Can Language Models Help in System Security? Investigating Log Anomaly Detection using BERT

Crispin Almodovar[1] **Fariza Sabrina**[1] **Sarvnaz Karimi**[2] **Salahuddin Azad**[1]
[1]Central Queensland University, Australia
[2]CSIRO Data61, Sydney, Australia
crispin.almodovar@cqumail.com
{f.sabrina,s.azad}@cqu.edu.au
{sarvnaz.karimi}@csiro.au

Abstract

The log files generated by networked computer systems contain valuable information that can be used to monitor system security and stability. Transformer-based natural language processing methods have proven effective in detecting anomalous activities from system logs. The current approaches, however, have limited practical application because they rely on log templates which cannot handle variability in log content, or they require supervised training to be effective. We propose a novel log anomaly detection approach named LogFiT. It utilises a pretrained BERT-based language model and fine-tunes it towards learning the linguistic structure of system logs. The LogFiT model is trained in a self-supervised manner using normal log data only. Using masked token prediction and centroid distance minimisation as training objectives, the LogFiT model learns to recognise the linguistic patterns associated with the normal log data. During inference, a discriminator function uses the LogFiT model's top-k token prediction accuracy and computed centroid distance to determine if the input is normal or anomaly. Our experiments on three different datasets show that LogFiT is effective.

1 Introduction

Cybercrime costs businesses billions of dollars annually (RiskIQ, 2019; Australia Department of Home Affairs, 2020; International Business Machines, 2022). Log anomaly detection helps to protect businesses' digital infrastructure from cyber-attacks by providing the ability to detect abnormal activities, such as network intrusions, from large volumes of event logs generated by networked computer systems.

Recently, approaches based on Deep Learning and Natural Language Processing (NLP) have been applied to address the log anomaly detection problem. A review of the literature indicates that Long Short-Term Memory (LSTM), represented by the DeepLog model (Du et al., 2017), and Transformers, represented by the LogBERT model (Guo et al., 2021), are the deep learning architectures used in the state of the art research in this domain. A practical consideration in log anomaly detection using deep learning is the availability of labeled data to be used in training predictive models. Because of the high cost of preparing labeled data, classification-based approaches such as LogSy (Nedelkoski et al., 2020) are of limited value in production settings. Thus, a majority of log anomaly detection approaches focus on the zero-positive training scenario, in which predictive models are trained in a self-supervised manner using normal log data only (Le and Zhang). Further, Yuan et al. (2021) identifies two general categories of self-supervised models for anomaly detection: (1) forecasting-based, which attempts to predict the next log entry given previous log entries; and (2) reconstruction-based, which recomposes log sequences that have been intentionally corrupted. The DeepLog model adopts the forecasting-based approach, while the LogBERT model uses the reconstruction-based approach.

We focus on log data that consists of sequences of log sentences. A key factor affecting the effectiveness of log anomaly detection models is how well it encodes representations of sequences of log sentences, especially as the content of the log sentences changes over time (Hendrycks et al., 2020; Ott et al., 2021). A common approach is to encode log sentences by first converting them to log templates (Du et al., 2017; Guo et al., 2021). However, this method is shown to negatively affect model effectiveness due to sub-optimal vector representation of the log sequences, and its inability to handle unexpected variability in the content of log sentences over time (Nedelkoski et al., 2020; Le and Zhang, 2021; Wittkopp et al., 2021).

To address the limitations of current approaches, we make the following contributions:

- An anomaly detection model named LogFiT, which uses a fine-tuned pre-trained Bidirectional Encoder Representations from Transformers (BERT)-based Language Model (LM) to learn the linguistic structure and sequential patterns of normal log data. The fine-tuning is done through transfer learning, where a base LM, pre-trained on a large collection of text corpora, is retrained on the normal log data. The use of a pre-trained LM allows LogFiT to generate representations for any sequence of log sentences. Therefore, LogFiT is robust to future changes in the syntactic structure of log sentences.

- A framework and workflow for implementing domain specific LogFiT anomaly detection models. The framework adopts a self-supervised, transfer learning approach based on the Masked Language Modeling (MLM) objective. The model is trained to minimise the cross-entropy loss combined with centroid distance loss. During inference, the model's top-k accuracy and centroid distance are compared against some threshold values to determine whether a log sequence is normal or anomalous. Furthermore, the framework incorporates techniques that are known to speed up model training: discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing.

