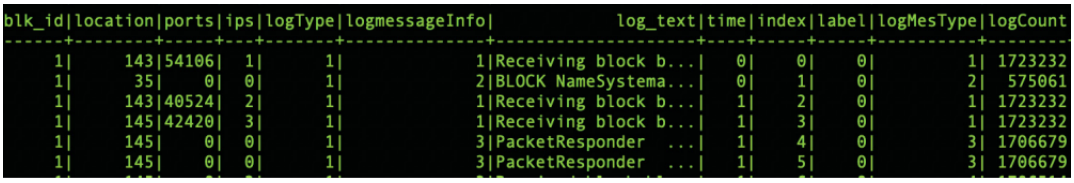
Log Type: A dictionary is created to map each unique log type to a numerical value. This helps in representing different log types in a numerical format, enabling the machine learning algorithms to process and analyze them effectively.

Log Message Info: Similar to the log type, a dictionary is created to map each unique log message info to a numerical value. This feature captures additional information related to the log messages and represents them numerically.

Log Message Type: The log messages undergo preprocessing to remove numeral values and retain only the constant part. Word embeddings, specifically the GloVe model, are used to vectorize the log statements. The GloVe model, pretrained on a large corpus of text data, is further trained using 930,000 lines of logs from the dataset. This enhances the quality of the word embeddings and improves the vectorization of log messages.

Log Count: For each log message type, a dictionary is maintained to store the log count value. This feature provides information about the frequency of occurrence of each log message type, which can be helpful in identifying anomalies or rare events.

All extracted elements displayed in Figure 4.1. By extracting these numerical features from the parsed log statements, the log data is transformed into a format suitable for training and applying machine learning algorithms. These features capture important information from the logs and enable efficient analysis and anomaly detection.



blk_id	location	ports	ips	logType	logmessageInfo	log_text	time	index	label	logMesType	logCount
1	143	54106	1	1	1	Receiving block b...	0	0	0	1	1723232
1	35	0	0	1	2	BLOCK NameSystema...	0	1	0	2	575061
1	143	40524	2	1	1	Receiving block b...	1	2	0	1	1723232
1	145	42420	3	1	1	Receiving block b...	1	3	0	1	1723232
1	145	0	0	1	3	PacketResponder ...	1	4	0	3	1706679
1	145	0	0	1	3	PacketResponder ...	1	5	0	3	1706679

Figure 4.1: Features Extracted Logs

4.2 Anomaly Detection Methods

4.2.1 SVM

Using supervised learning, the Support Vector Machine (SVM) is a classification technique. A hyperplane is built in SVM to divide different classes of examples in high-dimensional space. The goal of the optimization task in finding the hyperplane is to maximize the distance between the hyperplane and the closest data point for each class.Liang et al. compare SVM against other approaches in [13] and use it to detect failures. The training examples, like in logistic regression and decision trees, consist of event count vectors and their labels. If a new instance is discovered using SVM anomaly detection, it will be labeled as an anomaly if it is above the hyperplane and marked as normal otherwise. The two types of SVM are linear and non-linear, respectively. Because linear SVM performs better than non-linear SVM in the majority of our studies, we exclusively address linear SVM in this study.

4.2.2 Logistic regression

A statistical model for categorization that is frequently employed is logistic regression. Logistic regression calculates the probability p of each potential state

(normal or anomalous) to determine the state of an instance. A logistic function, which is based on labeled training data, determines the probability p. The logistic function could determine the probability p (0 a logistic function) that a new instance will occur. Once the model has been obtained, we enter a testing instance (X) into the logistic function to determine its possibility (p) of anomaly. If p 0.5, we label X as anomalous; otherwise, we label it as normal.

4.2.3 Decision Tree

The projected state for each instance is shown using branches in a decision tree, which is a tree structure diagram. The training data is used to build the decision tree top-down. The "best" attribute in use at the time, as determined by the information gain of each attribute, is used to generate each tree node [14]. For instance, the root node in Figure 4.2 reveals that our dataset has a total of 20 instances. The occurrence number of Event 2 is regarded as the "best" feature when separating the root node. As a result, the full set of 20 training instances is divided into two subsets based on the value of this property, one of which comprises 12 instances and the other of which has 8.

