

SYSTEM LOG ANOMALY DETECTION THROUGH SEQUENTIAL ANALYSIS

Garrett Devaney, Jun Liang Ho & Kevin Guernsey

College of Computing

Georgia Institute of Technology

Atlanta, GA 30332, USA

{gdevaney3, jho88, kguernsey3}@gatech.edu

ABSTRACT

United States critical infrastructure faces new cyber threats from nation-state actors in the form of Living-Off-The-Land (LOTL) attacks. LOTL attacks use native binaries and tools to carry out malicious actions that evade traditional rules-based intrusion detection systems (Lenaerts-Bergmans, 2023). System logs detail network states, events, and changes over time, offering insight into indicators of attack. This paper introduces a new approach to intrusion detection which leverages self-supervised Bidirectional Encoder Representations from Transformers (BERT) to detect out-of-distribution (OOD) network events using system log data. We show that previous state-of-the-art (SOTA) research inflated results using heuristics tied to specific datasets and developed two novel techniques to increase model robustness to diverse data: regularizing log anomaly scores through geometric and harmonic mean, and creating a tunable threshold that allows cybersecurity analysts to adjust model sensitivity to network dynamics.

1 INTRODUCTION

United States critical infrastructure faces new cyber threats from nation-state actors in the form of Living-Off-The-Land (LOTL) attacks. LOTL attacks, as opposed to traditional cyber-attacks, are fileless, meaning they rely upon native binaries and tools to advance and carry out attacks (Lenaerts-Bergmans, 2023). Traditional cybersecurity techniques use rules-based methods that only identify indicators of compromise (IOC) via known malware artifacts on networks, often missing detections of these sophisticated LOTL attacks.

Critical infrastructure in the United States is typically defended by intrusion detection systems which use both signature-based and anomaly-based detection (Hurst et al., 2014). Signature based detection systems rely upon patterns seen in previously discovered malware and cannot adapt quickly to zero-day exploits. Anomaly detection and baseline profiling are popular methods for spotting deviations from typical behavior in cybersecurity systems, notably those that safeguard vital infrastructure. By tracking devices, applications, and users over time, baseline profiling creates patterns of typical activity and leverages this information to identify behaviors that deviate from the norm.(Du et al., 2017).

These patterns and events created by the applications, devices, and users are denoted in logs, which help provide a chronological sequence of events during operation and show the data needed to perform failure analysis. Many traditional machine learning models have been developed in order to detect an anomaly using system logs as data. However, these models have not produced successful results due to the inability of the models to capture sequential information as some anomalies may appear normal in each log, but the sequence as a whole may actually be anomalous.

To alleviate this, some deep learning models have been built, many of which have used Recurrent Neural Networks (RNNs) to aid in capturing sequential information Brown et al. (2018). While these worked with the sequences, there were a few limitations that came with using RNNs. First, the biggest limitation was that the embeddings of the logs needed to include the context from both previous and succeeding logs, rather than only the previous logs as used in RNNs. Second, RNNs

predict if a sequence is anomalous based on if it can correctly predict log messages based on the previous logs, which is an inaccurate way of classification since. (Brown et al., 2018)

Thus, LogBERT was created using Bidirectional Encoder Representations from Transformers (BERT). Using BERT allows for a contextual embedding for each log that would capture information from the whole sequence. (Guo et al., 2021) However, LogBERT does not come without its own flaws. Since logs are naturally ever-changing, LogBERT cannot adapt to new tokens and may wrongly flag sequences with new tokens as anomalous. Studies on LogBERT also show that the performance of LogBERT drops when the data itself is more unstable (Hofman, 2023).

To help alleviate this, we conducted an experiment in which we created a modified architecture for LogBERT with the aims of increasing the robustness of LogBERT. We ran this experiment on three different public datasets, and compared our results to both the proposed original LogBERT results and our own observed LogBERT results. Our experimental results show that our changes to the LogBERT architecture lead to better accuracies in predicting anomalous logs.

2 RELATED WORKS

2.1 VoBERT

2.1.1 DEALING WITH UNSTABLE LOGS USING VOCABULARY-FREE MASKED LANGUAGE MODELING

Vocab-free BERT, or VoBERT (H., 2023; Hofman, 2023), is an anomaly detection method for sequences that builds on LogBERT. It claims to tackle several problems faced by LogBERT, the first of which is that LogBERT does not work well with unstable logs - logs that change in terms of content and in terms of messages (that are both removed and added). VoBERT tackles these unstable logs by introducing a different pre-training objective called Vocabulary-Free Masked Language Modeling (VF-MLM) which removes the constraint of having a fixed vocabulary - this means that it can handle log keys that have unseen vocabulary since there is no reliance on any vocabulary. This works by introducing a separate sentence encoder model that can embed log keys within a space without a need for vocabulary knowledge.

