Applied Intelligence (2023) 53:22297–22312
https://doi.org/10.1007/s10489-023-04674-6

Detecting log anomaly using subword attention encoder and probabilistic feature selection

M. Hariharan[1] · Abhinesh Mishra[1] · Sriram Ravi[1] · Ankita Sharma[1] · Anshul Tanwar[1] · Krishna Sundaresan[1] · Prasanna Ganesan[1] · R. Karthik[2]

Accepted: 26 April 2023 / Published online: 26 June 2023

Abstract
Log anomaly is a manifestation of a software system error or security threat. Detecting such unusual behaviours across logs in real-time is the driving force behind large-scale autonomous monitoring technology that can rapidly alert zero-day attacks. Increasingly, AI methods are being used to process voluminous log datasets and reveal patterns of correlated anomaly. In this paper, we propose an enhanced approach to learning semantic-aware embeddings for logs called the Subword Encoder Neural network (SEN). Solving upon a key limitation of previous semantic log parsing works, the proposed work introduces the concept of learning word vectors from subword-level granularity using an attention encoder strategy. The learnt embeddings reflect the contextual/lexical relationships at the word level. As a result, the learnt word representations precisely capture new log messages previously not seen by the model. Furthermore, we develop a novel feature distillation algorithm termed Naive Bayes Feature Selector (NBFS) to extract useful log events. This probabilistic technique examines the occurrence pattern of events to only select the salient ones that can aid anomaly detection. To our best knowledge, this is the first attempt to associate affinity to log events based on the target task. Since the predictions can be traced to the log messages, the AI is inherently explainable too. The model outperforms state-of-the-art methods by a fair margin. It achieves a 0.99 detection F1-score on the benchmarked BGL, HDFS and OpenStack log datasets.

Keywords Deep learning · Self-attention · Naive Bayes · Syslog · Anomaly detection · Encoder-decoder

1 Introduction

As cloud computing applications scale to high loads and complex distributed environments, they become vulnerable to software bugs and failures. Usually, these large-scale systems are designed for robustness and stability. However, when a service fault or outage occurs, the mean time to recovery is considerable due to the inherent complexity of various components. Recently, the Google Cloud was affected by an incorrect configuration change issue for nearly two hours [1], while AWS faced a major outage that caused traffic delays and latency for about 7 h [2]. Thus, the need for diagnosing system anomaly plays a vital role in building trustworthy and reliable software. A rapid and precise failure detection helps troubleshoot issues faster, also motivates the plausibility of fully self-healing workflows that can perform auto-actions.

Logs are the preferred data source for root cause diagnosis of anomaly, because they record important events and holistic system status information, and are available on most

✉ M. Hariharan
vshshv3@gmail.com

Abhinesh Mishra
abhinemi@cisco.com

Sriram Ravi
srravi@cisco.com

Ankita Sharma
ankitas6@cisco.com

Anshul Tanwar
atanwar@cisco.com

Krishna Sundaresan
ksundar@cisco.com

Prasanna Ganesan
prasgane@cisco.com

R. Karthik
r.karthik@vit.ac.in

[1] Cisco Systems India Pvt Ltd, Bengaluru, India

[2] Center for Cyber Physical Systems, Vellore Institute of Technology, Chennai, India

platforms [3]. However, using logs comes with certain challenges. Firstly, the structural format and semantics of logs can vary across systems. Often, linking log messages generated by different system modules can be non-compatible given their diverse semantics [4]. Further, concurrent/parallel execution of tasks can produce random events at different times which does not share proper ordering or a deterministic ordering of messages [5]. Also, unexpected events that occur in real-time need to be evaluated to see if they are anomalous or not, which can be demanding even for domain experts [6]. Though heuristics such as regular expressions or keyword searching help automate to a certain extent, their scope is highly limited. They generally report false positives, do not scale and need to be constantly updated for new failures and code churn.

Such practical difficulties call for efficient analytical models that can leverage massively large logs to automatically mine valuable information for anomaly capture. Consequently, many works explore Artificial Intelligence (AI) for processing logs. The state of the art can be roughly classified into methods that use template-based features [4, 6, 10, 18-20], and ones that encode logs into vectors [7, 9, 11, 12, 15]. In the first category, a log entry is parsed into its invariant part (template) by discarding numeric information as parameter values and timestamps. The templates are assigned unique event indexes and anomaly detection is performed over the event index sequences using LSTM. Some methods create vectors for time windows by counting unique events, or TF-IDF aggregation. Then, the anomaly is learnt through unsupervised techniques like Principal Component Analysis (PCA) or Invariant Mining (IM). However, when only using these event indexes as features, valuable information can be lost because the templates can still contain semantic relationships. Considering these drawbacks, some methods directly use the extracted log templates as textual features, but they cannot handle those templates not seen before. However, in both cases, the quality of features degrades when the parser is inaccurate. Commonly used FT-Tree or Longest Common Subsequence (LCS) based parsing requires ideal settings and extensive tuning to obtain the best results. In contrast, log vectorization has gained traction in more recent works because of its capacity to encode word context. The sequential ordering of words can also be retained by embedding logs using transformers or recurrent neural networks. However, word2vec or skipgram create huge vocabulary spaces that can dilute learning over large-sized datasets. words not observed in training make it less resilient to new events that emanate in real-time.

There are a few limitations in the current arts for learning robust log embeddings. Representability of out-of-vocabulary (OOV) tokens is a key research gap. Although the Log2vec [7] approach is able to produce semantic/lexically charged word vectors, it relies on an offline compositional MIMICK technique to handle OOV words. Tokenizing logs into subwords was tried by a few works, but it substantially increases the sequence length. There is a need to develop a robust word-2vec strategy for logs that inherently eliminates OOV without computation overheads. It should process from subword granularity but render embeddings that reflect the semantic/lexical properties at the word level. Second, rarely few works explore feature selection in the context of logs. Since an anomaly is typically below 2% of the entire log volume, there is a need to enhance its visibility by eliminating noisy and less useful logs that do not affect anomaly detection.

In this paper, we propose a novel log encoder and feature selection mechanism to process log texts. This encoder operates at subword precision but ensures that the aggregated word embeddings reflect the semantic/lexical relations. Additionally, the most desired set of events to learn about the presence of anomaly are auto-selected for anomaly classification. This is accomplished by modelling the probability distribution of event occurrences. For example, heartbeat signals, and daemon start/stops correlate to anomaly and are prioritized over standard info messages.

The main contributions of this work are as follows.

- We propose a novel Subword Encoder Neural network (SEN) as an enhancement over the current Log2vec algorithm. This novelty boosts the representability of program/system-level entities such as machine IDs, IP/MAC addresses, error tracebacks, etc.
- The SEN implicitly takes care of handling out-of-vocabulary words. It is robust to new kinds of events, as semantic/lexical dependencies are encoded in the distributed vectors through subword-level self-attention.
- A novel Naive Bayes-based algorithm is proposed for feature selection. By evaluating the occurrence probabilities of different log events under failures, this model precisely filters the key log messages aiding failure classification.
- The proposed selection mechanism improves the efficacy of existing Machine Learning (ML) detectors. It ensures better separability of features for outlier tagging.

The remainder of the paper is as follows. Section 2 reviews the latest trends in literature. Section 3 describes the proposed work in detail, followed by a comprehensive results analysis in Sect. 4. The conclusion section summarizes the key findings of this work.

2 Related works

This section reviews the recent AI developments in anomaly detection from logs. In addition, popular Natural Language Processing (NLP) strategies for feature selection are discussed.

2.1 Anomaly detection

Several works have explored Long Short Term Memory Networks (LSTM) to learn log dependencies and predict the likely next sequence in normal logs. An anomaly manifests in real-time when LSTM finds a lesser chance for its occurrence. This approach is unsupervised in nature, as the anomalous patterns are unknown to the model until inference time. For example, Du et al. proposed the DeepLog model that relied on the sequential occurrence of log events to judge anomalous next logs [8]. An LSTM was fit over the event identifiers and parameters to detect execution path and performance anomalies. Zhou et al. proposed a log pattern-driven anomaly detection model that uses statistical features (frequency, surge) to find transient anomalies [4]. The approach adopts LSTM to correlate long-range temporal patterns which are collectively supplied into a Back Propagation (BP) neural network to obtain anomaly decisions. Yin et al. developed a dual LSTM model that combinedly analyses both the sequence of log templates and the components that emitted them [5]. The fused outputs are iterated over time steps to predict the next logs. Along similar lines, Chen et al. exploited pre-order and post-order relationships across log event sequences to train a dual LSTM [10].