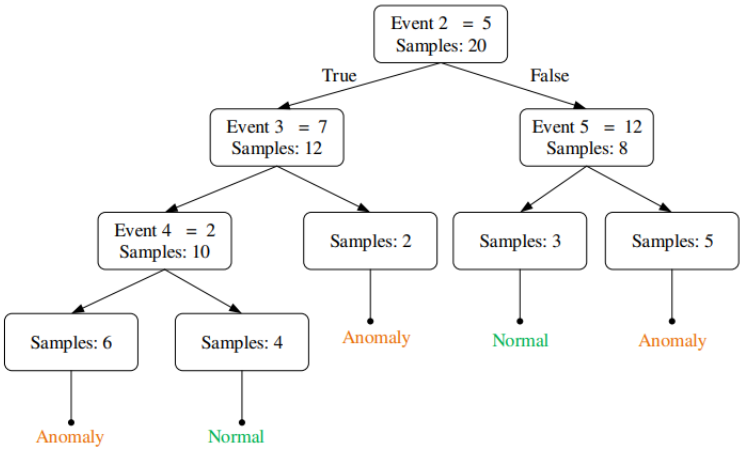


Figure 4.2: Decision Tree

Decision trees were used for the first time to diagnose failures in a web request log system in [10]. The decision tree is constructed using the event count vectors and their labels as specified in Section III-B. It goes through the decision tree

according to the predicates of each tree node it has already traversed in order to determine the state of a new instance. The instance will eventually arrive to one of the leaves, which will reflect the instance's current condition.

Chapter 5

Automated Notifications

A system that automatically sends emails to specific recipients in response to predefined triggers or events is known as automated email notification. It is frequently used in a variety of applications and sectors to give users or stakeholders timely updates, notifications, or alerts. Finding the trigger or event that will cause the email notification to be sent out is the first step in an automated email notification process. This may depend on a number of factors, including a particular user activity, a system event, a change in the data or status, or a predetermined time period.The system retrieves the pertinent data and configuration for the email notification after the trigger or event has been identified. This includes any additional information or attachments to include in the email, the recipient(s) or group(s) who should get the message, and the email template or content.Based on the selected template and content, the system generates the email message. This may include dynamic variables or placeholders that have data particular to the trigger or event put in for them. The email message can contain graphics, links, or other features and is commonly formatted using HTML or plain text. Based on the established settings, the system chooses the recipient(s) for the email notice. In order to do this, recipient data may be retrieved from user profiles, mailing lists, or databases. Individuals, teams, or dynamically selected recipients may be chosen based on a set of requirements or guidelines. The system sends the email notice using the proper email protocol (e.g., SMTP) after determining the email message and recipient(s). The sender's email server manages the transmission and

delivery procedure, and the email is transmitted to the recipients' email addresses.

5.1 Bug Tickets

In QA (Quality Assurance), a bug ticket is a formalized record or document that contains details on a software issue or defect that was found during the testing process. QA testers often create bug tickets [15], which are used for bug tracking and bug resolution between the QA team and the development team. Here is a list of the elements and details generally found in a bug ticket:

Title/Summary: The bug ticket begins with a title or summary that is clear and descriptive and highlights the primary issue or problem encountered. It ought to have sufficient details to enable easy comprehension of the bug.

Detailed information about the defect is provided in the description section. It outlines how to duplicate the problem as well as predicted and observed behavior. To aid engineers in understanding the issue, this part should be precise and easy to grasp.

Environment: The bug ticket details the situation where it was discovered. Information like the operating system, browser version, type of device, or any other pertinent system parameters are included in this. The environment details make it easier to reproduce the bug in the same setting.

Bug tickets frequently include a severity or priority rating to describe the significance and urgency of the bug. As a measure of the functional impact, severity levels might range from critical/blocker to minor/enhancement. Given their effect on the project, priority levels dictate the order in which problems should be fixed.

Attachments: If appropriate, bug tickets may contain addenda that give further information or context for the defect, such as images, log files, or videos. Developers can effectively comprehend and reproduce the issue with the use of these attachments.

Assignee/Owner: Bug tickets designate ownership to a particular programmer or team member who will be in charge of looking into and fixing the bug. This

guarantees unambiguous accountability and simplifies the bug-resolution procedure.

Bug tickets keep track of the status and development of bug fixes. There are several stages included here, such as "open," "in progress," "under review," or "resolved." The ticket receives frequent updates and comments to give visibility into the bug's lifespan and any pertinent conversations or discoveries.