We highlight two major changes that directly dealt with the above, and contained insights which could inform our current approach:

- **MLM Architecture:** VoBERT removes the Feed-forward Neural Network (FFN) and softmax function which are present in the traditional MLM model architecture. This outputs a raw contextual embedding vector for each token, instead of computed probability distributions over a fixed vocabulary. This removes reliance on a fixed vocabulary as well as creates a dense representation with both semantic and syntactic information for that given token, which can then be used in downstream tasks.
- **Element embedding:** Traditional MLM makes use of a matrix embedding layer, which learns an embedding vector of tokens during training and outputs a useful embedding during prediction only if it has been trained on it prior - any unseen tokens will be tagged with a [UNK] vector. VoBERT removes this reliance by making use of SentenceBERT, a pre-trained model which creates embeddings for English language sequences as the semantic embedding layer. On unseen tokens, SentenceBERT likewise generates an embedding that captures the semantic and syntactic meaning of them that provides more use than a generic [UNK] vector in the traditional MLM approach.

Our team was unable to reproduce or make use of the novel pre-trained MLM approach adopted by VoBERT due to computational constraints given the heavier requirements from per-element masking method requiring $O(n^2)$ complexity (against the original $O(n)$ using ratio masking). However, we noted the problems with out-of-distribution tokens raised by VoBERT and redacted sequences containing these token in doing prediction and computing results for LogBERT. Further details for this can be found in the Section 3.

2.1.2 DEALING WITH LIMITED EVALUATION USING REAL-WORLD DATASETS AND INTRODUCING LOG INSTABILITY

A second problem identified about LogBERT is that its performance was evaluated based on a very narrow scope of evaluation datasets, and did not reflect true performance. To better benchmark performance, the authors of VoBERT introduced a proprietary real-world dataset from ING bank that consists of 399,000 logs of which 2% are marked as anomalous. The authors also included evaluation against varying levels of introduced log instability to gauge the robustness of performance in a simulation of real-world scenarios.

Our team was unable to procure a similar real-world dataset nor to evaluate an additional dataset, but it was useful to note this discrepancy with regards to LogBERT’s reported performance and how it would perform using more unstable log data. Our team dealt with this by removing the convenient heuristic that classifies all sequences with unseen tokens as anomalous, which was a large factor in contributing to LogBERT’s reported good performance. This should provide more realistic performance and hence deal partially with the identified problem here. Further details for this can be found in Section 3.

3 DATA

3.1 DATASETS USED

Training and evaluation was conducted using 3 of the most frequently used datasets in the problem of log sequence anomaly detection.

3.1.1 HDFS

This refers to the Hadoop Distributed File System (HDFS) dataset. This is a collection of logs obtained from a high performance computing cluster of 203 nodes which executed a set of benchmarking jobs running MapReduce. The full dataset consists of 24 million logs gathered in a span of 2 days. A subset of 11 million logs from this dataset was used for this project, with approximately 10% of these logs marked as anomalous. Sequences are separated by session ID as present in the log messages - this is distinct from the other two datasets which generate sequences using a sliding window. Sequences are labeled manually, with anomalous samples mainly relating to performance problems like write exceptions.

3.1.2 BGL

This refers to the BlueGene/L Supercomputer System (BGL) dataset, which consists of approximately 5 million logs gathered in a span of 200 days from a BGL supercomputer, with approximately 10% anomalous samples. These logs are manually labeled in conjunction with the use of a severity field (classifies logs into several categories), with anomalous samples relating to both hardware and software faults. Sequences are generated using a sliding window of 5 minutes - this is similar to the TBird dataset. For both BGL and TBird datasets, sequences had to be trimmed to fit within the BERT architecture max token limit of 512 per sequence.

3.1.3 TBIRD

This refers to the the Thunderbird (TBird) dataset which is collected from a supercomputer system. The full dataset consists of approximately 200 million logs, and is the largest of the 3 datasets used. A subset of 2 million logs was used due to computational constraints, with approximately 5% anomalous samples. The log data is made up of manually labeled logs, with messages having alert category tags which informed their labels.