Meng et al. introduced the concept of lexical contrast embeddings to enhance the template representations over DeepLog [9]. The template vectors were modelled using LSTM to identify sequential/quantitative anomalies. In an earlier work Log2vec, the authors employed similar semantic dependency parsing and synonym/antonym concepts to learn word vectors in log texts [7]. Lv et al. filtered invalid log words based on parts of speech and learnt word vectors [11]. The derived log-level embedding was passed through LSTM to determine anomaly. Yang et al. added a self-attention layer over the LSTM to enhance the embedding representations [12]. Li et al. proposed time-semantic sentence embeddings that reflect changes in sequence order, log time interval, and events. This dual nature of logs is encoded into two vector sequences that are transformed using bidirectional LSTM [13]. In an updated version of this framework, the authors introduced identifier-based relationship graphs to group interleaved messages from different processes [14]. It also included an enhanced data-driven log parser for producing effective semantic-temporal embeddings.

Another set of studies involved Bidirectional Encoder Representations from Transformers (BERT) for efficiently encoding the log information. Lee et al. trained BERT with only normal system logs [15]. The predictive probability for masked tokens out to be low when abnormal logs were passed to this trained model. Wang et al. applied BERT and variational autoencoder to extract statistical and semantic features through contrastive adversarial training [16]. LogBERT method proposed by Guo et al. modelled the log contexts through masked log key prediction and minimizing closeness between normal logs [17].

Some works explore Convolutional Neural networks (CNN) for spatio-temporal processing of log sequence data. For instance, to process streaming logs, Lu et al. and Wang et al. devised a lightweight temporal (CNN) [6, 33]. The logs are parsed into key sequences which are then embedded via convolutions and determined to be anomalous or not. Similarly, Hashemi et al. designed a character-based hierarchical CNN that performs sequence classification [18]. It aggregates features level-wise from character to line to a sliding window sequence.

To effectively utilize raw logs without any prior anomaly information, unsupervised techniques are vastly explored. For instance, Niwa et al. created a relationship graph based on the interconnections of system components and their running state metrics [19]. A centroid-based clustering was applied to detect outliers in an unsupervised fashion. Similarly, Farzad et al. learnt an autoencoder to vectorize logs and spotted anomalies using isolation forest [34]. Zeufack et al. also exploited log keys to create event counting features useful for density-based clustering [20]. In this approach, an anomaly is evaluated considering its core and reachability distances from the found clusters.

2.2 Feature selection

Feature selection on textual datasets like logs is a strategy to distil significant keywords/variables that can form a basis for classification [21]. They effectively process high-dimensional feature spaces, while preserving information gain and reducing runtime complexity. Bommert et al. experimented with 22 filter methods to correlate and rank the best ones for different kinds of data [22]. It is seen that while the essential features contribute to the objective, their combination is more significant. In another work, Iqbal et al. discuss the taxonomy of feature selection and its applicability for text categorization, remote sensing, and image retrieval [23].

Typically, there are two approaches for picking the ideal features, the wrapper and filter models. The wrapper models generate different feature sets and apply classifiers to evaluate and identify the best combinations [24]. On the other hand, the filter techniques use statistical measures such as correlations between predictor and target variables to decide feature weight scoring [25]. These metrics are commonly drawn from information theory. For example, Prasetiyowati et al. used entropy criteria as the basis to weigh important features [26]. Wang et al. investigated trends in the number of features and their relationship to the classification performance to reach the optimal selections [27].

The mechanism proposed in our work defines feature affinity based on their Naive Bayes occurrence probabilities. It is a derivative of the Naive Bayes algorithm widely

used in text classification tasks, like sentiment analysis, topic labelling, etc. [28]. Other feature selection techniques include Chi-Square, Information Gain (IG), and Recursive Feature Elimination (RFE) [29]. Ismail et al. proposed a system that can accurately distinguish between real-human and bot-generated texts based on the measurements gathered in a study using the Naive Bayes and entropy classifiers [30]. Similarly, Bird et al. applied relative entropy to select highly polar sentiments from a dataset of word stem attributes [31]. This enabled precise sentiment recognition. In summary, both the percentage of information gain and the dependencies between predictor/target variables are major aspects in the design of accurate feature selectors.

3 Proposed work

The goal of the proposed deep learning model is to learn robust distributed representations for the individual words comprising the logs. These word embeddings are formed such that they accurately capture the semantic sense of system/program entities (numbers, IPs, emails) typically found in logs, also generalize well to out-of-vocabulary words in unseen logs.

From the architecture diagram presented in Fig. 1, the logs are considered in blocks of a fixed number of lines, *N*. In the first step, the log messages are converted to a vector sequence via a novel semantic encoder. The subsequent step selects the most useful log message vectors for anomaly detection. This is to ensure that noisy/irrelevant messages are discarded while the remaining are retained in time order. The embeddings are merged to produce a log-level representation which can be passed into an ML classifier. In summary, the model ascertains whether the log block carries an anomaly or not in a supervised fashion.

3.1 Subword encoder neural network (SEN)

Log parsing identifies the underlying format used to generate the logging statements. Conventional template extraction methods such as FT-Tree (Frequent Template Tree) or Longest Common Subsequence (LCS) produce distinct log event categories. However, these event categories are not entirely unique but can still be semantically similar/related to each other. As illustrated in Fig. 2, FT-Tree assigned different template IDs to the first two log events, although they convey the same information. Hence there is a need to cluster the log messages based on their underlying meaning that strengthens the semantic relationships for determining the degree of similarity.

In this log parsing layer, we present a new mechanism to learn rich contextual log embeddings. The steps are illustrated in Fig. 3. A log message is split into words on the space character. The aim here is to generate a unique word-level embedding for any log word in the dataset. A log message can then be expressed as a sum of its word vectors.

However, two key challenges are faced in creating robust word vectors. Firstly, the vocabulary of words in a log base constantly changes, resulting in new words/events being observed during real-time detection. Secondly, the parameter terms inherent to logs such as machine IDs, memory addresses, variables, etc. need to be precisely expressed so that any unseen IDs or entities can still have similar embedding. To overcome these issues, we introduce an encoder neural network that operates at the subword level. Specifically, the Byte-Pair Encoding (BPE) algorithm is run over all tokens in the corpus to form an all-inclusive vocabulary set. From 256 bytes base tokens, 50,257 vocabulary size is reached by performing 50,000 merges. This ensures that even complex entities can be efficiently decomposed to the most representative granular sub-entities.