Related Information: In order to provide extra context or background information regarding the defect, bug tickets may also include related user stories, feature specifications, or reference links.

All parties participating in the bug resolution process use bug tickets as a single point of reference. They support QA testers in tracking progress, ensuring that bugs are addressed and fixed in a timely manner to preserve the software's quality, and efficiently communicating found flaws to the development team.

5.2 SMTP

An Internet protocol called SMTP (Simple Mail Transfer Protocol) is used to transmit and receive email messages. By facilitating communication between email clients and servers, it is essential to the process of email delivery. Let's examine the SMTP protocol's operation, server configuration, authentication procedures, and the capabilities of the various SMTP commands. In order to send emails, an SMTP client connects to an SMTP server utilizing a client-server style of operation. The messages are received by the SMTP server, which then sends them to the targeted recipients. SMTP uses a store-and-forward protocol, which means that messages are forwarded from one SMTP server to another until they arrive at the intended server. By performing error checking, controlling message queues, and providing response codes to denote the success or failure of each step, the protocol provides dependable delivery.

It is possible to send anomaly detection messages or alerts quickly using SMTP (Simple Mail Transfer Protocol). Here is a description of how SMTP can deliver times for anomaly detection. establishing an anomaly detection system to keep an

eye out for any strange or anomalous behavior in the system or data. Detecting deviations from regular patterns may include employing machine learning algorithms, statistical analysis, or other methods. setting up the anomaly detection system so that it would send out notifications or alerts whenever an abnormality is found. This may be determined using predetermined thresholds, statistical models, or other standards unique to your method for detecting anomalies. The anomaly detection system generates a notification when an anomaly is found that includes pertinent details about the occurrence, including the timestamp, severity level, affected component, and any extra contextual information. acquiring the SMTP server's configuration information, such as the host address, port number, and authentication information (username and password) needed to connect and send emails. The system administrator or SMTP service provider often provides this data. using an SMTP communication-compatible library or framework to establish a connection, such as Python's smtplib package. Your anomaly detection system can communicate with the SMTP server using this connection. Making an email message with the anomaly detection details. Setting the subject line, body content, and email addresses of the sender and recipient may be necessary. Include pertinent information regarding the anomaly, its gravity, and any advised course of action.

You can immediately alert concerned parties about anomalies by using SMTP to transmit anomaly detection notifications, allowing them to respond promptly and take care of potential system concerns. The email sending protocol SMTP ensures that the anomaly detection times are transmitted effectively and efficiently by offering a dependable and commonly used mechanism.

Chapter 6

Results

The results of experimental evaluations that sought to compare the efficacy of the aforementioned strategies are presented in this section. Our algorithms were run on the HDFS log data on Amazon EC2 dataset to accomplish this goal. The HDFS logs from [**?**] in particular feature well-defined anomaly labels that individually indicate whether or not a request for a data block operation is an abnormality. The labels are created using domain expertise and are appropriate for our analyses of anomaly identification using various log parsers. With regards to the dataset, which contains approximately 11 million log messages, 575,000+ operation requests are recorded, totaling 29 event categories. 16,000+ of the 575,000+ queries are flagged as anomalous, which we utilize as the basis for our study [16].

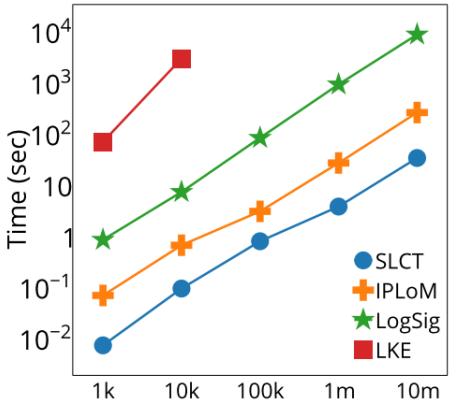


Figure 6.1: The effectiveness of Log Parsing Tools

By adjusting the quantity of raw log messages, we assess the log parsing algorithms' processing times across all datasets in Figure 6.4. Observe that there are more events as there are more raw log messages. The heuristic rule-based algorithms SLCT and IPLoM scale linearly with the quantity of log messages. Within five minutes, they could both parse 10 million HDFS log messages. LogSig also scales linearly as the quantity of log messages increases, unlike the other two clustering-based parsing techniques [17].