3.2 PREPROCESSING STEPS

Separate preprocessing scripts were used for each of these datasets due to differences in their log structure, however the achieved preprocessing is largely the same.

3.2.1 ASSIGNING INTEGER LABELS FOR EACH TOKEN

EventIDs were sorted in descending order by number of occurrences. Each of these EventIDs represent a single token, and was mapped to unique integers following a strictly increasing integer sequence. These integers would serve as the vocabulary, with the maximum integer serving as the vocabulary size.

3.2.2 TRUNCATION OF SEQUENCES

For both BGL and TBird datasets, there was a need to truncate sequences with lengths above 510 tokens. This was to facilitate processing with BERT, which has a maximum token limit of 512 (but which includes 2 special tokens - [CLS] and [SEP]). This form of preprocessing has the drawback of potentially missing out the context from the truncated sections of sequences - this could be studied in future work to document its impact.

For the HDFS dataset, sequences did not exceed 512 tokens in length, and there was no need for truncation.

3.2.3 REMOVAL OF OUT-OF-DISTRIBUTION TOKENS

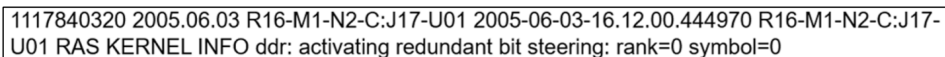
As noted in Section 2 by the authors behind VoBERT, LogBERT reported results which included a simple heuristic that treated sequences with any unseen tokens as anomalous. This may give a false picture of the efficacy of LogBERT, especially when it comes to new log data that is normal but unseen, and hence would not perform well with unstable log data.

To eliminate such perturbing factors and to provide a more balanced view of the results, we ran a script to obtain the vocabulary present in the train dataset, and removed any sequences in the test dataset that included one or more tokens not within the train vocabulary. Claimed LogBERT results were rerun with this pre-processing step. Results are reported in Section 5.

4 METHODS

4.1 MASKED LANGUAGE MODEL

Our model is built on the BERT Masked Language Model framework (Devlin et al., 2019), tailored for self-supervised learning on system log text. During training, 15% of all sequenced tokens are masked at random. However, selected tokens are only replaced with the reserved [MASK] token eighty percent of the time. A random token is selected from the vocabulary corpus and used as replacement ten percent of the time. Additionally, the selected masked token is left alone ten percent of the time. This prevents model confusion during inference when the [MASK] token is excluded from sequences. Once the sequences are masked and padded to uniform length, the transformer uses cross-entropy loss and Adam to learn the system log norms.



```
1117840320 2005.06.03 R16-M1-N2-C:J17-U01 2005-06-03-16.12.00.444970 R16-M1-N2-C:J17-U01 RAS KERNEL INFO ddr: activating redundant bit steering: rank=0 symbol=0
```

Figure 1: System log in the BGL dataset. The red underscored text indicates the detailed computational event (Guo et al., 2021).

For example, given the log message in Figure 1 and the word `redundant` selected to be masked:

- 80% of occurrences: replace `redundant` with the [MASK]

```
ddr: activating redundant bit steering: rank=0
symbol=0 →
ddr: activating [MASK] bit steering: rank=0
symbol=0
```
- 10% of occurrences: replace `redundant` with another word in the corpus

```
ddr: activating redundant bit steering: rank=0
symbol=0 →
```

```
ddr: activating symbol bit steering: rank=0
symbol=0
```

- 10% of occurrences: no change

```
ddr: activating redundant bit steering: rank=0
symbol=0 →
ddr: activating redundant bit steering: rank=0
symbol=0
```

4.2 MASKED KEY LOG PREDICTION

LogBERT generated a methodology called *Masked Key Log Prediction* (Guo et al., 2021), which was used during inference to generate anomaly scores. In this approach, 65% of sequence tokens are masked at random to simulate missing information. The trained model then predicts the top- g candidate tokens for each of the masked items using their token probability distribution. If the true token absent from the top- g candidates, that item is treated as anomalous. g is an integer calculated using a validation set during training. Figure 2 illustrates LogBERT’s inference process. If $X\%$ of the masked tokens are classified as anomalous, the log message is flagged as anomalous. As outlined in Section 3, and noting that the LogBERT team relied on a masked anomaly threshold of 0%, we propose an enhanced methodology.