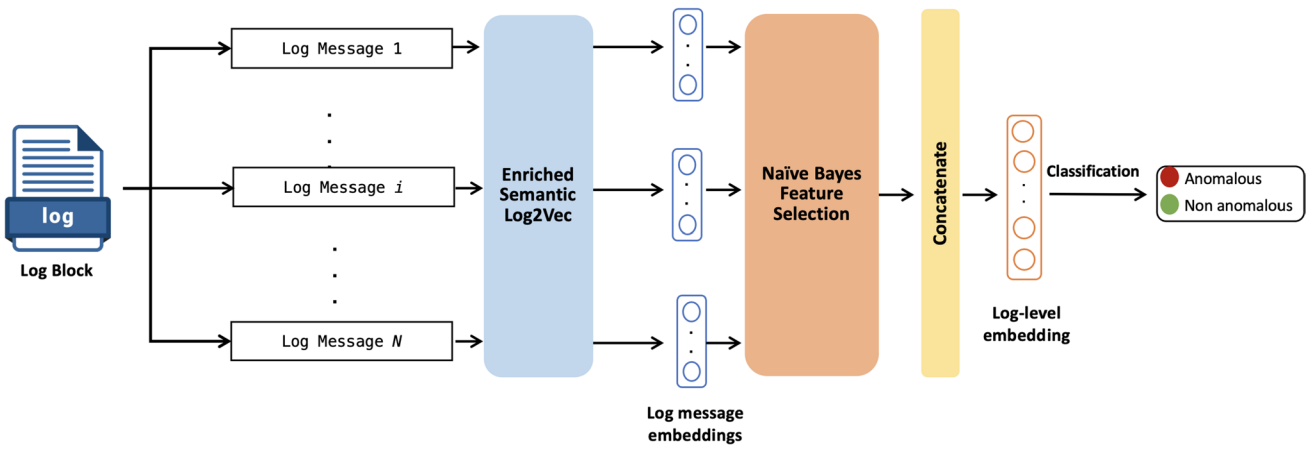


Fig. 1 Schematic diagram of the proposed log anomaly detection framework

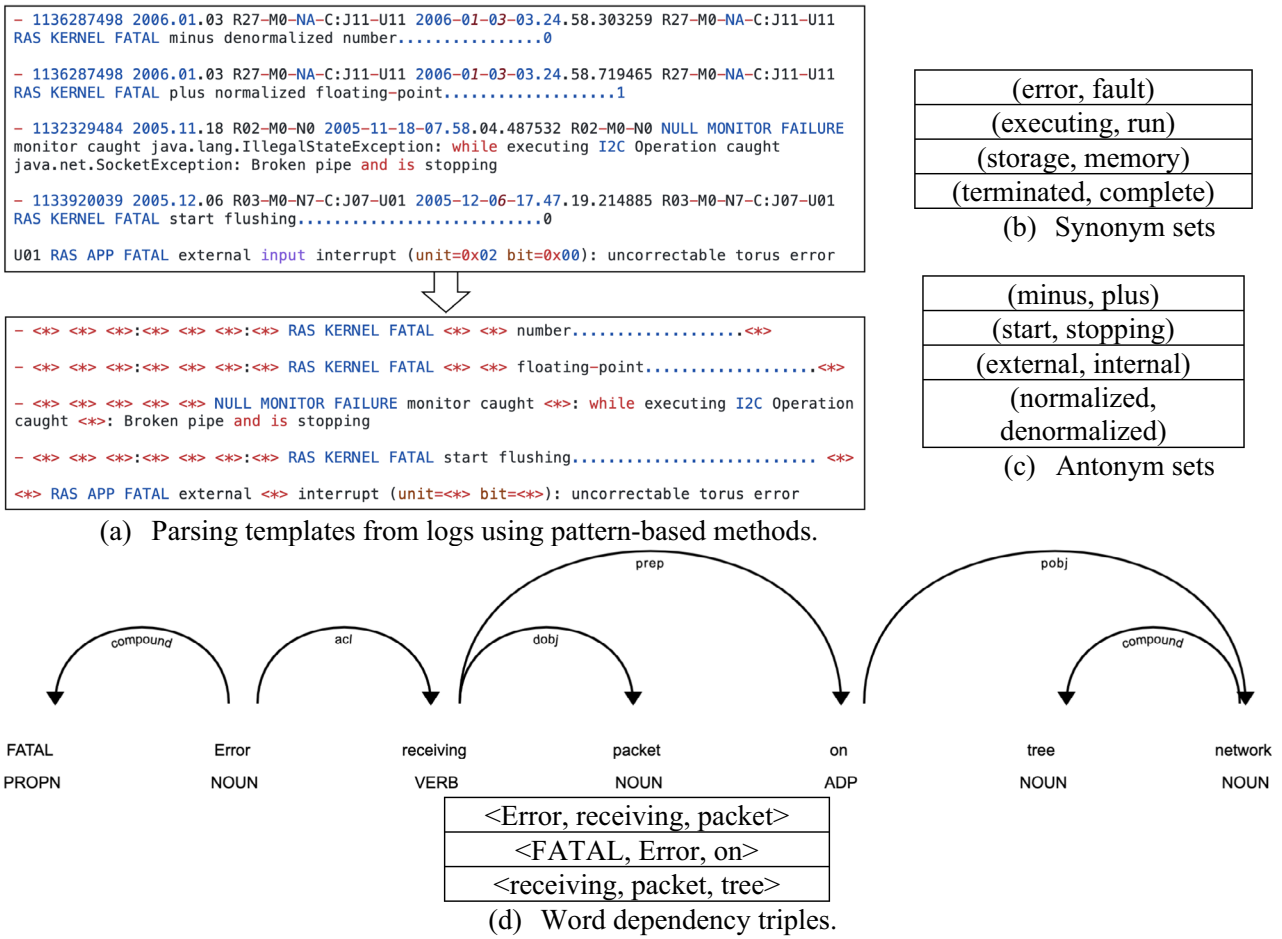


Fig. 2 Examples of logs ideal for lexical constraints and semantic similarity

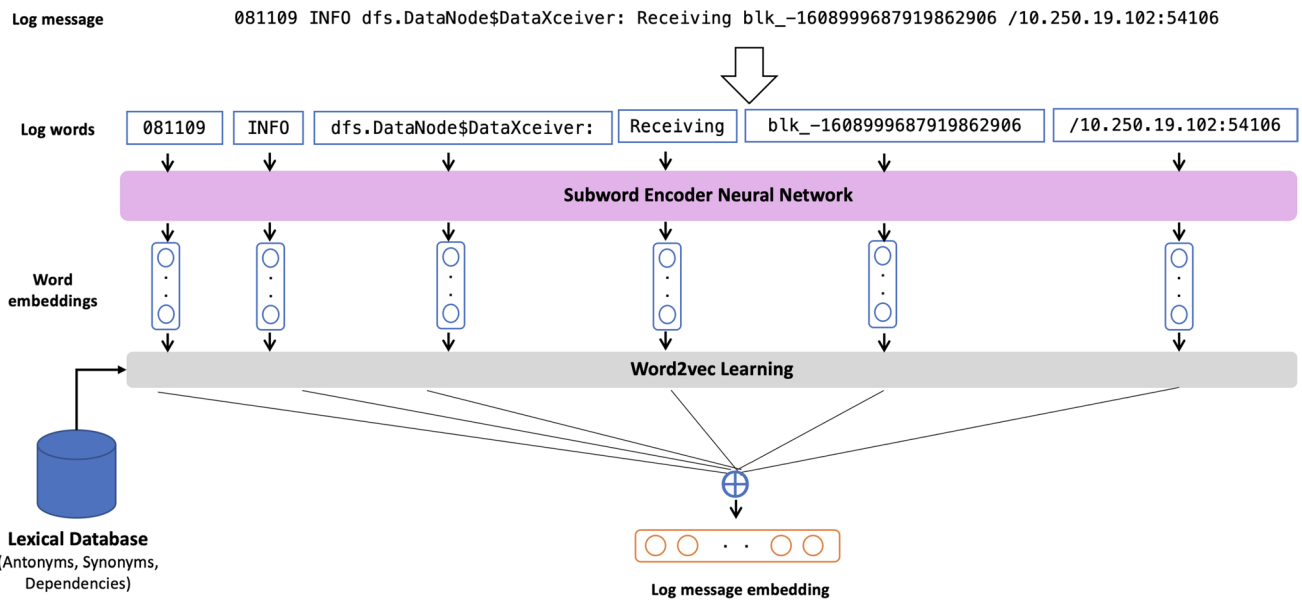


Fig. 3 Generating log event vectors

This proposed log word encoder is presented in Fig. 4. Here, every word w is treated as a sequence of m subwords. They are converted to D-dimensional features, $F \in \mathbb{R}^{m \times D}$ through an embedding lookup. To integrate deep contextual information across subword features in the word, self-attention learning is performed over F.

The self-attention layer works by first deriving three sets of features from F – queries Q, keys K, and values V. These entities are formed as linear projections on F using the weight matrices $W_Q \in \mathbb{R}^{D \times D}$, $W_K \in \mathbb{R}^{D \times D}$, and $W_V \in \mathbb{R}^{D \times D}$, as given by Eq. (2–4).

$$Q = FW_Q \tag{1}$$

$$K = FW_K \tag{2}$$

$$V = FW_V \tag{3}$$

The matrix dot product Q and K yields the attention coefficient map. Its values are row-wise softmax normalised to ensure appropriate weighing of the features. Each cell in this attention map carries the degree of correlation between the subwords referred to by that row and column. Finally, the transformed feature-set, $A \in \mathbb{R}^{m \times D}$ is calculated as a weighted combination of values V over the attention parameters. Equation (4) summarizes the computations involved.

$$A = softmax\left(\frac{QK^T}{\sqrt{D}}\right)V \tag{4}$$

By attending to every part of the word, each subword encoding selectively infuses semantic attributes and linkage information depending on the words it is present in. The attention mechanism also retains salient features suppressing any irrelevant details, thereby enhancing the intermediate subword representations for downstream processing.