By adjusting the quantity of raw log messages, we assess the log parsing algorithms' processing times across all datasets in Fig. 6.1. Observe that there are more events as there are more raw log messages. The heuristic rule-based algorithms SLCT and IPLoM scale linearly with the quantity of log messages. Within five minutes, they could both parse 10 million HDFS log messages. LogSig also scales linearly as the quantity of log messages increases, unlike the other two clustering-based parsing techniques. In the study conducted to assess the processing times of log parsing algorithms, the quantity of raw log messages was adjusted to observe the scalability of the algorithms. Figure 6.4 shows the results obtained from the evaluation. One important observation is that as the number of raw log messages increases, the number of events also increases. This correlation is expected, as each log message typically represents an event or activity within the system. Among the log parsing algorithms evaluated, the heuristic rule-based algorithms SLCT and IPLoM demonstrated linear scalability with the quantity of log messages. This means that their processing times increased proportionally as the number of log messages increased. For example, within a five-minute timeframe, both SLCT and IPLoM were able to parse 10 million HDFS log messages. LogSig, another algorithm evaluated in the study, also exhibited linear scalability as the quantity of log messages increased. This is a favorable characteristic, as it ensures that the processing time remains predictable and manageable even with larger log datasets. In contrast, the two clustering-based parsing techniques, whose names were not specified, did not scale linearly with the quantity of log messages. This suggests that their processing times increased at a higher rate compared to the other algorithms. The reasons for this behavior could be attributed to the inherent complexities of clustering algorithms, such as the need

to analyze log patterns and identify clusters based on similarity.

The study highlights the scalability of the heuristic rule-based algorithms SLCT and IPLoM, as well as LogSig, in handling larger quantities of log messages. These algorithms provide efficient and predictable processing times, making them suitable choices for log parsing tasks in scenarios where scalability is a concern.

The goal of log parsing is to divide the log message's contents into constant (fixed plain text) and variable parts. Then, using all the constant message templates combined into a list of log events, structured logs with each log message matching to a different event can be created. The log template "Receiving block * src: * dest: *" is used to modify the log message 2 to "Event2" as an example. A log parser produces two files that contain structured logs and log events. In contrast to structured logs, which include events in chronological order along with their times of occurrence, log events list the extracted message templates.

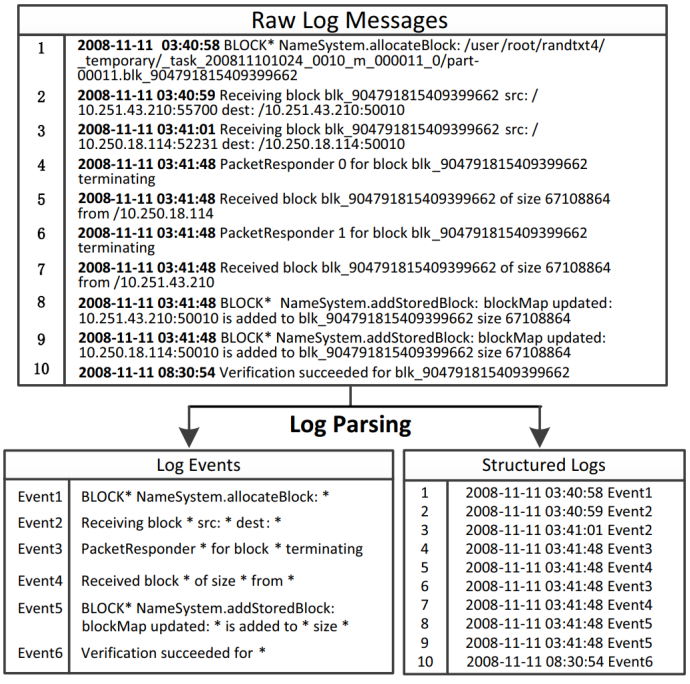


Figure 6.2: First ten lines from Amazon Log File

Each log event template can be converted into an index number, and the log index feature is created by combining the index numbers in order of execution time. Furthermore, because each number corresponds to the index of a different template in the log template sequence given in Figure 6.2, its log in-

dex vector maybe [Event1, Event2, Event2, Event3, Event4, Event3, Event4, Event5,Event5,Event6]. The log index feature is more frequently employed since, obviously, this approach can not only capture the number distribution of log events but also keep the positioning relationship of log events.

