Literature Review: Advancements and Challenges in Log-Based Anomaly Detection for Large-Scale Cloud Systems

Divyansh Agrawal

Supervisor: Andriy Miranskyy

June 24, 2025

1 Introduction

Large-scale cloud-based software systems, including Big Data processing ones, generate data variety and volume in system logs like never before. System logs are a rich source of runtime information critical for system monitoring, ensuring efficient operation, performance optimization, and identifying complex problems such as failures and security intrusions [1]. But the enormous bulk, velocity, and intrinsic disorganized nature of such logs render traditional manual inspection extremely inefficient and prone to mistakes [2]. This cyclical problem is the driving force behind the compelling need for sophisticated automated log analysis, with an increasing focus on anomaly detection [3].

2 Literature Review

This literature review will critically examine seminal and cutting-edge studies in log parsing and anomaly detection from logs, with a focus on key methodologies, their drawbacks and advantages, and how the present project aims to complement the current gaps within this ever-evolving environment. Below, we provide an overview of the papers related to our work, starting from log parsing in Section 3 to anomaly detection in Section 4. We then contrast these works with ours in Section 5.

3 The Foundational Role of Log Parsing

A necessary step in enabling nearly all log analysis automation techniques is log parsing, which transforms raw free-text log messages into structured machine-readable formats [4]. This critical step allows other analysis procedures such as filtering, grouping, and applying machine learning models [5].

3.1 An Evaluation Study on Log Parsing and Its Use in Log Mining

He et al. [6] mention log parsing as a significant yet overlooked step in log analysis automation. The main aim of the paper was to comprehensively evaluate various automated log parsers' performance and, more importantly, investigate their impacts on subsequent log mining activities. The authors observed a significant gap where users would re-run or re-design log parsers since they did not have sufficient information regarding weaknesses and strengths of existing methods. Their method involved packaging four notable log parsers (SLCT, IPLoM, LKE, and LogSig) into a reusable toolset and thoroughly testing them over five diverse datasets containing over ten million raw log messages. Perhaps the most significant finding was their empirical demonstration of parser performance and thorough examination of their effectiveness in an empirical log mining case study. The strength of this work is its empirical approach and bottom-line analysis, providing valuable concrete recommendations to researchers and practitioners in general- such as preferring Drain for high accuracy for online purposes, using IPLoM for structured log formats , hence bolstering the foundations of log mining. Its weakness is that the paper is comparing parsers that were built up to 2016; newer, perhaps more advanced parsing techniques (e.g., deep learning-based LogRobust [7]) could have since emerged that would impact on the best option for modern log mining pipelines. Still, though, the paper provides a foundation guideline in the understanding of log parsing inner dynamics and realities directly applicable to our project by highlighting the importance of parsing accuracy in effective downstream anomaly detection.

3.2 Drain: An Online Log Parsing Approach with Fixed Depth Tree

He et al. [5] created Drain, a critical online log parsing method created to process high-volume, unstructured system logs effectively in real time, which is essential in effective Web service management. The paper sought to overcome the limitations of previous offline, batch-based parsing methods that became more and more impractical since there was an exponential increase in the amount of log data. Drain's primary strategy is to employ a specially designed fixed-depth parse tree to enable faster and online log processing by keeping a compact representation of the log templates. The key conclusions were Drain's superior accuracy with the top performance on four of five tested data sets and significant performance gains (51.85% to 81.47% reduction in running time over state-of-the-art online parsers). One of its powers is that it was proven to work well as a preprocessing step for later log analysis steps like anomaly detection, and therefore extremely well tailored for real-time cloud streaming log data. One potential weakness is that its fixed tree depth structure does not handle extremely complex or extremely variable log structures far away from standard patterns well, though experiments show resilience in standard cases. Drain's application to our project is important because it is a robust and efficient parsing

method well adapted to the needs of large-scale real-time cloud systems, and it is a likely candidate method for structuring the log data in our chosen LogHub datasets.