2 Related Work

System log data consists of log sentences representing events that occur within computer systems. Several log anomaly detection methods use log parsing as its initial step, in which the log data is converted into a standardised format called "log templates" (Chen et al., 2021; Zhao et al., 2021; He et al., 2021), such that every log sentence can be mapped to a specific log template. The list of log templates thus forms the vocabulary of the model, instead of words or tokens as is typical in NLP. An example of system log data as it is converted to log templates is shown in Figure 1.

The DeepLog and LogBERT approaches are illustrated in Figure 2. In both of these approaches, the input log data is pre-processed to convert them into log sentence templates, which form the *vocabulary* of these models. The input to the model is a sequence of log keys, which are indexes used to look up the corresponding log sentence template from the vocabulary. In the case of DeepLog, the last log key is removed from the input, and the model is trained to predict the missing log key given the previous log keys. In the case of LogBERT, some percentage of log keys are masked in the input, and the model is trained to predict what the masked log keys are.

Some studies (Nedelkoski et al., 2020; Le and Zhang, 2021; Wittkopp et al., 2021) suggest that log templates often result in significant loss of contextual information that is beneficial to a predictive model's performance. The problem with log templates is that it assumes the list of log templates invariant. However, changes in the content of log sentences will naturally happen over time. Thus models that rely on log templates will not be able to map new log sentences to an entry in the list of log templates. Consequently, LogSy (Nedelkoski et al., 2020), Neuralog (Le and Zhang, 2021) and A2Log (Wittkopp et al., 2021) do not use log templates; instead the log data is pre-processed using simple cleanup scripts to remove unnecessary details such as specific IP addresses, file paths, port numbers, and URLs.

Recently, the linguistic capabilities of pretrained LMs such as BERT (Devlin et al., 2019) has been a subject of increasing interest. Several studies have concluded that BERT-based language models learn syntactic and semantic information that can be used to increase the effectiveness of downstream NLP tasks (Jawahar et al., 2020; Lin et al., 2019; Goldberg, 2019; Yenicelik et al., 2020). The LogFiT model therefore leverages a pre-trained BERT LM to accurately "understand" the linguistic structure and sequential properties of normal system logs.

3 Method

LogFiT is trained on normal log data which is first transformed into semantic vectors before being passed to the anomaly detection model. In contrast to DeepLog and LogBERT, the LogFiT model does not require the extraction of log templates during the pre-processing step. By inheriting from a BERT-based language model, LogFiT has the capabilities of an auto-encoder that can reconstruct log data that have been intentionally corrupted via masking. Specifically, LogFiT uses the Longformer (Beltagy et al., 2020) variant of the BERT family of models. The Longformer model allows LogFiT to handle log paragraphs that contain up to 4096 tokens, much higher than BERT's

```
081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating
081109 203807 222 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-6952295868487656571 terminating
081109 204005 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.73.220:50010 is added to blk_7128370237687728475 size 6710886
081109 204015 308 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_8229193803249955061 terminating
081109 204106 329 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_-6670958622368987959 terminating
081109 204132 26 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.43.115:50010 is added to blk_3050920587428079149 size 6710886
081109 204324 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.203.80:50010 is added to blk_7888946331804732825 size 6710886
```

```
PacketResponder <*> for block blk_<*> terminating
PacketResponder <*> for block blk_<*> terminating
BLOCK* NameSystem.addStoredBlock: blockMap updated: <*>:<*> is added to blk_<*> size <*>
PacketResponder <*> for block blk_<*> terminating
PacketResponder <*> for block blk_<*> terminating
BLOCK* NameSystem.addStoredBlock: blockMap updated: <*>:<*> is added to blk_<*> size <*>
BLOCK* NameSystem.addStoredBlock: blockMap updated: <*>:<*> is added to blk_<*> size <*>
```

Figure 1: Sample system log data converted to log templates, from the HDFS dataset.