Here's a Python code example for an SLCT log parser that focuses on extracting error log entries in Appendix A 7.

For log parsers, there are numerous evaluation criteria, including precision, effectiveness, and so on. The flexibility and capacity for incremental learning of the parsers have also been compared and confirmed by researchers. The accuracy of log parsing is the most crucial characteristic for the log parser. It is not overstatement to argue that the success of downstream processes directly affects the accuracy of parsing. Log parsing is evaluated using typical metrics for machine learning tasks as Precision 6.0.2, Recall 6.0.1, and F-measure 6.0.3 [15]. By this parameters we will indicate anomalies. True Positive (TP): The number of events correctly classified as anomalous. False Positive (FP): The number of events wrongly classified as anomalous. False Negative (FN): The number of anomalous events misclassified as normal.

$$Recall = PT/PT + FN \tag{6.0.1}$$

$$Presicion = PT/(PT + FP) \tag{6.0.2}$$

$$F1 = 2 * Precision * Recall/Precision + Recall \tag{6.0.3}$$

PT stands for the quantity of logs that have been correctly parsed into the log template, FN for the quantity of incorrectly parsed logs into the current log template, and FP for the quantity of incorrectly parsed logs into other error log templates. We utilize precision, recall, and F-measure, three of the most popular metrics, to assess the accuracy of anomaly detection systems. The percentage of correctly reported anomalies [11] measured by precision, the percentage of

	Parcing Accuracy	Reported Anomaly	Detected Anomaly
SLCT	0,83	18,450	64%
IPLoM	0,99	10,997	63%
LKE	0,87	10,678	63%

Table 6.1: Anomaly Detection results with different Log Parsing Methods

accurately detected anomalies measured by recall, and the harmonic mean of precision and recall are all displayed here.

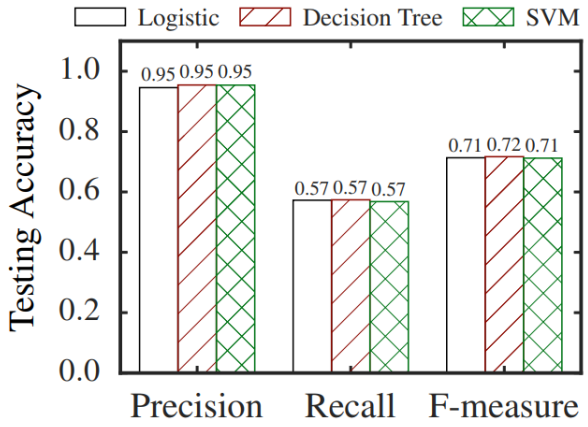


Figure 6.3: Anomaly Detection SVM, Decision Tree, Logistic

We analyze the benefits and drawbacks of various techniques to anomaly detection in order to assist developers better grasp the aforementioned three approaches and make better method selections. Labels are necessary for anomaly detection in supervised algorithms. The decision tree is easier to understand than the other two techniques since developers can find anomalies and provide logical justifications for them. Problems that are linearly non-separable cannot be solved by logistic regression, but they can be using SVM utilizing kernels. However, because some SVM parameters are difficult to set, creating a model frequently involves a lot of manual labor.

Methods for detecting outliers in time series data might aid in finding odd trends or abnormalities over time. The blue line on the graph reflects the typical approval rate for each hour of the day, while the x-axis represents the day of the month. Dashed lines indicate a typical degree of approval. The anomalies