3.3 System Log Parsing: A Survey

Zhang et al. [4] provide an extensive overview of existing log parsing techniques, which facilitates an understanding of the tooling ecosystem for dealing with complex information and communication systems. The authors reference the increasing system complexity and reliance on system logs for administration, commenting that while manual log examination is impossible with the amount of data, automated systems are inclined to require normalized input that raw logs cannot offer. Its classification categorizes solutions as clustering-based, frequent pattern mining-based, heuristic, and program analysis and describes the methodologies, performance profiles, and operational characteristics of 17 open-source log parsers. The purpose of the work is to aid practitioners select or construct parsing solutions, and its key finding is the extensive solution space, with varying performance and operational attributes. One advantage of this work is that it is comprehensive in taxonomy and empirical examination and therefore provides a helpful guide for one coming into the field. Its use for this project is foundational in that it provides a full overview of parsing techniques, informing the need of this initial step and directing possible choices for log structuring for our Big Data log analysis. A limitation is that while it provides a broad survey of techniques, it does not make direct quantitative comparisons that imply a "best" parser for diverse scenarios, so real-world benchmarking has to be done.

3.4 Tools and Benchmarks for Automated Log Parsing

Zhu et al. [8] address directly the critical need for fast log parsing in the face of fast-increasing log sizes produced by modern software systems. This paper emphasizes log parsing as an imperative step prior to automated log analysis, based on the impossibility of manual inspection on enormous volumes. Their greatest contribution is a thorough comparison study of automated log parsing, along with making open-source tools and the massive LogHub benchmark dataset publicly available. They evaluated 13 log parsers on 16 real-world log datasets, such as HDFS, BGL, Zookeeper, OpenStack, and Thunderbird, with respect to accuracy, robustness, and efficiency. Perhaps the most significant finding is the extensive benchmarking results and experiences of an industrial use at Huawei, which provides useful guidance for the use of automated log parsing in practice. The empirical rigor of this paper and LogHub, the primary source dataset of this current work, is that it directly addresses the previous absence of publicly available, real-world logs. A limitation is that while it provides full benchmarks for the parsers until 2019, the domain of log parsing, especially with the advent of large language models, is very dynamic. The relevance of this paper to our project is essential because it provides the baseline benchmark

datasets and evaluation metrics for testing our anomaly detection models and guides selecting an appropriate log parsing method initially.

3.5 Log Parsing: How Far Can ChatGPT Go?

Le et al. [9] investigate the capability of ChatGPT, an LLM, for log parsing automation. The authors emphasize the significance of log parsing and examine whether LLMs can serve as a viable alternative to conventional data-driven approaches that consume significant training or fine-tuning. Their method involved testing ChatGPT on 16 LogPai benchmarking datasets (from LogHub) with various prompting techniques (zero-shot and few-shot). Zero-shot means the model is given a task without any prior examples, whereas few-shot provides the model with a few example inputs and outputs to help it understand the task before generating its own answers. One of the key observations was that ChatGPT was good with hopeful performance, particularly with few-shot prompting, and in zero-shot, it outperformed Logram and matched Drain's performance on Group Accuracy (GA). It significantly improved in Message Level Accuracy (MLA) and Edit Distance (ED). Group Accuracy (GA) measures how accurately log messages are grouped into the correct templates. Message-Level Accuracy (MLA) evaluates whether each individual log message is parsed into the correct template. Edit Distance (ED) counts the number of insertions, deletions, or substitutions needed to transform the parsed template into the correct one — lower is better. A strength is the potential of LLMs to reduce the demand for enormous labeled training sets and computation previously associated with data-driven parsers, with the prospect of more adaptive solutions. A limitation is the novelty of this usage, with full practical deployment implications, cost-effectiveness, and real-time response still being explored with respect to conventional parsers. This paper is relevant to our project as it investigates a cutting-edge approach to log parsing and offers another option above traditional methods, which could be utilized for the enlargement of future work or as a comparative basis in the preprocessing process, particularly on handling unstructured elements. However, unlike this study which focuses on LLM-based log parsing by ChatGPT, our work emphasizes feature engineering methods across multiple(5) structured datasets, aiming for efficient and reproducible anomaly detection in real-world cloud systems.

4 Evolution of Log-Based Anomaly Detection Methodologies

Having structure in logs at hand, the second fundamental challenge is to find anomalous patterns. Old data mining or statistical methods were employed in early anomaly detection with log data [1]. However, these methods get overwhelmed by the scale, high dimensionality, and dynamics of modern system logs and therefore miss complex temporal and contextual dependencies [2].