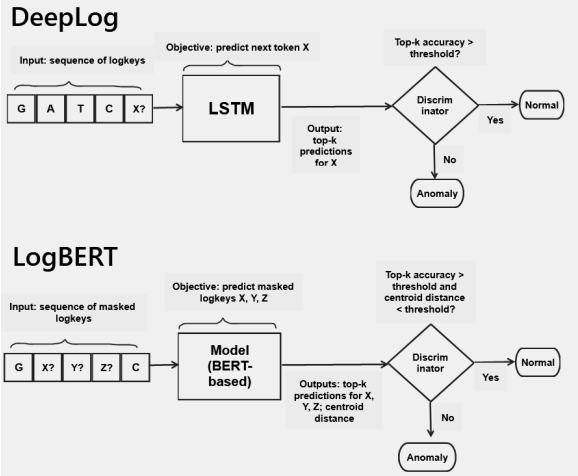

Figure 2: The DeepLog and LogBERT log anomaly detection approaches.

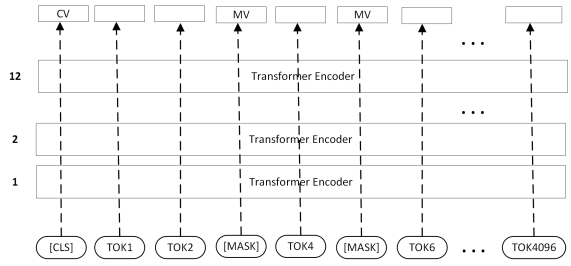

Figure 3: LogFiT Transformer layers.

limit of 512 tokens.

The input to the LogFiT model is a log paragraph consisting of individual log sentences joined together with a line separator character. LogFiT supports up to 4096 tokens, which follows from the limit of the Longformer model. It is noted that the "tokens" in LogFiT differs from the "tokens" in DeepLog and LogBERT - in LogFiT the tokens are words or sub-words, while in DeepLog and LogBERT the tokens are log sentence templates. The output of the final layer of LogFiT are 768-dimension vectors that are the learned contextual representations of the input tokens. Of interest are the **[CLS]** token vector **CV** and the masked token prediction vector **MV**. By convention, in BERT-based models, the [CLS] token is the first token in the input sequence, and is typically used for classification tasks. The **CV** vector corresponds to the representation of the entire log paragraph, while the masked token prediction vectors **MV** correspond to the model's predictions for the masked tokens. At the beginning of each training epoch, the **CV** vector is used to compute the centroid of all normal training data. During training proper, the **CV** vector of each log paragraph is used to compute its distance from the current centroid. In contrast, the **MV** vector is used to compute the masked token prediction loss (cross-entropy loss) following the BERT masked language modeling algorithm. An important detail related to LogFiT's use of Longformer is the use of global attention for the **[CLS]** token and all line separator characters only, while all other tokens are limited to local attention with a window size of 16 to 32 - this value is based on findings discussed in Dai et al. (2022).

3.1 Training Objectives

The LogFiT model is trained in a self-supervised manner using two training objectives:

Objective 1: Masked Language Modeling. This training objective is a variation of the training objective used to pre-train BERT-based language models (Devlin et al., 2019). In this training objective, the model randomly masks up to 75% of the sentences that comprise the log paragraph. The tokens of the log sentences are then masked according to the BERT masking algorithm (80% masked, 10% replaced with a random token, 10% left unchanged). Subsequently the model predicts what

the masked tokens are. The intuition behind this training objective is that, for the model to accurately predict the masked tokens, it must learn the contextual relationships of the tokens and the sentences that make up the training data. Thus, the model is thought to gain an understanding of the syntax and semantics of the language domain of normal system logs. Further, because the model is trained on the normal log data, it is expected that the model will be able to learn patterns associated with the normal data and thus distinguish it when the normal data is presented with anomalous data. The masked language modeling training objective is implemented by minimising the cross-entropy loss between the model's predictions of the masked tokens and the correct tokens. Aggregating the cross-entropy loss across all samples in a mini-batch produces the MLM loss and is described by equation 1.

$$Loss_{mlm} = -\frac{1}{b}\sum_{j=1}^{b}\sum_{i=1}^{m} y^{j}_{mask_i} \log(p^{j}_{mask_i}) \quad (1)$$

where b is the mini-batch size, m is the count of masked tokens, y is the true value, and p is the probability of the predicted value.