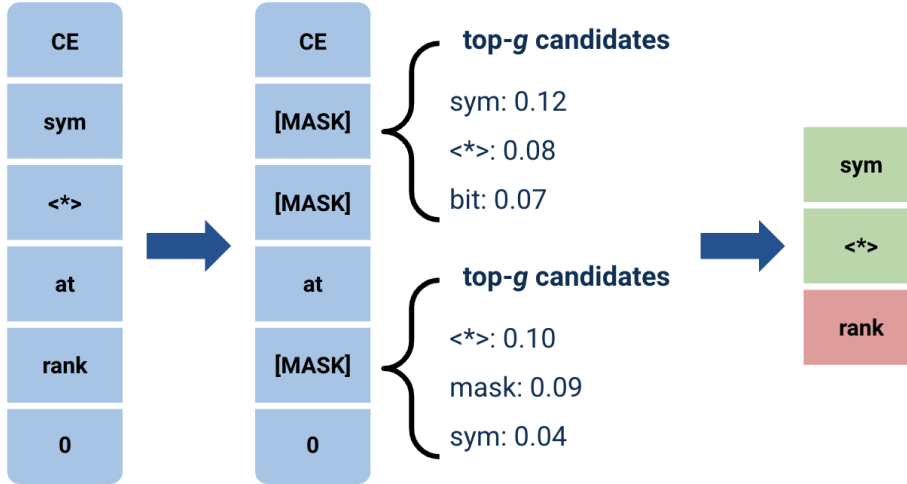


Figure 2: LogBERT’s top- g selection process.

Instead of primarily focusing on the detection of out-of-distribution (OOD) tokens that were unseen during training, our approach evaluates the entire sequence to guide anomaly classification. For each token in a sequence, we computed its assigned probability and applied a novel formula to mitigate the influence of excessively anomalous or OOD tokens that could disproportionately affect the classification.

The geometric mean is a well-established method for averaging probabilities, so we first calculated the geometric mean of each token’s probability in the sequence to obtain a sequence probability score. To convert this into an anomaly score, we subtracted the sequence probability from 1. However, through experimentation, we found that super anomalous tokens disproportionately influenced the geometric mean, skewing the results. To address this, we transformed each token’s probability into an anomaly score by subtracting it from 1, and then recalculated the geometric mean across the sequence. Since values closer to 1 have significantly less influence than values closer to 0 in geometric mean calculations, this adjustment helps reduce the impact of extreme outliers.

Finally, to balance the contributions of methods, we computed the harmonic mean of the two scores. This approach effectively reduced the likelihood of legitimate system logs, which may contain min-

imal OOD tokens, being incorrectly flagged as anomalous. Equation 1 shows the formula used to generate our scores, and Figure 3 visualizes the inference architecture.

$$\text{Sequence Anomaly Score} = \frac{2 \cdot \left(1 - \prod_{i=1}^N p_i\right)^{\frac{1}{N}} \cdot \left(\prod_{i=1}^N (1 - p_i)\right)^{\frac{1}{N}}}{\left(1 - \prod_{i=1}^N p_i\right)^{\frac{1}{N}} + \left(\prod_{i=1}^N (1 - p_i)\right)^{\frac{1}{N}}} \quad (1)$$

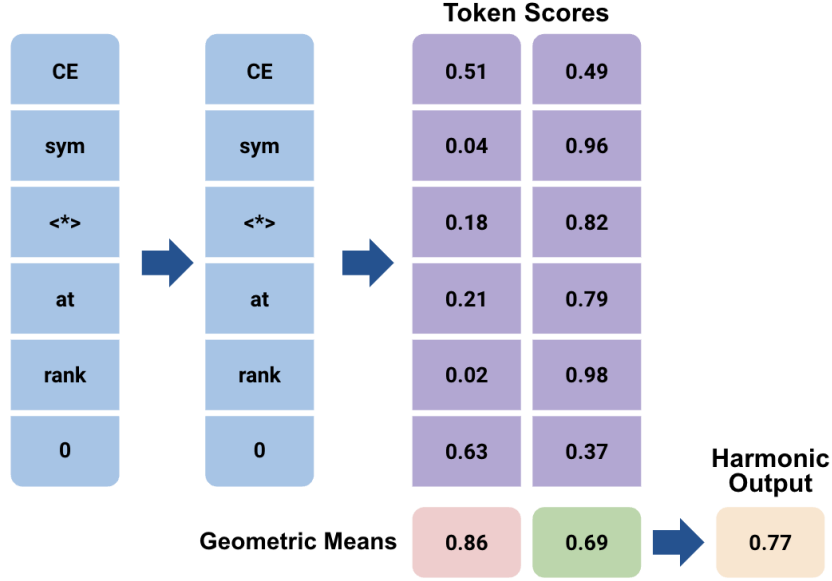


Figure 3: Our modified anomaly score generation process.