4.1 Experience Report: System Log Analysis for Anomaly Detection

He et al. [1] address the imperative of effective automatic anomaly discovery in big distributed systems, in which log examination becomes impossible due to astronomical volume and intricacy. The purpose of the paper is to bridge industrial practice and academy by offering an integrated review and empirical examination of six state-of-the-art log-based anomaly detection methods (three supervised and three unsupervised). Their approach contrasted these diverse techniques in detail on two publicly released production log datasets of more than 15 million log messages and more than 365,000 anomalies. A central strength of this paper is its applied contribution: it offers dense comparative analysis and results from its large-scale evaluation, and notably, releases an open-source toolkit to allow these methods' reuse and deployment by developers and operators. This study is highly applicable to our project since it provides practical guidance in selecting a method, highlights the real-world complexity of anomaly detection, and demonstrates the merits of empirical comparison. Its limitation is that it provides a firm snapshot of methods as of 2016, prior to the major breakthroughs in anomaly detection using deep learning and hybrid approaches that have been made since then.

4.2 Supervised Learning for Log-Based Anomaly Detection

Meghana et al. [10] address the pressing need for accurate log-based anomaly detection, advocating for supervised learning as a more accurate and interpretable replacement for popular unsupervised methods. The authors aim to counteract the "blind spots" and lack of clarity of unsupervised approaches. Noise removal and parsing the logs to extract data into data frames were their way of preprocessing the BGL log dataset. To draw features, they observed the occurrence of some specific keywords in log messages, binned by Block IDs. They also worked on their anomaly detection model using a Random Forest Ensemble learning algorithm since it is efficient in reducing variance and overfitting. The model worked really well with 99% accuracy and high precision, recall, and F1-scores for all classes. One of the advantages of this paper is that it centers around the high accuracy and interpretability of supervised learning if labeled data are given. One potential limitation is the inherent reliance on labeled data, which are usually not plentiful in actual-world anomaly detection scenarios, and the potential lack of capability to generalize a keyword-frequency-based feature set across heterogeneous log environments without extensive domain-specific engineering. This study applies to our project via the showcasing of supervised methods' potential with the availability of labels and the stimulation of our investigation into various log features and feature engineering, particularly frequency-based features, as a point of reference.

4.3 DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning

Du et al. [11] propose DeepLog, an LSTM-based deep neural network model, in a bid to enhance anomaly detection and diagnosis through the ability to treat logs as natural language sequences. The primary intention is to extend beyond typical data mining methods by automatically learning normal log patterns from complex and evolving contemporary system logs. Their method is to tokenize raw log records into "structured log keys" and parameter value vectors, and train separate LSTM models on log key sequences (for execution path anomalies) and parameter value sequences (for performance anomalies). DeepLog has the benefit of having dual utility both for detecting anomalies and aiding diagnosis by constructing workflows from underlying system logs for ease of root cause analysis. Additionally, it also supports incremental online updates, adapting to evolving system behavior as well as user feedback. While the paper demonstrates great detection capability on large datasets like HDFS and OpenStack, actual limitations include its reliance on system-specific training data, sensitivity to incomplete coverage of normal behavior in the training set, and reduced effectiveness when encountering unseen log templates or variable-length sequences. DeepLog's combination of log semantics with sequential modeling renders it very pertinent to our project's objective of knowing how sequence modeling can enhance detection performance.

4.4 LogAnomaly: Deep Learning for Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs

Meng et al. [12] present LogAnomaly, a new, unsupervised deep-learning framework that can simultaneously detect sequential and quantitative anomalies in unstructured log streams. Their objective was to outperform the limitations of existing approaches relying solely on log template indexes that often create false positives and overlook semantic associations. LogAnomaly utilizes template2vec, an embedding-based method, to acquire syntactic and semantic knowledge from log templates. It subsequently employs an LSTM network to train on both sequential (predicting the next log) and quantitative (indicating discontinuities in log counts) patterns from these template2vec representations. Its strongest aspect is its ability to handle newly emerging log templates by "approximating" them with vector similarity, minimizing retraining and feedback requirements. LogAnomaly's improved performance on HDFS and BGL datasets with improved F1 values and reduced false alarms demonstrates its efficacy. The sole limitation is that while template2vec is a significant advancement, its utility in identifying weak, domain-specific semantic relationships for all log variations remains an area of potential future development or optimization. LogAnomaly's unsupervised nature and focus on semantic interpretation make it extremely useful as a benchmark and inspirational idea for our project's exploration of alternative feature engineering techniques and models for anomaly

detection without the requirement of annotated anomaly data.