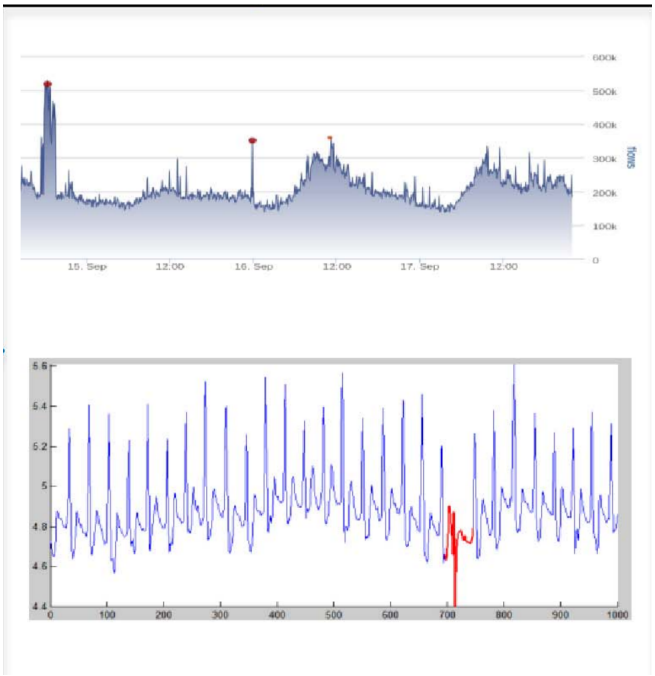


Figure 6.4: Anomaly Detection System

identified by the outlier detection techniques are represented by the red dots on the graph. These anomalies show occasions where the approval rate deviates greatly from the average rate and may be brought on by outside forces that have an impact on the approval rate. We opt for line graphs because time series data benefit greatly from their use. Using line graphs, you may visualize how a variable evolves over time. We can quickly identify any abrupt shifts or patterns in the approval rate that might point to the existence of an anomaly. We can look into these anomalies by identifying them.

For anomaly times in our anomaly detection system Figure 6.4, we need to create Bug Ticket for developers and testers in Table 6.2.

The significance of automated notifications in alerting system administrators or developers about anomalies lies in their ability to ensure timely awareness and prompt action. We can have tight control over our email data while avoiding paying for pricey email marketing firms or other third-party software by employing an SMTP server [18]. This is crucial for regulatory compliance since it guarantees that we follow stringent industry requirements. Overall, the choice to adopt SMTP was made after giving considerable thought to the organization's unique demands and resources. Additionally, I said earlier in my research that the Python

Title	Anomaly Detected
Description	Errors(event4) anomaly
Enviroment	Linux Log File
Severity	5
Attachments	2023-05-10 03:40:58 BLOCK* NameSystem.allocateBlock:
Asignee	Developer
Status	New
Related Info	

Table 6.2: Bug Ticket

programming language would be suitable for our situation. The SMTP functionality in Python can be implemented using the built-in module'smtplib'. We may connect to an SMTP server and send emails by using the server thanks to this module.

This sample of code illustrates how to send an email using the email.message and smtplib libraries from Appendix B of the Python SMTP 7.

When the system discovers anomalies, the installation of SMTP notifications to developers has a major positive impact on proactive problem solving and effective communication. Developers can receive prompt notification of abnormalities and be prepared to act right away by using the SMTP protocol to send email notifications.

SMTP notifications are a dependable and efficient way to inform developers about systemic problems. An automatic method causes an email message to be sent to the appropriate developers whenever anomalies, such as mistakes or strange behavior, are discovered. This makes sure that the appropriate people are alerted in a timely manner, enabling them to look into and quickly address the problems.

The connection between the system and developers is streamlined by the use of SMTP notifications for anomaly detection. It makes sure that developers are notified of important concerns, allowing them to efficiently set priorities for their jobs and manage resources. Developers have all the information they need to pinpoint the underlying causes of anomalies and launch the proper troubleshooting or issue-fixing procedures when they get bug tickets or thorough reports via email

notifications.

Additionally, SMTP notifications offer a traceable record of anomalies and the actions that followed. Organizations can use this material to track and audit activities, assess the frequency and character of anomalies, and make defensible judgments about system enhancements or optimizations.

In general, the deployment of SMTP alerts for anomaly detection and the subsequent development of bug reports or tickets for developers improves the effectiveness of issue resolution, allows better communication, and adds to the system's general stability and dependability. Anomalies are swiftly addressed, minimizing the negative effects on system performance and user experience.

Chapter 7

Conclusions

In conclusion, automated log analysis and anomaly reporting play critical roles in strengthening system monitoring and incident identification and response effectiveness. These systems can scan and analyze massive amounts of log data in real-time by utilizing machine learning techniques and clever algorithms, allowing for the early discovery of anomalies, errors, or security concerns.

Automated log analysis entails sifting through log files produced by various systems and applications and extracting pertinent data. With the use of this procedure, unstructured log data can be converted into understandable and structured representations for later analysis. Anomalies can be found by looking for departures from expected log patterns or predetermined thresholds using statistical and machine learning approaches.