Objective 2: Centroid Distance Minimisation. This training objective is motivated by the observation that normal log data samples tend to cluster close to each other (Ruff et al., 2019; Nedelkoski et al., 2020; Guo et al., 2021). Therefore, as an additional training objective, the distance of each vectorised log paragraph from the computed centroid of all normal log paragraphs is minimised. The centroid is computed at the start of each epoch to leverage improvements to the model weights from the previous epoch. It has been demonstrated in the works of (Ruff et al., 2019), (Nedelkoski et al., 2020) and (Guo et al., 2021) that the performance of self-supervised log anomaly detection models improves with the addition of this training objective. The centroid distance loss is the mean squared error between the **CV** vector (vectorised log paragraph) and the best centroid computed from the previous training epochs. The centroid is the average of all **CV** vectors of all normal training samples. Additionally during the centroid distance minimisation objective, the q-quantile centroid distance (where q is set to between 0.65 to 0.9 in the experiments) is determined - this distance is then considered as the radius or the hypersphere that encloses all normal samples and is used as threshold value during inference. Equation 2 shows the formula for computing the centroid distance loss for a mini-batch of log data.

$$Loss_{cdist} = \frac{1}{b}\sum_{j=1}^{b}(CV_j - centroid)^2. \quad (2)$$

LogFiT's loss function, shown in Equation 3 is a combination of the cross-entropy loss computed from the masked language modeling objective and the centroid distance loss computed from the centroid distance minimisation objective. The contribution of the centroid distance loss to the final loss value is weighed via hyper parameter cw, which is set to 0.25 in the experiments. The resulting composite loss is then minimised using the Adam optimiser, using hyper parameters recommended by the FastAI framework: momentum $= 0.9$, sqr_momentum $= 0.99$, $\epsilon = 1e-5$, weight decay $= 0.01$.

$$Loss = Loss_{mlm} + cw * Loss_{cdist}. \quad (3)$$

3.2 Anomaly Detection

The trained LogFiT model can be used to detect anomalous log data because it is trained to recognise normal data. During inference, the input data (in the form of log paragraphs) goes through the same tokenisation, vectorisation, masking, and prediction steps as at training time. Taking inspiration from both DeepLog and LogBERT approaches, LogFiT's anomaly score is composed of two separate scores: the top-k accuracy (with k=5..12) which represents how well LogFiT reconstructs the masked sentences in the input data; and the centroid distance of the CV vector computed by LogFiT for the input data (the centroid is determined during training, based on the average of all CV vectors of all normal log samples). If either of these two scores passes some threshold value then the input data is considered an anomaly. Specifically, if the top k accuracy falls below some threshold (set to between 0.65 and 0.99 in the experiments) or the centroid distance of the CV vector exceeds some multiple of the normal centroid distance (set to between 1.2 to 1.9) computed during training, the input log paragraph is considered an anomaly, otherwise it is considered normal.

Dataset	Avg #W	Avg # S	Unique W
HDFS	176.04	18.63	146
BGL	128.66	15.73	6,046
Thunderbird	1445.70	126.63	15,557

Table 1: Average counts of words (W) and sentences (S) per log paragraph for different datasets.

4 Datasets and Experimental Setup

We use three public datasets: HDFS (Xu et al., 2010), BGL (Oliner and Stearley, 2007) and Thunderbird (Oliner and Stearley, 2007). These datasets are selected because they are used by the baseline models. Some statistics on these datasets are shown in Table 1. There is a noticeable difference in terms of diversity of vocabulary used in these datasets, with HDFS having a very limited vocabulary of only 146 unique words, as opposed to Thunderbird which is more diverse with its vocabulary close to the size of what an adult native English speaker would have, which is approximately 15,000 to 30,000 (Brysbaert et al., 2016).

The HDFS dataset consists of log entries (sentences) that are grouped into sessions, identified by the block ID field. In contrast the BGL and Thunderbird datasets do not have session identifiers, so a time-based grouping of log sentences is used. During deployment LogFiT is intended to be used in an online mode (as opposed to batch) therefore for datasets where the grouping of log sentences is based on time window, the chosen interval is 30 seconds so that a system utilising LogFiT can provide timely feedback to system operators. Each group of log sentences (i.e., a log paragraph) becomes a single sample that is then fed in batches to the models during training, tuning and evaluation.