4.5 ADA: Adaptive Deep Log Anomaly Detector

Yuan et al. [3] present ADA, an online, unsupervised deep neural network model to detect system log anomalies precisely and effectively. Following the observation of the inefficiency of existing supervised and offline methods to handle dynamic attacks and heavy computational and storage demands involved in processing large logs, ADA utilizes LSTM networks for real-time anomaly detection. Another of its key contributions is the adaptive model selection mechanism for dynamically choosing Pareto-optimal network configurations to optimize resources and latency. Specifically, ADA also includes a dynamic threshold mechanism, adjusting the threshold of the anomaly detection based on existing observed log patterns and directly taking into account the log-normal distribution of the event losses. They also introduce a new direct storage decision methodology based on abnormal events themselves, with tremendous savings in storage cost. Compared to the Los Alamos National Laboratory cyber security data set, ADA performed an extremely good F1-score of approximately 95% and was significantly quicker (97 times) compared to its nearest substitutes at an extremely low storage expense. One of the weaknesses would be the a priori window assumption of "normal behavior" to bootstrap train that degrades performance in extremely turbulent or dirty environments from the beginning. ADA's focus on online adaptation, dynamic thresholding, and low resource usage is a great innovation for real-world log anomaly detection systems and thus highly relevant to the objective of our project of real-time anomaly detection and risk mitigation in cloud-based systems.

4.6 Improving Log-Based Anomaly Detection with Component-Aware Analysis

Yin et al. [13] identify a critical flaw in most existing log-based anomaly detection methods: they ignore the corresponding system component of a log message. Traditional methods using log template sequence only lose location sequence-based anomalies that reflect changes in system module interactions. The authors introduce LogC, a "component-aware analysis" method to identify unusual log sequences. LogC transforms unstructured log messages into log template sequences and component sequences and feeds both into a combined LSTM model that has been trained on only normal log sequences and therefore is an unsupervised learning method. A major advantage of LogC is its ability to capture out-of-pattern component workflows that are not visible from log template sequences alone. Experimental comparison on HDFS and Thunder-Bird data sets showed improved performance against baselines (PCA, IM, and DeepLog). One major limitation is its discrete attention to event sequences and the authors cite future work in integrating richer log features and attention mechanisms. LogC is highly relevant to our project, specifically motivating our

exploration of component patterns as a salient feature for detecting anomalies in distributed systems.

4.7 Log Based Anomaly Detection: Correlation Between The Logs

Sunil et al. [14] mention the continually growing complexity of logs in modern IT systems and the challenge this presents to traditional anomaly detection approaches. The authors argue that current research overlooks the interrelations and relationships between logs, which makes system diagnosis not possible. Their goal is enhancing parsing of logs and anomaly detection through learning and leveraging such relationships between logs explicitly with deep learning techniques. They utilize a multi-step process: parsing unparsed logs from multiple Apache projects (Hadoop HDFS, Hive, Zookeeper, ActiveMQ, Maven) to structured format, resolving original template IDs, and filtering variables to prevent log injection" or data corruption. The paper also introduces a new classification model for log relationships in terms of new categories such as "Step," "Strategy,"Exception,"Elaboration," and "Complement." One advantage is that it focuses on determining and encoding log dependencies as opposed to mere simple sequences, so that more is learned about underlying system behavior. One disadvantage is that its current approach, where it analyzes static Java projects, will not necessarily be trivially adaptable to other programming languages or to diverse logging conventions. This study is relevant to our project by showing the importance of semantic context and log relationships as potential predictive log features, which directs our exploration of more dense feature spaces for anomaly detection.