In order to compare ML detection methods, it was necessary to assess each one's effectiveness in terms of log anomaly detection, classification precision, computational effectiveness, and scalability. All three techniques—SVM, Decision Trees, and Logistic Regression—proved successful at spotting log anomalies and achieving high classification accuracy. While Logistic Regression offers probabilistic insights, Decision Trees give interpretability, and SVM is known for its robustness and capacity to handle complex data.

The individual needs of the log analysis work must be taken into account while choosing a log parsing method and ML detection algorithm. Different ways to log

parsing are provided by SLCT, LKE, and IPLoM, each having advantages and disadvantages. For log analysis, SVM, Decision Trees, and Logistic Regression are trustworthy ML techniques, each with particular advantages. When deciding which option is best for a particular log analysis scenario, considerations including log complexity, dataset size, processing resources, desired interpretability, and the existence of noisy data should be taken into account.

System administrators or developers will receive messages when abnormalities are found thanks to the automatic anomaly notification component. This guarantees that the proper parties are informed as soon as possible, enabling them to take the necessary steps to address the problems. When delivering email notifications, the SMTP (Simple Mail Transfer Protocol) protocol is frequently utilized, offering a dependable and widely accepted form of communication.

Organizations stand to gain in a number of ways by incorporating automated log analysis and anomaly notification systems into the infrastructure. In the beginning, it offers proactive monitoring, allowing for early detection and control of problems before they worsen into significant concerns. Second, by automating the notification and log analysis procedures, it decreases manual labor and human error while enabling quicker response times. By finding and addressing anomalies that may affect system operation or security, it also enhances system performance and dependability.

In conclusion, the integration of automated log analysis and automated anomaly reporting offers businesses a reliable and effective method for spotting and handling anomalies in their systems. These systems make use of cutting-edge algorithms and technologies to assure the stability, security, and peak performance of the systems, which ultimately improves user experience and business results.

Bibliography

[1] M. Comuzzin. Detecting anomalies in business process event logs using statistical leverage. Inf. Sci, 548(1):50–67, 11 December 2019. URL https://www.sciencedirect.com/science/article/pii/S0020025520311038.

[2] Log analysis from a to z: A literature survey. Journal of Ohio State University, 6(1):1–104, 01 September 2021. URL https://repository.tudelft.nl/islandora/object/uuid.

[3] A. Wadekar A. Patil. Explainable lstm model for anomaly detection in hdfs log file using layerwise relevance propagation. IBSSC, 987(1):1–6, 01 Jul 2019. URL https://ieeexplore.ieee.org/document/8973044.

[4] Lakshmi Geethanjali Mandagondi. Anomaly detection in log files using machine learning techniques. Computer Sc., 93(1):740–750, 01 2007. doi: 10.25560/80802. URL http://www.diva-portal.org/smash/get/diva2:1534187/FULLTEXT02.pdf.

[5] Caroline Eriksson and Emilia Kallis1. Nlp-assisted workflow improving bug ticket handling. Industrial Engineering and Management, Royal Institute of Technology , 987(1):1–6, 01 Jul 2019. URL http://www.diva-portal.org/smash/get/diva2:1591483/FULLTEXT01.pdf.

[6] H. Qin D. Zou. Uilog: Improving log-based fault diagnosis by log analysis. Computer Science, 31(5):1033–1050, 05 2016. doi: 10.25560/80802. URL https://inria.hal.science/hal-01403114/document.

[7] Configuring automatic email notifications of events and reports. IBSSC, 9(1):1–6, 01 2021. URL https://www.netiq.com/

documentation/platespin-protect-11-2/protect_user/data/email-notification-events.html.

[8] Arailym Talasbek. Understanding re-finding behavior in naturalistic email interaction logs. SDU, 5(1):1–16, 04 2023.

[9] E. Ezat A.Awad. Extracting accurate performance indicators from execution logs using process models. Int.Conf.Comput.Syst.Appl., 93(1):1–7, 01 Nov 2015. doi: 10.25560/80802. URL https://ieeexplore.ieee.org/document/7507173.

[10] Y. Hui B. H. Park. A big data analytics framework for hpc log data: Three case studies using the titan supercomputer log. Proc. IEEE Int. Conf. Cluster Comput., (1):570–578, 01 Jul 2019. URL https://ieeexplore.ieee.org/document/8514917.