4.3 THRESHOLD GENERATION

To empower cybersecurity analysts with adaptable tools, a tunable threshold was implemented. This mechanism allows adjustment of the model’s sensitivity, aligning detection with dynamic network environments. The threshold can be customized based on the operational priorities and network behavior. When defining a threshold, it is important to consider both network characteristics and analyst priorities. In high-volume networks that generate millions of events daily, excessive anomalous alerts can overwhelm analysts, making it impractical to review all flagged logs. In such cases, analysts prioritize precision over recall, as their limited resources are better spent investigating a smaller number of highly accurate alerts rather than sifting through thousands of potentially benign anomalies.

To generate a suitable threshold, we reserved a portion of the training data specifically for threshold calibration. Analysts are then prompted to specify the percentage of normal traffic they are willing to classify as anomalous. The trained model is subsequently evaluated on the validation set, where anomaly scores are calculated for each event. Based on the analyst’s specified tolerance level, we determine the threshold that segments the events accordingly. For example, if an analyst is willing to accept one anomaly per 10,000 normal events, and the validation set contains 20,000 events, the anomaly score corresponding to the second most anomalous event is selected as the threshold. This approach empowers analysts to influence model behavior directly, providing intuitive control without requiring specialized machine learning expertise.

5 EXPERIMENTS AND RESULTS

5.1 EXPERIMENTS SETUP

As mentioned in Section 3.2.3, we chose to remove all sequences that contained out-of-distribution (OOD) tokens from each of the three test datasets, to eliminate anomalous sequences that came about just from containing OOD tokens.

As a gauge of the token distributions between the train and test datasets, we plotted histograms as can be seen in Figure 4 below for the HDFS dataset. From the figure, it can be seen that the vocabulary present in the train dataset is only of size 15, whereas those in the normal and abnormal test dataset has size 47. While largest frequencies of occurrences are still pertaining to the vocabulary seen within the train dataset, there is still roughly 2.1 times the number of OOD tokens as there are tokens within distribution. With the heuristic labeling any sequence with just one OOD token as anomalous, it was important to do the removal of OOD tokens to give a fair result.

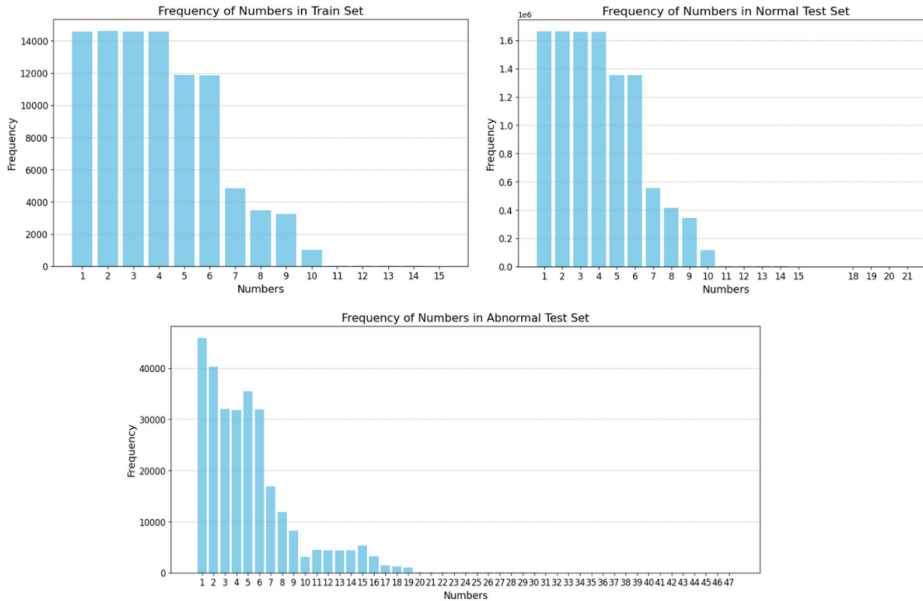


Figure 4: Distribution of tokens for train and test splits of the HDFS dataset

Table 1 shows the reduction in number of sequences for the test datasets before and after eliminating the sequences according to the criteria discussed in Section 3. The largest reduction came from abnormal sequences, which is in line with our expectations. Of particular note would be the HDFS dataset, which experienced a 36.2% decline in sequences and highlighted how many anomalous sequences were classified as such simply because of OOD tokens.