The datasets are split into training/validation, tuning, and evaluation sets. The training/validation set is created from 6,000 normal samples for training, and 5,000 normal plus 1,000 anomaly samples for parameter tuning. The evaluation set is created from 5,000 normal plus 1,000 anomaly samples. No random shuffling is performed on the datasets - the chronological order of the logs is used; this is to prevent models from "peeking into the future" during training. The evaluation set consists of log data that appear after (in chronological order) the train/validation set.

Implementation Details. LogFiT is implemented using Pytorch (Paszke et al., 2019), FastAI (Howard and Gugger, 2020), and HuggingFace (Wolf et al., 2020).

Evaluation Metrics. To measure the effectiveness of the models, the following metrics are used:

- *Precision* (P) is percentage of correctly detected anomaly samples (TP), among all the anomalies detected by the model as *P = TP / (TP+FP)*.
- *Recall* (R) is percentage of log samples that the model correctly identified as anomaly, over all real anomalies, as *R = TP / (TP+FN)*.
- *F1 Score* ($F1$) is the harmonic mean of the Precision and Recall, as *F1 = 2 * (P*R)/(P+R)*.
- *Specificity* (S) is the percentage of log samples that the model correctly detected as normal, over all real normal samples, as *S = TN/(TN+FP)*.

In practical deployment scenarios a model with high specificity is more valuable, in that it minimises occurrences of false positives or false alarms. A model with high Specificity will accurately identify normal samples, thus if a sample is detected as an anomaly it is highly likely that the sample is really an anomaly. Furthermore, Le and Zhang found that Specificity helps mitigating the effect of imbalanced class distribution.

5 Results and Discussion

Log Anomaly Detection Performance. Table 2 shows the result of running anomaly detection inference using LogFiT, as compared to the metrics obtained when running the publicly available implementations of DeepLog and LogBERT on the same data. The LogFiT model is used to detect anomalous log paragraphs from the HDFS, BGL and Thunderbird datasets. The results show that LogFiT's F1-scores outperform DeepLog and LogBERT on the HDFS and BGL datasets, and comparable to LogBERT on the Thunderbird dataset.

Effect of delaying centroid distance computation. Table 3 shows the effect of a warm-up period of 5 epochs before computing the centroid and the centroid distance loss. LogFiT is trained using three stages of gradual unfreezing (Howard and Ruder, 2018), with five epochs for each stage. The result indicates that a warm-up period negatively affects the model's effectiveness on the HDFS dataset. This could be because LogFiT relies on a pre-trained Longformer which is already

	HDFS				BGL				Thunderbird			
Method	P	R	F1	S	P	R	F1	S	P	R	F1	S
DeepLog	100.0	60.90	75.70	100.0	90.2	70.68	79.25	98.32	65.05	99.4	78.64	89.30
LogBERT	24.02	82.80	37.24	47.62	88.92	88.35	88.63	97.59	91.75	95.7	93.69	98.28
LogFiT (ours)	99.78	90.60	94.97	99.96	98.83	84.70	91.22	99.00	89.90	98.80	94.14	97.78

Table 2: Comparison of anomaly detection effectiveness of different methods in terms of Precision (P), Recall (R), F1 score (F) and Specificity (S) on three log datasets (HDFS, BGL, Thunderbird).

Warm-up	F1	Specificity
No warm-up	94.97	99.96
5-epoch warm-up	87.70	100.0

Table 3: Effect of delaying centroid distance computation on LogFiT/HDFS F1 and specificity.

capable of producing good vector representations of log paragraphs.

Figure 4 shows how the two threshold parameters, top-k token prediction accuracy and centroid distance, contributes to the anomaly decision for the HDFS evaluation set (which consists of 5,000 normal samples and 1,000 anomaly samples). The figure indicates that centroid distance is not an important decision factor for discriminating normal and anomaly HDFS log paragraphs.