4.8 Hybrid CAE-VAE for Unsupervised Anomaly Detection in Log File Systems

Wadekar et al. [15] suggest a new solution to unsupervised anomaly detection from unstructured log files, but especially in big data settings such as Hadoop. Their aim is to bridge the gap of supervised approaches with labeled data and the inadequacy of conventional approaches to discover new types of anomalies. Their main approach is a hybrid Convolutional Autoencoder-Variational Autoencoder (CAE-VAE) model specifically tailored for discrete event sequences in log files. It's a two-step procedure: a CAE that is learned on a compressed image representation of an image of one-hot encoded log key sequences and then a VAE that is learned on the compressed image representation to produce an anomaly score directly in the form of a likelihood measure. Its primary advantage is to remain unsupervised so that it can operate on unlabeled data and identify anomalies without presuming beforehand their category, a primary requirement because real logs hold an enormous surplus of regular over anomalous data. Empirical comparison with HDFS logs proved the CAE-VAE model superior to a supervised CNN model as well as a stand-alone CAE. One likely

limitation is its existing focus on discrete sequences of events, to continuous events or handling variable system parameters directly from logs in future work. This paper relates to our project by providing a robust paradigm for unsupervised learning to be explored for baseline performance and to inform feature engineering strategies, that is, how the keys of logs are represented.

4.9 LogBERT: Self-Supervised BERT-Based Log Anomaly Detection

Guo et al. [2] present LogBERT, a self-supervised in-real-time log anomaly detection model that exploits Bidirectional Encoder Representations from Transformers (BERT). The authors are aware of the limitations of traditional machine learning and recurrent neural network (RNN) models in dealing with complex temporal and contextual relationships among sequences of logs. LogBERT also contains a transformer encoder that has been learned on two new self-supervised tasks: masked log message prediction to capture relationships between isolated log messages, and hypersphere volume minimization to keep the normal order of logs close to one another in the embedding space. It is strong in learning regular system behavior's good feature representations and subsequently detecting anomalies whenever system behavior differs from what the log sequences learn. The research showed the effectiveness of LogBERT on three diverse log datasets (HDFS, BGL, and Thunderbird), outperforming existing state-of-the-art anomaly detection methods. The innovativeness of LogBERT from self-supervised learning utilizing a Transformer model on log sequences is well in line with our project's exploration of sequence modeling and establishing baseline performance for next-generation anomaly detection techniques.

4.10 BERT-Log: Supervised System Log Anomaly Detection Using Pre-trained Language Model

Chen et al. [16] also propose BERT-Log, a supervised method that treats system log sequences as natural language and detects anomalies with a fine-tuned BERT model. The goal of this research is to address significant challenges for log-based anomaly detection—e.g., handling unstructured logs, maintaining semantic context, and employing appropriate sliding windows. Their method is tokenizing logs using Drain parser and log template to semantic vectors with BERT and then feeding them into a fully connected neural network to classify normal/anomalous sequences. Their method was evaluated using the HDFS and BGL benchmarks with outstanding F1-scores of 99.3% and 99.4%, respectively, and higher than current state-of-the-art. Some of the improvements are the good generalization robustness, which is observed with good performance even at 1% training data. Another improvement is their proposed log feature extractor for BGL logs with a sliding window augmented with node ID, by which the model can identify the anomalies better in distributed systems. One of the limitations is that it is supervised learning from pre-defined log templates, and this

could be its weakness in dealing with unknown or dynamic log format. BERT-Log's better performance, few parameters, and stability make it an attractive and scalable approach for real-time log anomaly detection, and hence a significantly useful contribution to our project's research on predictive log features and advanced feature engineering methods for distributed systems.