	BGL		HDFS		TBird	
	Anomalous	Normal	Anomalous	Normal	Anomalous	Normal
Before	3,018	20,579	16,838	553,368	104,992	7,739
After	2,927	20,578	10,734	553,327	104,988	7,739

Table 1: Number of sequences in test datasets after removing Out-of-distribution (OOD) tokens

5.2 COMPARISON AGAINST LOGBERT

We evaluated four approaches against the LogBERT framework using our modified dataset:

1. Geometric mean over token probabilities
2. Geometric mean over token anomaly scores, calculated as $(1 - \text{probabilities})$
3. Harmonic mean between Approach 1 and Approach 2
4. Approach 3 applied only to the top 40% most anomalous tokens

These methods were designed to explore variations in scoring strategies and their impact on anomaly detection performance. By comparing their effectiveness, we aim to highlight the robustness of different aggregation techniques in addressing the limitations of LogBERT (LB).

	BGL			HDFS			TBird		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
LB Claimed	89.40	92.32	90.83	87.02	78.10	82.32	96.75	96.52	96.64
LB Actual	41.49	70.90	52.34	92.77	36.88	52.77	86.32	99.19	92.31
Experiment 1	97.63	54.97	69.28	54.13	45.85	49.65	97.15	71.70	82.51
Experiment 2	96.62	49.91	65.83	85.15	46.58	60.21	58.62	3.40	6.43
Experiment 3	91.22	52.51	66.65	93.10	45.18	60.83	63.41	5.20	9.61
Experiment 4	92.68	53.23	67.62	25.88	24.60	25.22	89.08	15.50	26.41

Table 2: Validation Results

Experiment 1 performs best on both the BGL and Thunderbird datasets, while Experiment 3 performs best on HDFS. This likely means that BGL and Thunderbird datasets have a few tokens which are strong indicators of anomalous activity in anomalous logs and standardly distributed tokens in normal log messages. HDFS likely is more complex and relies on the context of the entire log message to reveal anomalies instead of a few keywords.

We prioritize precision as it aligns with the performance metrics most valued by cybersecurity analysts. Precision reflects the model’s ability to minimize false positives, which can otherwise trigger unnecessary reactionary measures for non-existent threats, wasting resources and time. However, recall is also critical in cybersecurity because it measures the proportion of missed anomalous events. An inference cycle with near-zero recall while an attacker is present indicates a catastrophic failure, as the model would entirely overlook attack-related network events. Such oversights leave systems vulnerable, defeating the model’s purpose in threat detection.

Our models achieve a strong balance by maintaining reasonable recall while outperforming LogBERT in precision across all datasets. This makes our approach more practical for deployment, offering analysts a reliable tool that reduces false positives while retaining adequate sensitivity to actual threats.

5.3 ABLATION STUDIES

We wanted to test analyst acceptance rates for our anomaly score threshold, so we ran ablation studies on the data using 2 fixed thresholds (1% and 0.1%) and a variable optimal threshold. It’s clear that the thresholds created by these ratios drastically increase precision and decrease recall. Analysts who value precision would appreciate this parameter to assist their workload and network needs. Tables 3, 4, 5, and 6 use the analyst acceptance ratios to drive classification. In many cases, the optimal rate is below 1% acceptance; however, analyst preference may drive that ratio up with understood tradeoffs.