Transfer learning. Due to transfer learning, the LogFiT model starts training with an inherited knowledge of the linguistic characteristics of the English language, while neither DeepLog or LogBERT have this benefit. This allows LogFiT training to converge in fewer number of epochs compared to the two baseline models. Furthermore, because LogFiT uses a large pre-trained language model to vectorise log paragraphs, it is more robust to changes in the content of the log data (Nedelkoski et al., 2020; Ott et al., 2021).

Log parsing. Unlike DeepLog, LogBERT and other approaches that depend on a log parsing step, LogFiT works directly with the text data. Figure 5 shows an example of LogFiT's input log paragraph which have been masked according to the BERT masking algorithm, and shows the two criteria (top-k accuracy and centroid distance) used by LogFiT to decide whether the input is normal or an anomaly.

Note that in our initial experiments on the Thunderbird dataset, LogFiT's effectiveness was below that of baselines. This was attributed to the length of the log paragraphs being input to the LogFiT model. After reducing the time window from 60 seconds to 30 seconds, LogFiT's F1 score and specificity were comparable to that of LogBERT. The same reduction in time window was applied to the baselines as well.

Statistical significance. Figure 6 shows the predictions of the LogFiT model compared against the predictions of the LogBERT model on the HDFS dataset, presented in a McNemar contingency table. Applying McNemar's test with continuity correction and a significance level of $\alpha = 0.05$ produces $\chi^2 = 2553.83$ and $P = 0.0$ which confirms that LogFiT performs better than LogBERT.

6 Limitations

Due to the size and computational requirements of the Longformer model, training on log data where the length of a paragraph is longer than 2048 tokens takes a long time to complete. Further, it can be prone to out-of-memory errors even when training on an NVIDIA RTX A6000 with 48 GB of GPU memory. Addressing this limitation will be the subject of a follow up study.

7 Conclusions

Detecting abnormal computer system behavior from the log files that the system generates is an important capability in today's hyper-connected world. Natural language processing techniques and in particular transformer-based models using BERT are investigated for anomaly detection in system logs. We presented a novel log anomaly detection model named LogFiT. The LogFiT model leverages the general knowledge embodied in the pre-trained weights of a BERT-based language model and fine-tuned it to learn the specific linguistic patterns of system logs. LogFiT is trained in a self-supervised manner using only the normal logs and combining two training objectives: Masked token prediction and centroid distance minimisation. It learns to recognise only the linguistic structure of normal system logs and can reconstruct normal log

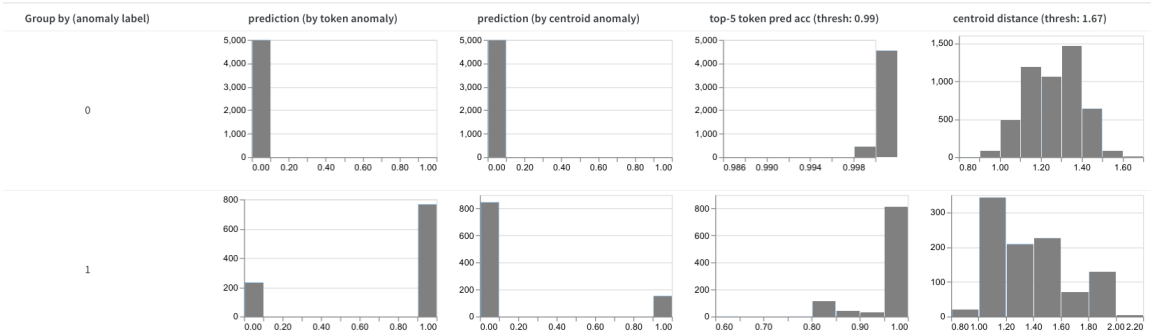

Figure 4: Contribution of each threshold criteria (token prediction accuracy and centroid distance) to the anomaly decision.