In the subsequent step, A is additively pooled along the rows to generate the word embedding E. Given such deep representations from the encoder, the goal is to train context-based word vectors to predict the target word. Let $W \in \mathbb{R}^{|W| \times D}$ be defined as the collection of SEN encodings for all word tokens in the dataset V. Then this skip-gram objective can be expressed as per Eq. (5).

$$L_{skipgram} = -\sum_{-c \le j \le c} log\left(p\left(w_{i+j}|w_i\right)\right) \tag{5}$$

where c is the length of the context window. w_i is the input central word and w_{i+j} denotes its neighbouring words. The conditional probability function is given by Eq. (6).

$$p\left(w_x|w_y\right) = \frac{exp\left(W_x.W_y\right)}{\sum_{k=1}^{V} exp\left(W_k.W_y\right)} \tag{6}$$

Here, W_x and W_y refer to rows in the SEN matrix. In addition to contextual meaning, it is useful to encode lexical similarities and contrast among the log words. As demonstrated in Fig. 2, certain synonymous words convey the same semantic sense. Naturally, their word vectors should be closer compared to their antonyms. We adopt the LSWE (Lexical-contrast Semantic Word embeddings) to constrain such semantic relations on the learnt embeddings [7], as in Eq. (7).

$$L_{LSWE} = -\sum_{u \in SYN_{w_i}} log\left(p\left(w_i|u\right)\right) + \sum_{u \in ANT_{w_i}} log\left(p\left(w_i|u\right)\right) \tag{7}$$

where, SYN_{w_i} and ANT_{w_i} denote the synonym and antonym sets of w_i.

We further augment the semantic embeddings by associating word dependencies, as described in the Log2vec

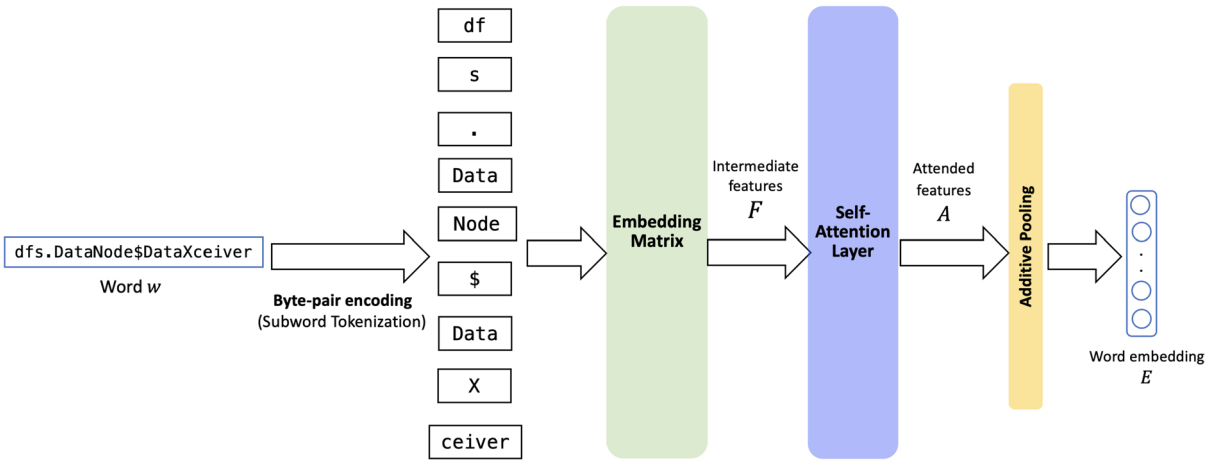


Fig. 4 Architecture of the subword encoder neural network

technique by Meng et al. [7]. Dependency parsing examines the grammatical relationships that exist between phrases in the log message. From examples in Fig. 2, a dependency links a head/root node to a child node via an association. Therefore, connected pairs (x, y) need to be closer than other words z in the message. Consequently, a set of relation triples (x, y, z) can be formed to represent this inequality, where the $sim(x, y) > sim(y, z) or sim(x, z)$. This constraint can be imposed on the objective function as a triplet loss [32].

$$L_{DP} = \sum_{u \in REL_{w_i}} max\left(\|w_i - u\|^2 - \|w_i - v\|^2 + \alpha, 0\right) \tag{8}$$

Thus, to train the SEN model, the final objective to minimize is the addition of these individual losses as presented in Eq. (9).

$$L = L_{skipgram} + L_{LSWE} + L_{DP} \tag{9}$$

Post optimization, any log message can be obtained as a vector sum of their component word embeddings.

3.2 Naive bayes feature selection (NBFS)

In practical scenarios, the occurrence of certain log events can lead to an anomalous pattern. Instead of supplying all log events in a time window for ML analysis, one level of feature engineering can be performed to determine the most helpful log messages that aid in anomaly detection. The advantages of template selection are three-fold: 1) eliminates unnecessary noise, 2) compresses input without information loss, and 3) enables faster convergence and precise model fitting.

To achieve this goal, we propose a novel probability-based technique to determine suitable templates. Leveraging the SEN model, all log lines in the dataset can be mapped to their corresponding log embeddings. An Agglomerative Nesting (AGNES) clustering algorithm is run to group similar messages into unique event categories. These log clusters are indicative of a specific kind of event, similar in terms to a template extracted by FT-Tree or LCS techniques. Let C be the set of distinct clusters obtained.

Since the ML classifier is trained on a fixed-length log block (N events at a time), let the dataset be prepared as sliding windows, X. Let $y \in \{0, 1\}$ be the target for anomaly classification, i.e., whether X points to a failure or not. From Bayes theorem, the posterior probability of determining y from X is given by Eq. (10).

$$P(y|X) = \frac{P(X|y) \times P(y)}{P(X)} \tag{10}$$

Here, the likelihood $P(x|y)$ can further be expressed as Markov chain function as per Eq. (11).

$$P(X|y) = P\left(x_1|y\right) \times P\left(x_2|y\right) \times \ldots P\left(x_n|y\right) \tag{11}$$

where x_i denotes a log event type from C that is found in the block X. Therefore, the probability of an event x_i to occur in a specific class of logs can be calculated as per Eq. (12).

$$P\left(x_i|y\right) = \frac{\#occurrences\ of\ x_i\ iny}{\#templates\ in\ y} \tag{12}$$

Using Eq. (12), $P\left(x_i|y = 0\right)$ gives the plausibility for event x_i to manifest in a normal log. Similarly, $P\left(x_i|y = 1\right)$ is the possibility of seeing x_i in an anomalous log.

The proposed feature selection technique computes the $P\left(x_i|y = 0\right)$ and $P\left(x_i|y = 1\right)$ statistics for all events $x_i \in C$ by traversing the entire dataset. In case the difference between these values for an event falls below a set threshold T, that event is discarded. It means that the distribution of occurrence of that event did not significantly vary/distinguish between normal and abnormal logs. Evidently, it is a frequent event that bears no significance for the task of anomaly classification. The value of T is determined empirically. In summary, only messages belonging to log clusters that satisfy this property are subjected to ML.

3.3 Anomaly detection

In a log stream, a specific time frame is sampled. The log messages are encoded via SEN and filtered on the NBFS event selector layers. Assuming the feature stack to have k log row embeddings in D-dimensions, they are reshaped into a 1-D vector. The concatenated feature-set is padded to a uniform length and classified on supervised ML models, as illustrated in Fig. 1.

4 Results and Discussions

This section discusses the datasets, experiments and findings based on various aspects of the model. In the final subsection, a performance comparison is drawn with the state-of-the-art methods for anomaly detection.

4.1 Data collection

The proposed log analytics framework was evaluated on three standard open-source datasets: 1) BGL, 2) HDFS, and 3) OpenStack. Their details are listed in Table 1. These datasets are a collection of real-time system logs that were labelled and freely released by authors of previous works. In the existing literature, they are commonly used as a benchmark for assessing the effectiveness of anomaly detectors.

The BGL logs are from the BlueGene/L supercomputer at the Lawrence Livermore National Labs (LLNL) in

Livermore, California [35]. The log messages are marked with alert category tags to indicate anomaly. This label information was utilized for supervised learning.

OpenStack is a set of software components to manage cloud services and infrastructure [36]. The dataset was generated on the CloudLab platform. It provides VM instances that have injected anomalies and the abnormal log sections pertaining to them.

The Hadoop Distributed File System (HDFS) log events describe the insertion, deletion, and updation of blocks in the Hadoop ecosystem [8]. An error/failure here can be traced to the associated block IDs.