[11] The hybrid approach combining rule-based and machine learning techniques for effective log analysis and anomaly detection. Computer Scince., 2(1):1–7, 01 2018. URL https://dl.acm.org/doi/10.1145/3460345.

[12] J. Sosnowski D. Obrâbski. Log based analysis of software application operation. Int.Conf.Comput.Syst.Appl., 987(1):371–382, 01 Oct 2020. URL https://www.researchgate.net/publication/333038897_Log_Based_Analysis_of_Software_Application_Operation.

[13] Roberto Natella Domenico Cotroneo. Assessing and improving the effectiveness of logs for the analysis of software faults. Computer Scince., 90(1):740–750, 01 2014. doi: 10.25560/80802. URL https://www.researchgate.net/publication/224165666_Assessing_and_improving_the_effectiveness_of_logs_for_the_analysis_of_software_faults.

[14] M. Dunaev and K. Zaytsev. Logs analysis to search for anomalies in the functioning of large technology platforms. IBSSC, 97(11):31000–3123, 01 2019. URL http://www.jatit.org/volumes/Vol97No11/18Vol97No11.pdf.

[15] Caroline Eriksson and Emilia Kallis1. Nlp-assisted workflow improving bug ticket handling. Industrial Engineering and Management, Royal In-

stitute of Technology (KTH), 987(1):1–6, 01 Jul 2019. URL http://www.diva-portal.org/smash/get/diva2:1591483/FULLTEXT01.pdf.

[16] A. Juvonen and T. Sipola. Explainable lstm model for anomaly detection in hdfs log file using layerwise relevance propagation. Comput. Netw, 91(1): 45–56, 01 2015. URL https://ieeexplore.ieee.org/document/8973044.

[17] H. Li S. Locke. Logassist: Assisting log analysis through log summarization. IEEE Trans. Softw. Eng, 30, 5 May 2021. URL https://ieeexplore.ieee.org/document/9442364.

[18] M. Harvey D. Elsweiler. Understanding re-finding behavior in naturalistic email interaction logs. Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf., 5(1):35–44, 01 2011. URL https://www.researchgate.net/publication/221301374_Understanding_re-finding_behavior_in_naturalistic_email_interaction_logs.

Appendix A

```
import re

def slct_log_parser(log_dataset):
    log_templates = {}
    parsed_logs = []

    # Step 3: Log template extraction and counting
    for log_entry in log_dataset:
        # Custom function to extract log template
        log_template = extract_log_template(log_entry)
        if log_template in log_templates:
            log_templates[log_template] += 1
        else:
            log_templates[log_template] = 1

    # Step 4: Sort log templates based on counts
    sorted_templates =
    sorted(log_templates.items(), key=lambda x: x[1], reverse=True)

    # Step 5: Set threshold for log template selection
    threshold = 10  # Adjust the threshold as per your requirement

    # Step 6: Select top log templates based on threshold
```

```
    selected_templates =
    [template for template, count in sorted_templates[:threshold]]

    # Step 7: Log template mapping and parsing
    # Custom function to extract log template
    for log_entry in log_dataset:
        log_template = extract_log_template(log_entry)
        if log_template in selected_templates:
            parsed_logs.append(log_template)

    # Step 8: Return parsed log sequences
    return parsed_logs

# Example usage
log_dataset = [
    "ERROR: Connection timed out",
    "INFO: Application started",
    "ERROR: Invalid input received",
    "WARNING: Disk space running low",
    "ERROR: Database connection failed"
]

parsed_logs = slct_log_parser(log_dataset)
print(parsed_logs)
```

setlength-.5in

Appendix B

```
from email.message import EmailMessage
import smtplib

# Define sender and recipient email addresses
sender = "your_email@example.com"
recipient = "recipient_email@example.com"

# Define the message to be sent
message = bugticket.xls

# Create an EmailMessage instance
and set the sender, recipient, subjec, content
email = EmailMessage()
email["From"] = sender
email["To"] = recipient
email["Subject"] = "Urgent: Anomaly detected"
email.set_content(message)

# Set up an SMTP server and login to the sender's email account
smtp = smtplib.SMTP("smtp-mail.outlook.com", port=587)
smtp.starttls()
smtp.login(sender, "password")

# Send the email
```

```
smtp.send_message(email)

# Close the connection to the SMTP server
smtp.quit()
```