4.11 HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log

Huang et al. [17] also address crucial limitations of current log-based anomaly detection methods, i.e., that they possess zero ability to handle surprise log templates and that they are blind to parameter values. Noting that anomalies can occur not only in log template pattern but also as deviating parameter values, the authors propose HitAnomaly, a hierarchical transformer-based log-based anomaly detection model. The method makes use of two domain-specific encoders: a log template sequence encoder for context and semantic information from sequences of log templates, and an encoder for parameter value semantic information, e.g., interactions between parameters and templates. A main contribution is the usage of an attention mechanism to selectively combine representations from both encoders such that the model is allowed to assign different weights to sequences of logs or to parameter values. Mass-scale experimentation on HDFS, BGL, and OpenStack proves strong relative outperformance over peer state-of-the-art approaches with established robustness on dynamic log data and strong competency in handling unknown events. A strong advantage is its extreme testing of log semantics, structural flow as well as numeric/categorical values, and its ability to combat parsing errors through parameter value encoding. As it relies on a prior parsing step (i.e., Drain), the ability of HitAnomaly to eliminate part of the parsing noise is an added value. This is a major enhancement and directly connects with our research project in this area of predictive log attributes and feature engineering methods, namely how it learns complex patterns from sequences of logs and parameters.

4.12 Improving Log-Based Anomaly Detection by Pre-Training Hierarchical Transformers

Huang et al. [18] describe inherent limitations of using general-purpose pre-trained language models like BERT directly on log data for anomaly detection. They assume that log data have specialized word and domain distributions, and hierarchical structure (log statements in sequences) that models do not take advantage of. The authors' primary focus is to introduce HilBERT, a novel pre-trained log representation model for system logs, based on a hierarchical transformer model to converts raw logs to templates and then pre-trains HilBERT on the templates. What is special for HilBERT is that it has both a log encoder (for processing single log events) and a sequence encoder (for processing sequence-level information), and two new pre-training objective functions specifically designed to learn local and sequence-level patterns. One of its most

compelling aspects is its empirical comparison on HDFS (stable) and BGL (unstable) datasets, in which it demonstrates how HilBERT outperforms recent baselines like BERT, especially on unstable log data with unseen log events or sequences. They also analyze HilBERT's resilience to spurious logs through LogBug, with improved performance, and investigate the usefulness of "continual pre-training." Future limitation is the dynamic adjustment of the hierarchical model to logs of varying levels of granularity or poor hierarchical definitions. It is a nice contribution, a solid and good solution, and is closely aligned to our project's pursuit of new sequence modeling and feature engineering techniques for anomaly detection.

4.13 Log Sequence Anomaly Detection Based on Local Information Extraction and Globally Sparse Transformer Model

Zhang et al. [19] present LSADNET, locally sparse Transformer-based log sequence anomaly detection network, which consists of local information extraction and a globally sparse Transformer.The authors attempt to solve the problem of capturing local correlations between adjacent logs and long-range dependencies within log sequences while eliminating the effect of unnecessary information. LSADNET employs multi-layer convolution to extract local features and an extended Transformer structure for global dependency, enhancing the self-attention mechanism to support adaptive retention of informative information and noise removal. One of theirs is its novel log vectorization using log template transfer values that beats existing state-of-the-art methods on public benchmarks. The focus of the paper on local and global dependencies in log sequences and its novel log vectorization technique makes it critically important to our project on sequence modeling and feature engineering for enhancing anomaly detection performance, particularly for complex distributed system logs.

4.14 Log-based Anomaly Detection Without Log Parsing

Le et al. [20] introduce a new anomaly detection approach, NeuralLog, which takes advantage of log data without the conventional error-prone and time-consuming log parsing procedure. The authors recognize that existing approaches are significantly hindered by parsing errors, resulting from poorly handled out-of-vocabulary (OOV) words or ambiguities from logs' semantic nature. Log parsing step has the tendency to make anomaly detection techniques lose useful information. NeuralLog is distinct in that it directly learns the semantic content of the raw log message and represents such semantic content as semantic vectors using a pre-trained language representation model (e.g., BERT). These semantic vectors can be fed into a classification model based on Transformers that can "see" contexts framed by previous log messages in a log message sequence. Having been tested on four publicly available datasets, its performance shows NeuralLog outperforming state-of-the-art approaches (F1-scores > 0.95),

confirming both NeuralLog's ability to accurately seize the semantics of the log messages and the model's ability to determine whether a log event is an outlier, without making the parsing step a prerequisite. One of the strengths is the removal of the parsing step, which removes a significant source of error and complexity. One limitation could be computational overhead in processing raw text with large language models, especially in throughput-intensive environments. This work is highly relevant to our project by questioning the widespread need for log parsing and giving an opposing perspective on feature extraction and semantic interpretation of logs, having direct implications on our investigation of best log attributes.