4.2 Experimental setup

All experiments are run on a Ubuntu 18.04 server with 32 GB NVIDIA Tesla V100 GPUs. The memory and processors are 128 GB RAM, 96CPUs. The embedding layers are learnt as PyTorch modules through Adam optimization, while the log classification is performed using Sklearn libraries. The data processing scripts involve NLTK, regex and Spacy functions. For the log vector neural network, the initial learning rate is set to 5.0. It is controlled by a multiplicative learning rate decay scheduler, that drops by a factor of 0.1 when no improvement is observed over 10 epochs.

4.3 Model training and validation

To perform training the logs are traversed as sliding windows. Each window constitutes a chunk of log events that occurred in that fixed-length time frame. The window size is a hyper parameter that takes optimal values based on the dataset. The log blocks are assigned to be normal or anomalous based on the labelling information provided (refer Table 1).

Such data instances are partitioned into train, validation, and test splits in the ratio of 80:10:10 respectively. A shuffled and stratified data sampling is applied to preserve the proportions of normal and erroneous blocks in these split sets. The evaluation criteria are Precision, Recall and F1-score. These metrics are widely used for benchmarking log failure prediction models.

The entire log corpus is first subjected to SEN training as per Eq. (9) in order to learn appropriate weights for embedding layers. Table 2 captures the decay in this semantic word2vec loss function. The subword encoder is able to converge well on all three datasets and generate effective representations of their textual contents. This is evident as the loss on the unseen validation set dropped closer to the training loss towards convergence. It took 3–4 epochs to fully absorb the semanticity constraints. These log features are then classified as normal or anomalous using supervised ML. In the experiments, models tried include logistic regression, Support Vector Machine (SVM), random forests, and extreme gradient boosting.

In Table 2, learning curves are plotted for the ideal classifier and the best set of hyperparameters that gave the highest results on each dataset. The cross-entropy loss decays steadily through epochs. The number of epochs differs based on dataset complexity and type of classifier. Since HDFS blocks require processing a larger contextual window, it takes a longer training time. However, accuracy on unseen data matches that of the training set over time. Similarly, the F1 scores on BGL stagger initially but stabilize after 30 epochs. OpenStack logs are the most amenable to anomaly detection out of the three, hence their learning is smooth across iterations.

4.4 Ablation studies

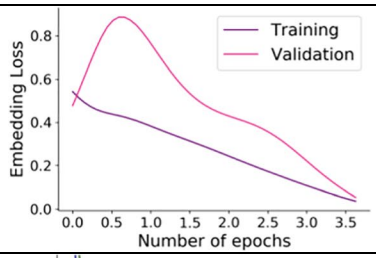
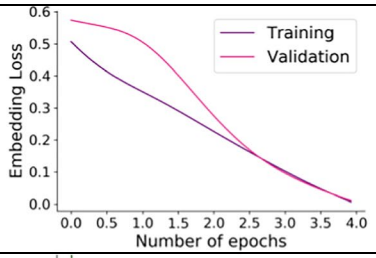
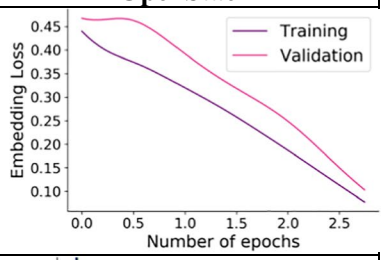
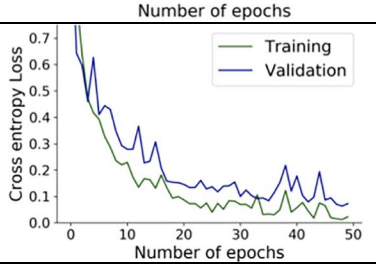
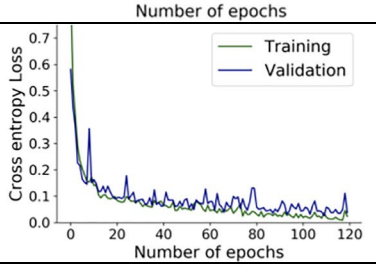
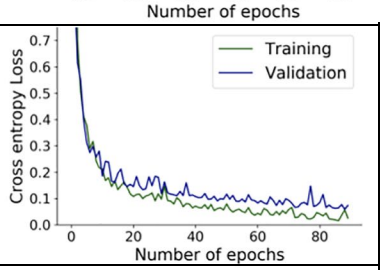
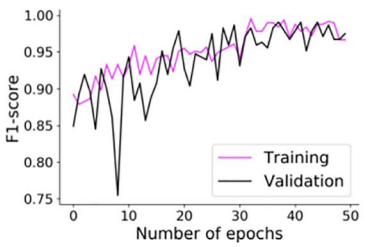
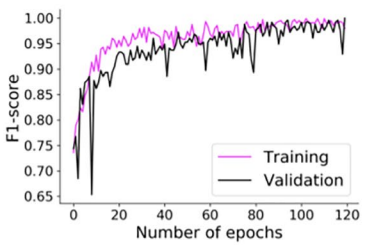
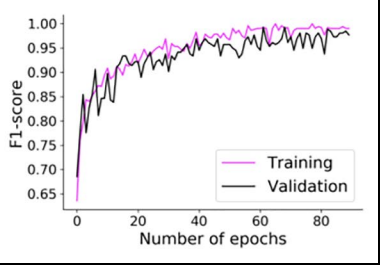
This section aims to evaluate the efficacy of individual components in the proposed framework. The effectiveness of the two key building blocks: 1) SEN and 2) NBFS is studied via ablation experiments.

Firstly, a vanilla baseline is established so that performance gains from the proposed modules can be measured incrementally. This base algorithm is a combination of parsed log template features and LSTM for self-supervised next log prediction. Such an approach is popular in current arts. It utilizes the error-free log sequences prepared in the previous step to model the distribution of logs, thereby identify outliers on the unseen set. Table 3 captures a holistic view of the ablation studies. It is evident this base method registers reasonable accuracy close to 90%, leaving much scope for enhancements. Especially, it needs to handle new events not observed during training. Also, the template extraction demands rigorous fine-tuning to predict the event types precisely.

Table 1 Datasets obtained from three sources. Provided are the counts of normal and erroneous log events present in the files

S. No	Dataset	Description	Total number of messages	Number of Anomalous messages	Number of normal log messages	Percentage of Anomaly (%)
1	Blue Gene/L (BGL) [35]	Super computer logs	4,747,963	348,460	4,399,503	7.33
2	HDFS [36]	Distributed File System Log	11,175,629	288,250	10,887,379	2.57
3	OpenStack [8]	Cloud Infrastructure log	207,820	18,434	189,386	8.87

Table 2 Trends observed during training on the datasets. The embedding loss applies to the word2vec learning at the SEN phase. Cross-entropy loss is used to classify log blocks at the final detection stage. F1 scores emitted at this ML classifier are tracked through epochs

Trends	BGL	HDFS	OpenStack
Embedding Loss			
Classification loss			
F1-score			

To augment this baseline, the SEN is introduced at the embedding learning stage. It solves both the challenges by learning robust representations for out-of-vocabulary words, also eliminates need for pattern-based log parsing. The SEN technique improves recall by 9.19% on BGL compared to the conventional method. A major contributing factor is self-attention learning which only focuses on significant details, unlike the LSTM that processes the entire context. On HDFS and OpenStack, the encoder enhances precision by 5.74% and 10.84% reducing any false positives.

By adding the NBFS layer, only the differential log events whose occurrence pattern greatly differs in the anomaly scenarios are supplied to ML. Inducing this event awareness into the context increases precision close to 1.00 on all the datasets. Evidently, the model exploits useful attributes to enhance boundary separation and distinguish the outliers, resulting in a higher recall by 4%.

4.5 Effects of embedding dimensionality on anomaly detection

Embedding size is a determinant factor of model performance. Choosing an appropriate length can improve the semantic effectiveness of the produced embedding features.