log paragraph (masked)	log paragraph (top-1 token predictions)	label	prediction	prediction (top-12 accuracy threshold)	prediction (centroid distance threshold)	top-12 accuracy	centroid distance
<s>synchronized to IPADDR, stratum NUM syn<mask>ized to<mask><mask><mask><mask><mask><ma Administrator: Instrumentation Service EventID: NUM Tempera <mask><mask> from<mask><mask><mask>d<mask> synchron<mask><mask><mask><mask><mask><mask><mask><ma <mask><mask><mask><mask><mask><mask><mask><mask><masl [ib_sm_sweep.c:NUM]: No topology change [ib_sm_sweep.c:NUM]: No configuration change required synchronized to IPADDR, stratum NUM synchronized to IPADDR, stratum NUM Administrator: Instrumentation Service EventID: NUM Tempera <mask><mask><mask> peer<mask><mask>d<mask> <mask><mask><mask><mask><mask><mask> Service<mask>ID<mask> Got trap from peer on fd NUM <mask>chronized<mask><mask><mask><mask><mask><mask><ma <mask><mask><mask><mask><mask><mask> Service<mask><mask> Got trap from peer on fd NUM <mask><mask><mask><mask><mask><mask><mask><mask>.<mas <mask><mask><mask><mask><mask><mask><mask><mask><masl Got trap from peer on fd NUM [<mask><mask><mask>_swe<mask><mask>c<mask><mask><mask> [ib_sm_sweep.c:NUM]: No configuration change required connection<mask><mask><mask>#<mask> Now running on #NUM#'s privileges. Kerberos TGT authentication as user #NUM# accepted for #NU <mask><mask>NUM#<mask><mask><mask><mask><mask><mask> synchronized to IPADDR, stratum NUM <mask>chron<mask><mask> IP<mask><mask><mask><mask><mas synchronized to IPADDR, stratum NUM</s><pad><pad><pad> <pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>	<s>synchronized to IPADDR, stratum NUM synchronized to IPADDR, stratum NUM Administrator: Instrumentation Service EventID: NUM Tempera Got trap from peer on fd NUM synchronized to IPADDR, stratum NUM [ib_sm_sweep.c:NUM]: ********** NEW SWEEP *** [ib_sm_sweep.c:NUM]: No topology change [ib_sm_sweep.c:NUM]: No configuration change required synchronized to IPADDR, stratum NUM synchronized to IPADDR, stratum NUM Administrator: Instrumentation Service EventID: NUM Tempera Got trap from peer on fd NUM Administrator: Instrumentation Service EventID: NUM Tempera Got trap from peer on fd NUM synchronized to IPADDR, stratum NUM Administrator: Instrumentation Service EventID: NUM Tempera Got trap from peer on fd NUM [ib_sm_sweep.c:NUM]: ********** NEW SWEEP *** Administrator: Instrumentation Service EventID: NUM Tempera Got trap from peer on fd NUM [ib_sm_sweep.c:NUM]: No topology change [ib_sm_sweep.c:NUM]: No configuration change required connection from "#NUM#" Now running on #NUM#'s privileges. Kerberos TGT authentication as user #NUM# accepted for #NU User #NUM#, coming from #NUM#, authenticated. synchronized to IPADDR, stratum NUM synchronized to IPADDR, stratum NUM synchronized to IPADDR, stratum NUM</s><pad><pad><pad> <pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>	0	0	0	0	0.99	2.96

Figure 5: An example of LogFiT's input, output prediction, and decision criteria.

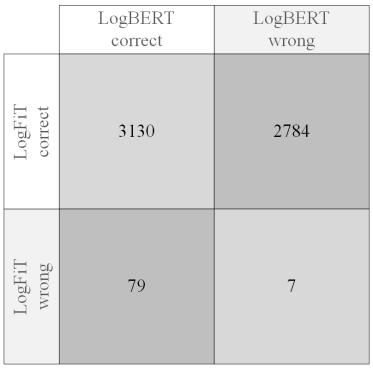

Figure 6: McNemar table comparing LogFiT and LogBERT predictions on the HDFS dataset.

data that have been intentionally corrupted. LogFiT flags as anomalies any log sample that it fails to reconstruct. We showed that our method outperform baseline models on the HDFS and BGL datasets, and produces comparable performance on the Thunderbird dataset. Finally, LogFiT is robust to future changes in the syntactic structure of log paragraphs because of its built-in ability to handle out-of-vocabulary tokens.