4.15 Log-based Anomaly Detection Through CNN model with Parameter Entity Labeling for Improving Log Preprocessing Method

Sutthipanyo et al. [21] propose a new log-based anomaly detection mechanism that includes a Convolutional Neural Network (CNN) model and an enhanced log parsing method. The authors seek to improve existing log parsing by replacing parameters in log messages with their semantic entities (e.g. "$< IP >$" to represent an IP address) rather than simple placeholders. Directly incorporating semantic data into parsed logs with parameter entity labeling, the authors argue that they retain more semantic and contextual information, thus providing the input features to the CNN model with more richness. Experiments with ThunderBird and BlueGene/L datasets proved that while complete training datasets are of high recall, their method achieves similar precision and specificity but with slightly better overall detection performance than earlier log parsing methods, which indicates promising results regarding system dependability and security. A strength is its novel parameter entity labeling approach, which enhances machine learning model input feature representations with richer semantic content of structured logs, thus resulting in more effective feature representations. A weakness is the possible need for manual work to consistently define and label these semantic entities over various types of logs. This article is closely tied to our project, in that it informs our investigation of why log features are important and how more informative semantic data from parameter values can be engineered as features to improve anomaly detection.

4.16 LogGPT: Exploring ChatGPT for Log-Based Anomaly Detection

Qi et al. [22] present LogGPT, a log-based anomaly detection solution which investigates the application of ChatGPT on this important issue. Whereas traditional deep learning approaches are beset with issues of high-dimensional and noisy data, generalization, imbalance in classes, and interpretability, LogGPT avows to unlock the advanced language interpretation power of ChatGPT. The design is based on three core elements: log preprocessing (of unstructured raw

logs to structured data for ChatGPT), prompt construction (for setting up the anomaly detection query), and response parsing (for understanding responses from ChatGPT). LogGPT is directed toward knowledge transfer from big corpora to the field of system log analysis. The initial experiments on the BGL and Spirit datasets show encouraging results with good interpretability and provide very early evidence that prompt-based models like ChatGPT can spot anomalies in system logs. The strength is in the possibility of boosting interpretability and generality with the large knowledge base of LLMs, possibly reducing the requirement for excessive domain-specific feature engineering. The weakness is the preliminary nature of this work, with scalability, cost considerations, and fine-tuning for a specific log environment yet to be thoroughly explored. This paper is applicable to our project by showcasing an innovative direction in LLM-based anomaly detection, which may guide future work or be an extremely high-level point of comparison for our selected ML models.

5 Gaps and Opportunities: Positioning Our Research

5.1 RQ1: What is the baseline performance that traditional ML methods can achieve for anomaly detection in large-scale cloud systems?

Although deep learning-based methods like LogBERT [2] and BERT-Log [16] demonstrate high detection accuracy, they tend to behave like black boxes and have a high computational cost. In contrast, traditional machine learning models like Isolation Forest, Random Forest, Decision Trees and Hidden Markov Models (HMMs) are more explainable and offer resource efficiency. However, there is limited benchmarking of such classical approaches on real-world structured log datasets such as those of LogHub [8]. This motivates our project to empirically evaluate these baseline models on HDFS, BGL, Zookeeper, and other datasets for determining their real-world applicability and limitations in modern log anomaly detection.

5.2 RQ2: What log features (e.g., event templates, component patterns, time-based session attributes) are most indicative of operational anomalies?

Though recent approaches like HitAnomaly [17] and LogC [13] do utilize semantic and component-level information, most of the current research still widely relies on log templates or simple sequential patterns [11, 12]. A comprehensive comparison of different log features and their corresponding predictive capabilities is lacking. Our work seeks to address this lacuna by explicitly comparing various categories of features, including template sequences, component IDs, and temporal session features — for usefulness in anomaly detection across var-

ious system architectures. This will separate the most contributory features to anomaly prediction and how they aggregate in complex environments. The objective of RQ2 is to identify features that are specific to each dataset and to investigate whether any common, generalizable features exist across all datasets for detection of anomaly across them.