Table 3 Experimental results to validate efficacy of the proposed enhancements

S. No	Dataset	Models	Precision	Recall	F1-score
1	Blue Gene/L (BGL)	Baseline Log template LSTM model	0.93	0.87	0.89
		SEN log vectorization	0.96	0.95	0.96
		SEN filtered with NBFS	0.98	0.99	0.99
2	HDFS	Baseline Log template LSTM model	0.87	1.00	0.91
		SEN log vectorization	0.92	0.95	0.93
		SEN filtered with NBFS	1.00	0.99	0.99
3	OpenStack	Baseline Log word vector LSTM model	0.83	0.91	0.87
		SEN log vectorization	0.92	0.96	0.95
		SEN filtered with NBFS	0.99	1.00	0.99

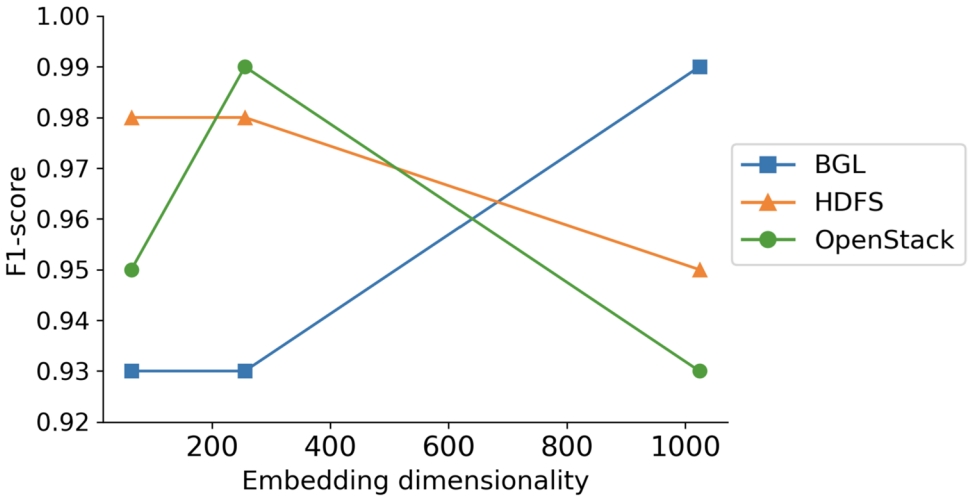


Fig. 5 Selecting optimal embedding size on different datasets

Consequently, it aids in the discriminability of keywords representing anomalous patterns. In this experiment, the dimensionality is varied in steps of 64 256, and 1024 and trained for the best classification model on each dataset (refer Fig. 5).

The BGL logs contained a greater number of distinct tokens and synonym-antonym sets. Therefore, it requires bigger dimensions of 1024 elements to encode this knowledge. Comparatively the OpenStack logs have lesser semantic dependencies, but more machine entities and identifiers. So, a small size such as 64 dimensions does not capture categorical similarity in these variable words. A length of 256 fits accurately. On the other hand, HDFS comprises fewer unique events and variations in the structure of messages. It responds well at 64 and 256 dimensions but degrades after that due to overfitting.

4.6 Effects of window size on the accuracy of anomaly detection

The size of the contextual window is a critical hyperparameter in the design of a log anomaly detector. A good enough time range will allow for sufficient correlations across the events to determine the presence of abnormality. Too small a size can deprive the ML of essential details that are helpful. On the other hand, very large size will invite noise and complex boundary fitting. It also incurs more memory and processing.

Specifically, the ideal window size is a characteristic of the dataset that depends on its nature and complexity. To obtain the optimal value, the search space for this parameter is varied through 1, 5, 10, 20, 50, 100, and 200. The F1 scores registered for the three datasets are visualized in Fig. 6.

It is observed for BGL, that a single alert/non-alert message is known to indicate whether an aberration occurred

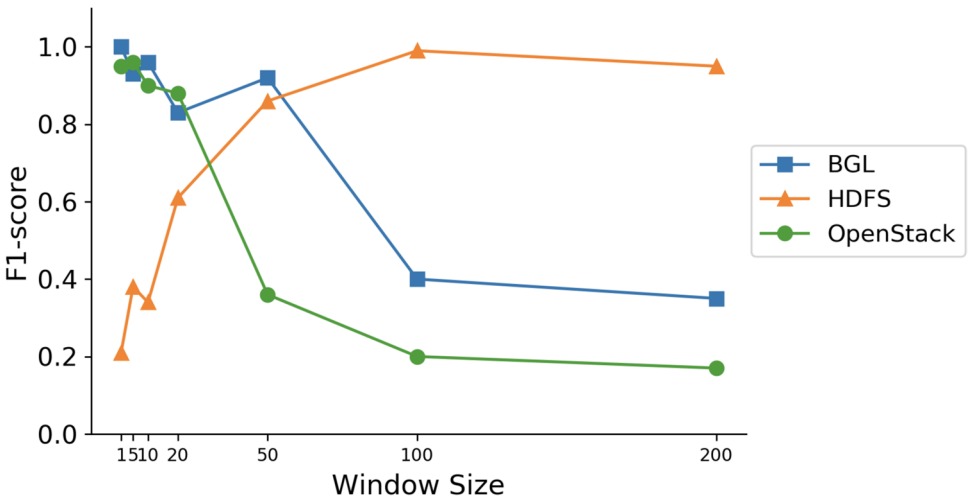


Fig. 6 Choosing the best window size to maximize accuracy. Scores are computed on the test set

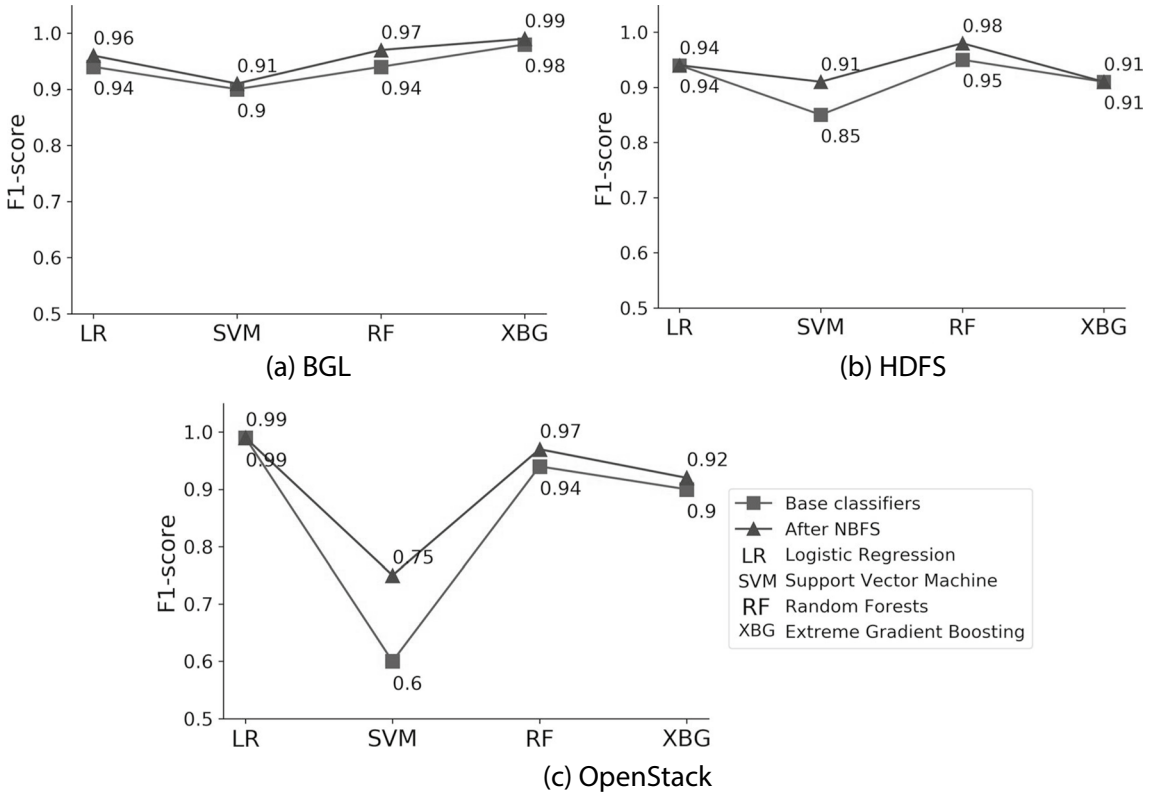


Fig. 7 Trends in ML model performance before and after applying NBFS. (**a**) BGL. (**b**) HDFS. (**c**) OpenStack

or not, hence one-sized window gives the highest F1 score. For OpenStack, a window size of 5 achieves the best performance. Since anomalies are present as transactions revolving around VM instances, it is expected to span a few sets of lines. In contrast, models tried on HDFS converge only for a window size of 100 and slightly dropped beyond that. As an HDFS anomaly manifests at a file-block level, a bigger window can trap most of that block-related messages amongst other log events. Therefore, it demands a larger size to observe event activity.