8 Future Work

The LogFiT model at its core is a BERT-based language model trained to reconstruct normal log data. As such, it can be adapted for use in any log analysis task where the log samples consist of textual description of system events. While the domain and task tackled in this study is system log anomaly detection, LogFiT is intended to be used in the cyber-security domain. In future we focus on applying the LogFiT anomaly detection approach on cyber-security datasets.

References

Australia Department of Home Affairs. 2020. Australia's cyber security strategy 2020.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer.

Marc Brysbaert, Michaël Stevens, Paweł Mandera, and Emmanuel Keuleers. 2016. How many words do we know? practical estimates of vocabulary size dependent on word definition, the degree of language input and the participant's age. *Frontiers in psychology*, 7:1116.

Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R. Lyu. 2021. Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection.

Xiang Dai, Ilias Chalkidis, Sune Darkner, and Desmond Elliott. 2022. Revisiting Transformer-based Models for Long Document Classification. In *The 2022 Conference on Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *The 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, pages 4171–4186.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1285–1298.

Yoav Goldberg. 2019. Assessing BERT's Syntactic Abilities.

Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. LogBERT: Log Anomaly Detection via BERT. *Proceedings of the International Joint Conference on Neural Networks*, 2021-July.

Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R. Lyu. 2021. A Survey on Automated Log Analysis for Reliability Engineering.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained Transformers Improve Out-of-Distribution Robustness. In *The 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751.

Jeremy Howard and Sylvain Gugger. 2020. Fastai: A layered api for deep learning. *Information*, 11(2):1–26.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 328–339.

International Business Machines. 2022. Cost of a data breach report 2022.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2020. What does BERT learn about the structure of language? In *ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf.*, pages 3651–3657. Association for Computational Linguistics (ACL).

Van-Hoang Le and Hongyu Zhang. Log-based anomaly detection with deep learning: How far are we? In *Proceedings of the 44th International Conference on Software Engineering*, page 1356–1367.

Van-Hoang Le and Hongyu Zhang. 2021. Log-based anomaly detection without log parsing. In *The 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 492–504.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open Sesame: Getting inside BERT's Linguistic Knowledge. pages 241–253.

Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. 2020. Self-attentive classification-based anomaly detection in unstructured logs. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, volume 2020-Novem, pages 1196–1201. Institute of Electrical and Electronics Engineers Inc.

Adam Oliner and Jon Stearley. 2007. What supercomputers say: A study of five system logs. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 575–584.

Harold Ott, Jasmin Bogatinovski, Alexander Acker, Sasho Nedelkoski, and Odej Kao. 2021. Robust and Transferable Anomaly Detection in Log Data using Pre-Trained Language Models. *Proceedings - 2021 IEEE/ACM International Workshop on Cloud Intelligence, CloudIntelligence 2021*, pages 19–24.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Neural information processing systems foundation.

RiskIQ. 2019. The evil internet minute 2019.

Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. 2019. Deep Semi-Supervised Anomaly Detection.

Thorsten Wittkopp, Alexander Acker, Sasho Nedelkoski, Jasmin Bogatinovski, Dominik Scheinert, Wu Fan, and Odej Kao. 2021. A2Log: Attentive Augmented Log Anomaly Detection. *HICSS 2022 : Hawaii International Conference on System Sciences*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *arXiv preprint arXiv:1910.03771*, pages 38–45.

Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. 2010. Detecting large-scale system problems by mining console logs. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 37–44.

David Yenicelik, Florian Schmidt, and Yannic Kilcher. 2020. How does BERT capture semantics? A closer look at polysemous words. In *The Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 156–162. Association for Computational Linguistics (ACL).

Lun Pin Yuan, Peng Liu, and Sencun Zhu. 2021. Recompose Event Sequences vs. Predict Next Events: A Novel Anomaly Detection Approach for Discrete Event Logs. In *ASIA CCS 2021 - Proc. 2021 ACM Asia Conf. Comput. Commun. Secur.*, volume 1, pages 336–348. Association for Computing Machinery.

Nengwen Zhao, Honglin Wang, Zeyan Li, Xiao Peng, Gang Wang, Zhu Pan, Yong Wu, Zhen Feng, Xidao Wen, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2021. An empirical investigation of practical log anomaly detection for online service systems. In *ESEC/FSE 2021 - Proceedings of the 29th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, volume 21, pages 1404–1415.