5.3 RQ3: How do feature engineering methods (e.g., session windowing, event clustering, and sequence modeling) improve the performance of anomaly detection for structured logs?

Feature engineering is an essential step for log-based anomaly detection, yet most papers either employ implicit methods (e.g., fixed windowing in DeepLog [11]) or overlook their influence. Although Drain [5] offers structured parsing amenable to sequence modeling, there is limited research that discusses how alternative engineering approaches influence anomaly detection results. Our work fills this gap by comparing windowing approaches (fixed vs. adaptive), encoding schemes (e.g., n-grams, Transformer-based embeddings) for their relative effect on detection accuracy, interpretability, and runtime efficiency.

5.4 Summary

Concisely, our research aims to bridge the gap between efficient but non-interpretable anomaly detection techniques and interpretable, efficient, and practical methods suitable for large-scale cloud environments. By addressing the three research questions outlined above, we anticipate producing actionable knowledge for enhancing reliability and risk reduction in distributed software systems.

References

[1] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *2016 IEEE 27th international symposium on software reliability engineering (ISSRE)*. IEEE, 2016, pp. 207–218.

[2] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 international joint conference on neural networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[3] Y. Yuan, S. S. Adhatarao, M. Lin, Y. Yuan, Z. Liu, and X. Fu, "Ada: Adaptive deep log anomaly detector," in *Ieee Infocom 2020-ieee Conference on Computer Communications*. IEEE, 2020, pp. 2449–2458.

[4] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. S. Chen, and F. Pianese, "System log parsing: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 8596–8614, 2023.

[5] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE international conference on web services (ICWS)*. IEEE, 2017, pp. 33–40.

[6] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in *2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2016, pp. 654–661.

[7] 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*, 2019, pp. 807–817.

[8] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 121–130.

[9] V.-H. Le and H. Zhang, "Log parsing: How far can chatgpt go?" in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 1699–1704.

[10] C. Meghana and B. Kariyapppa, "Supervised learning for log-based anomaly detection," in *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*. IEEE, 2024, pp. 1–4.

[11] 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*, 2017, pp. 1285–1298.

[12] 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*, vol. 19, no. 7, 2019, pp. 4739–4745.

[13] K. Yin, M. Yan, L. Xu, Z. Xu, Z. Li, D. Yang, and X. Zhang, "Improving log-based anomaly detection with component-aware analysis," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 667–671.

[14] S. Sunil, A. Suresh, and V. Hemamalini, "Log based anomaly detection: relation between the logs," in *2023 International Conference on Networking and Communications (ICNWC)*. IEEE, 2023, pp. 1–5.

[15] A. Wadekar, T. Gupta, R. Vijan, and F. Kazi, "Hybrid cae-vae for unsupervised anomaly detection in log file systems," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2019, pp. 1–7.

[16] S. Chen and H. Liao, "Bert-log: Anomaly detection for system logs based on pre-trained language model," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2145642, 2022.

[17] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "Hitanomaly: Hierarchical transformers for anomaly detection in system log," *IEEE transactions on network and service management*, vol. 17, no. 4, pp. 2064–2076, 2020.

[18] S. Huang, Y. Liu, C. Fung, H. Wang, H. Yang, and Z. Luan, "Improving log-based anomaly detection by pre-training hierarchical transformers," *IEEE Transactions on Computers*, vol. 72, no. 9, pp. 2656–2667, 2023.

[19] C. Zhang, X. Wang, H. Zhang, H. Zhang, and P. Han, "Log sequence anomaly detection based on local information extraction and globally sparse transformer model," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4119–4133, 2021.

[20] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 492–504.

[21] T. Sutthipanyo, T. Lamsan, W. Thawornsusin, and W. Susutti, "Log-based anomaly detection using cnn model with parameter entity labeling for improving log preprocessing approach," in *TENCON 2023-2023 IEEE Region 10 Conference (TENCON)*. IEEE, 2023, pp. 914–919.

[22] J. Qi, S. Huang, Z. Luan, S. Yang, C. Fung, H. Yang, D. Qian, J. Shang, Z. Xiao, and Z. Wu, "Loggpt: Exploring chatgpt for log-based anomaly detection," in *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2023, pp. 273–280.