4.7 Improving efficiency of base classifiers using naive bayes feature selection

To demonstrate the impact of the proposed NBFS logic in pruning unnecessary logs, the behaviour of four classifiers in the presence and absence of this layer are investigated. Figure 7 plots the trends observed in F1 score before and after applying NBFS.

It is seen that regardless of the dataset or type of classification model, NBFS improves the distinguishability of anomaly. For SVM, it leads to a 25% and 7% increase in the correctness of predictions on OpenStack and HDFS respectively. With NBFS, the non-linear kernel feature space is well-formed to enable large-margin separation of outliers. A similar trend saw Random Forest increase by 3% uniformly across datasets. The decision trees had a lesser overfitting effect as the ideal depth decreased post-NBFS. In contrast, Logistic regression and Extreme Gradient Boosting models show lesser response to NBFS, as the base classifiers already reached maximal results.

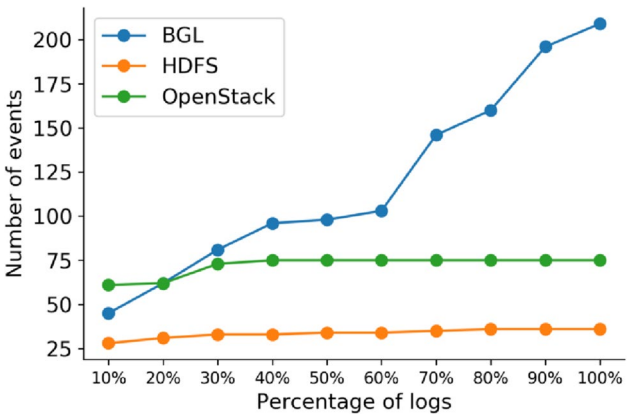


Fig. 8 Number of unique events discovered in the logs by parsing as proportions in the timestamp order

Table 4 Measuring accuracy when only a proportion of logs is used for training and rest taken into testing

S. No	Percentage of logs taken into training set	Test set performance (F1-score)		
		Blue Gene/L (BGL)	HDFS	OpenStack
1	0.1	0.93	0.96	0.93
2	0.2	0.93	0.97	0.95
3	0.3	0.93	0.99	0.97
4	0.4	0.94	0.99	0.98
5	0.5	0.96	0.99	0.98
6	0.6	0.97	0.99	0.99
7	0.7	0.97	0.99	0.99
8	0.8	0.99	0.99	0.99
9	0.9	0.99	0.99	0.99
10	1.0	0.99	0.99	0.99

4.8 Generalizability to unseen logs

Resiliency to new log events is a key strength of the proposed model. Even new messages only encountered in real-time can still be assessed as normal or erroneous. Figure 8 provides a plot of total log events encountered at different percentages of the log corpus.

While the HDFS and OpenStack logs contain a lesser number of distinct events that remain constant over time, the BGL comparatively shows more variations. The BGL dataset produced new log message clusters frequently as the trendline expands steeply. This property makes it an ideal candidate to test the robustness of the subword encoder to handle words unseen during training. The results of such an experiment for all three datasets are summarized in Table 4.

Especially for BGL, even when only the first 10% logs were subjected to learning, the model still gave a reasonable 0.93 F1-score on the remainder 90% data. It improves with more data, as it reached 0.96 at halfway mark. This trend confirms efficacy of the approach to function well even under new logs. In slight contrast, on HDFS and OpenStack the model already achieved maximal accuracy at 30% and 60% dataset respectively. These datasets had a redundant pattern of messages that did not greatly impact the unknown subword representations.

4.9 Performance comparison

This section presents a comparison of the proposed work with several state-of-the-art methods for log anomaly detection. To ensure fairness in comparison, only works that have experimented with the same datasets as the current work have been considered. Tables 5, 6 and 7 show the analysis of existing techniques on BGL, HDFS, and OpenStack datasets respectively. It is seen that the proposed SEN encoder alongside NBFS-augmented classification reached the best Precision, Recall and F1 score compared to most of the other works. The biggest advantage of our method lies in low memory requirement, fast compute times and simplified workflow integration for streaming logs.

From Table 5, Chen et al. achieved a below-par score on BGL, because the conventional log parsing to extract template/parameters does not cover all possible events (keys) in the training set for the next log key prediction [10]. This method gave a 0.97 F1-score on HDFS logs that contained fixed event types, whereas it could not adapt to the characteristics of the BGL dataset that produces more irregular events. Similar approaches such as Du et al., Meng et al., and Yang et al. also face the same drawback of not being

Table 5 Performance analysis of existing methods on the BGL logs dataset

S. No	Source	Method	Precision	Recall	F1-score
1	Chen et al. [10]	Dual LSTM	0.68	0.99	0.81
2	Yang et al. [12]	Self-attention LSTM	0.82	0.94	0.88
3	Guo et al. [17]	LogBERT	0.89	0.92	0.90
4	Meng et al. [7]	Log vectors clustering	0.94	0.94	0.94
5	Meng et al. [9]	LSTM on template features	0.97	0.94	0.96
6	Lee et al. [15]	BERT masked language model	0.95	0.96	0.96
7	Z Wang et al. [33]	Temporal CNN	0.95	0.99	0.97
8	Lv et al. [11]	LSTM on word2vec	0.97	0.99	0.98
9	Farzad et al. [34]	Isolation Forest on autoencoder features	0.97	0.98	0.98
10	Q Wang et al. [16]	BERT and variational autoencoder	0.99	0.99	0.99
11	Hashemi et al. [18]	Hierarchical CNN	0.99	0.99	0.99
12	Li et al. [13]	Attention Bi-LSTM	0.97	1.00	0.99
13	Proposed Work	Subword Encoder and Naïve Bayes Feature Selection	0.98	0.99	0.99

Table 6 Results comparison of the proposed model with similar research works on the HDFS dataset

S. No	Source	Method	Precision	Recall	F1-score
1	Guo et al. [17]	LogBERT	0.87	0.78	0.82
2	Zeufack et al. [20]	OPTICS outlier clustering	0.71	1.00	0.83
3	Meng et al. [9]	LSTM on template features	0.96	0.94	0.95
4	Du et al. [8]	LSTM on log keys and parameters	0.95	0.96	0.96
5	Lee et al. [15]	BERT masked language model	0.95	0.96	0.96
6	Meng et al. [7]	Log2vec clustering	0.96	0.96	0.96
7	Yin et al. [5]	Dual LSTM	0.93	0.98	0.96
8	Z Wang et al. [33]	Temporal CNN	0.94	0.99	0.97
9	Zhou et al. [4]	LSTM	1.00	0.94	0.97
10	Chen et al. [10]	Dual LSTM	0.96	0.98	0.97
11	Lv et al. [11]	LSTM on word2vec	1.00	0.98	0.98
12	Yang et al. [12]	Self-attention LSTM	0.97	0.99	0.98
13	Q Wang et al. [16]	BERT and variational autoencoder	0.99	0.99	0.99
14	Lu et al. [6]	CNN	0.98	0.99	0.99
15	Hashemi et al. [18]	Hierarchical CNN	0.99	0.99	0.99
16	Li et al. [13]	Attention Bi-LSTM	0.97	1.00	0.99
17	Proposed Work	Subword Encoder and Naïve Bayes Feature Selection	1.00	0.99	0.99

generalizable for new log keys [7, 8, 12]. Another popular approach involves modelling BERT to predict the likelihood of masked tokens. Lee et al. and Guo et al. obtain adequate F1 scores of 0.96 and 0.90 using BERT [15, 17]. However, due to diverse variability in logs, BERT predictive probabilities are unlikely to cover all possibilities in each context, resulting in false positives. Instead, considering these deep BERT features as input for a supervised classifier layer prevents precision errors.

Employing word vectors in place of log templates improves resilience in the embeddings for new log formats. These methods display better results. For instance, Z Wang et al. designed a semantic vector space model that cleanly highlights anomalous logs on a temporal CNN [33]. Li et al. use time and semantic embeddings to detect sequential anomalies [13]. These approaches acquire efficient representations but draw excessive contextual details and invariably noise too. The NBFS module proposed in our work eliminates such less relevant factors from impacting decision making. In place of text parsing, an end-to-end character-level neural network was presented by Hashemi et al. that achieved a 0.99 F1 score on BGL and HDFS [18]. This technique fails on OpenStack logs where the anomalies are too finely spread over certain VM instance messages to be solely distinguished at the character level.