	BGL			HDFS			TBird		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
1/100	96.10	51.38	66.96	63.21	0.67	1.33	97.15	71.70	82.51
1/1000	99.36	42.53	59.57	90.41	0.66	1.31	99.82	54.20	70.25
Optimal	97.63	54.97	69.28	54.13	45.85	49.65	81.47	98.90	89.34

Table 3: Experiment 1 Ablation Studies

	BGL			HDFS			TBird		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
1/100	96.41	48.65	64.67	95.28	30.25	45.92	58.62	3.40	6.43
1/1000	98.37	18.52	31.17	99.73	27.73	43.39	60	0.30	0.60
Optimal	96.62	49.91	65.83	85.15	46.58	60.21	76.32	97.00	85.42

Table 4: Experiment 2 Ablation Studies

	BGL			HDFS			TBird		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
1/100	96.69	49.85	65.78	97.81	29.03	44.77	63.41	5.20	9.61
1/1000	99.40	17.01	29.05	99.46	27.43	42.99	55.56	0.50	0.99
Optimal	91.22	52.51	66.65	93.10	45.18	60.83	80.46	24.70	37.80

Table 5: Experiment 3 Ablation Studies

	BGL			HDFS			TBird		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
1/100	96.63	49.91	65.83	96.31	2.94	5.71	89.08	15.50	26.41
1/1000	97.92	4.82	9.18	99.66	2.92	5.88	95.00	1.90	3.73
Optimal	92.68	53.23	67.62	25.88	24.60	25.22	86.63	52.50	65.38

Table 6: Experiment 4 Ablation Studies

6 KEY TAKEAWAYS AND CONCLUSION

In this paper, we experimented with and evaluated different methods of calculation of sequence scores as an effort to tune the performance of LogBERT. Tuning of the threshold hyperparameter for classification of anomalies was also another effort to optimize anomaly detection. A last optimization was to remove OOD tokens from the test split of each dataset. As a result of these optimizations, we did manage to obtain better performance than the original LogBERT on the same three datasets.

One key takeaway from this project is showing the effect of the simple heuristic used by LogBERT to classify all sequences with one or more OOD tokens as anomalous, and how it contributed to LogBERT’s perceived better performance. This emphasized the importance of interpreting results and how these results were obtained and not just taking them at face value.

The key takeaways from our project highlight the significant advancements made to the LogBERT framework. By introducing the aforementioned new scoring methods, we address limitations to the original LogBERT framework and improve the model’s robustness - results before and after removal of OOD tokens were not affected much and consistently outperformed LogBERT on two out of three evaluated datasets. The inclusion of a tunable threshold hyperparameter (tunable by analysts using our framework) in detecting anomalies changes the LogBERT framework from being more heuristic and pattern-based into a more reliable and robust framework.

Future work might include expanding on this updated architecture by leveraging VoBERT’s ability to handle OOD tokens to improve robustness and performance. Training and evaluation could also be conducted on full-sized datasets with larger computational capacity, as current work was limited by this and used only a sampled subset of data.

To deal with critiques regarding poor performance with log instability, more robust evaluation could be conducted with the use of real-world datasets or even to evaluate results by intentionally including varying levels of log instability and measuring performance.

7 CONTRIBUTIONS

As seen in Table 7, every member contributed to the project. The three datasets used in the project were split among the three members for editing code, training models, and running test/validation

code. Since each dataset had its own format, each member had to make adaptations to the code to match the assigned dataset.

Contributor	Contribution
Garrett Devaney	Led the project and developed the base code for the modified LogBERT used in the project. Generated results for the HDFS dataset.
Jun Liang Ho	Managed project progress and standardized project code. Generated results for the BGL dataset.
Kevin Guernsey	Oversaw project operations and progress. Generated results for the Thunderbird dataset.
All	Worked on presentation and report. Communicated with and aided others with troubleshooting issues.

Table 7: Contributions of team members

REFERENCES

- Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. *CoRR*, abs/1803.04967, 2018. URL <http://arxiv.org/abs/1803.04967>.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL <https://arxiv.org/abs/1810.04805>.
- Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. 2017. URL <https://users.cs.utah.edu/~lifeifei/papers/deeplog.pdf>.
- Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert, 2021. URL <https://arxiv.org/abs/2103.04475>.
- Daan H. Vobert: Vocabulary-free bert for log anomaly detection, 2023. URL <https://github.com/daanh99/VoBERT>.
- Daan Hofman. VoBERT: Unstable log sequence anomaly detection. Master’s thesis, Delft University of Technology, 2023. URL <https://repository.tudelft.nl/record/uuid:f8593fb8-aa20-4caa-b12e-2521416c98a1>.
- William Hurst, Madjid Merabti, and Paul Fergus. A survey of critical infrastructure security. *Critical Infrastructure Protection VIII*, pp. 127–137, 2014.
- Bart Lenaerts-Bergmans. What are living off the land (lotl) attacks. *CrowdStrike*, 2023. URL <https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/living-off-the-land-attack/>.