Amongst LSTM methods, analysing multiple relationships such as pre/post order of events and component-aware templates improves the efficacy of LSTM. Yin et al. and Chen et al. demonstrate Dual LSTM that can inspect such patterns [5, 10]. These models capture long-term dependencies yet are not explicitly trained for semantic/contextual similarities between log words. On the other hand, Lv et al. that utilized word vectors converged with better precision [11]. In our proposed architecture, the SEN module ensures appropriate semantic-aware features for anomaly detection. Handcrafted features such as frequency, surge and variables have also been effective inputs for LSTM, as shown by Zhou et al. [4]. Nevertheless, they are derivative statistics and do not directly express inherent log contents, which in turn enables the LSTM to form better correlations.

Besides these approaches, autoencoders are shown to generate useful low-dimensional features for differentiating anomalies [16, 34]. Autoencoder presents a risk of lossy transformation. Training an autoencoder requires a lot of data, processing

Table 7 Evaluation of AI methods on the OpenStack dataset

S. No	Source	Method	Precision	Recall	F1-score
1	Hashemi et al. [15]	Hierarchical CNN	0.11	0.99	0.21
2	Niwa et al. [19]	MeanShift clustering	0.94	0.86	0.90
3	Du et al. [8]	LSTM on log keys and parameters	0.95	0.99	0.97
4	Farzad et al. [34]	Isolation Forest on autoencoder features	0.96	0.97	0.97
5	Zhou et al. [4]	LSTM	0.99	0.97	0.98
6	Proposed Work	Subword Encoder and Naïve Bayes Feature Selection	0.99	1.00	0.99

time and hyper-parameter tuning, whereas the proposed SEN is tuned directly for the word2vec objective and converges faster. Overall, in terms of lesser complexity and highest accuracy on multiple datasets, our method comes on par with the state of the art.

5 Conclusion

This article proposes a novel approach to learn log word embeddings that takes advantage of semantic/lexical relationships across words. It processes from a subword byte-pair vocabulary but ensures that contextuality is retained in the word-level embeddings. Learning such compositional word vectors inherently solves the representability of out-of-vocabulary tokens which is a key research challenge in this area. The ability of this module to operate under irregular events was confirmed through experiments. By only observing the first 10% logs, it gave a 93% F1 score on the BGL dataset, which proves its resiliency to new messages. Additionally, this paper introduces a probabilistic mechanism for selecting the most significant logs that can aid anomaly detection. It learns a Naive Bayes probability distribution for the occurrence pattern of events. Then, it identifies the salient ones that can reflect the difference between regular logs and abnormal logs. To our best knowledge, this is the first attempt to develop such a feature selector for logs. Empirically it was observed that this module improves performance of the base classifiers, to the extent of 25% for Support Vector Machine on OpenStack dataset.

The proposed framework was demonstrated on three benchmarked datasets. The learning curves imply that the models converged optimally. It reached mean 0.99 F1 scores on all three datasets, which exceed the current arts. As future work, the model can be expanded to more kinds of logs. The explainability of target predictions can be back-traced to features on the logfile, thereby opening pathways to self-healing workflows.

Data availability The syslog data that support the findings of this study are available in the LogHub public repository with the identifier(s). https://doi.org/10.48550/arXiv.2008.06448

Declarations

Competing interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests.

References

1. Google Cloud Fixes Outage That Hit Home Depot, Snap, Spotify. https://www.bloomberg.com/news/articles/2021-11-16/home-depot-amazon-web-services-websites-reportedly-see-outages (Accessed 28 June 2022)
2. Amazon Web Services' third outage in a month exposes a weak point in the Internet's backbone. https://www.washingtonpost.com/business/2021/12/22/amazon-web-services-experiences-another-big-outage/ (Accessed 28 June 2022)
3. Lin Q, Zhang H, Lou JG, Zhang Y, Chen X (2016) Log clustering based problem identification for online service systems. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp 102–111
4. Zhou P, Wang Y, Li Z, Wang X, Tyson G, Xie G (2020) Logsayer: Log pattern-driven cloud component anomaly diagnosis with machine learning. In: 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS). IEEE, pp 1–10
5. Yin K et al (2020) Improving Log-Based Anomaly Detection with Component-Aware Analysis. IEEE Int Conf Softw Maint Evol (ICSME) 2020:667–671. https://doi.org/10.1109/ICSME46990.2020.00069
6. Lu S, Wei X, Li Y, Wang L (2018) Detecting anomaly in big data system logs using convolutional neural network. In: 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, pp 151–158
7. Meng W, Liu Y, Huang Y, Zhang S, Zaiter F, Chen B, Pei D (2020) A semantic-aware representation framework for online log analysis. In: In 2020 29th International Conference on Computer Communications and Networks (ICCCN). IEEE, pp 1–7
8. Du M, Li F, Zheng G, Srikumar V (2017) Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 1285–1298
9. Meng W, Liu Y, Zhu Y, Zhang S, Pei D, Liu Y et al (2019) LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. IJCAI 19(7):4739–4745
10 Chen Y, Luktarhan N, Lv D (2022) LogLS: Research on System Log Anomaly Detection Method Based on Dual LSTM. In Symmetry. MDPI AG 14(3):454. https://doi.org/10.3390/sym14030454
11 Lv D, Luktarhan N, Chen Y (2021) ConAnomaly: Content-Based Anomaly Detection for System Logs. In Sensors. MDPI AG 21(18):6125. https://doi.org/10.3390/s21186125
12 Yang R, Qu D, Gao Y, Qian Y, Tang Y (2019) nLSALog: An Anomaly Detection Framework for Log Sequence in Security Management. In IEEE Access. Ins Electr Electron Eng (IEEE) 7:181152–181164. https://doi.org/10.1109/access.2019.2953981
13. Li X, Chen P, Jing L, He Z, Yu G (2020) Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults. In: 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE). IEEE, pp 92–103
14. Li X, Chen P, Jing L, He Z, Yu G (2022) SwissLog: Robust anomaly detection and localization for interleaved unstructured logs. IEEE Transactions on Dependable and Secure Computing
15. Lee Y, Kim J, Kang P (2021) LAnoBERT: System log anomaly detection based on BERT masked language model. arXiv preprint arXiv:2111.09564
16. Wang Q, Zhang X, Wang X, Cao Z (2021) Log Sequence Anomaly Detection Method Based on Contrastive Adversarial Training and Dual Feature Extraction. In Entropy. MDPI AG 24(1):69. https://doi.org/10.3390/e24010069
17. Guo H, Yuan S, Wu X (2021) LogBERT: Log Anomaly Detection via BERT. Int Joint Conf Neural Net (IJCNN) 2021:1–8. https://doi.org/10.1109/IJCNN52387.2021.9534113
18. Hashemi S, Mäntylä M (2021) OneLog: Towards end-to-end training in software log anomaly detection. arXiv preprint arXiv:2104.07324

19. Niwa T, Kasuya Y, Kitahara T (2017) Anomaly detection for openstack services with process-related topological analysis. In: 2017 13th International Conference on Network and Service Management (CNSM). IEEE, pp 1–5
20 Zeufack V, Kim D, Seo D, Lee A (2021) An unsupervised anomaly detection framework for detecting anomalies in real time through network system's log files analysis. In High-Confidence Computing. Elsevier BV 1(2):100030. https://doi.org/10.1016/j.hcc.2021.100030
21 Chakraborty B, Divakaran DM, Nevat I, Peters GW, Gurusamy M (2021) Cost-Aware Feature Selection for IoT Device Classification. In IEEE Internet of Things Journal. Inst Electr Electron Eng (IEEE) 8(14):11052–11064. https://doi.org/10.1109/jiot.2021.3051480
22 Bommert A, Sun X, Bischl B, Rahnenführer J, Lang M (2020) Benchmark for filter methods for feature selection in high-dimensional classification data. In Computational Statistics & Data Analysis. Elsevier BV 143:106839. https://doi.org/10.1016/j.csda.2019.106839
23. Iqbal M, Abid MM, Khalid MN, Manzoor A (2020) Review of feature selection methods for text classification. In International Journal of Advanced Computer Research (Vo 10, Issue 49, pp 138–152). Association of Computer, Communication and Education for National Triumph Social and Welfare Society (ACCENTS). https://doi.org/10.19101/ijacr.2020.1048037
24 Liu Y, Ju S, Wang J, Su C (2020) A New Feature Selection Method for Text Classification Based on Independent Feature Space Search. In Mathematical Problems in Engineering. Hindawi Limited 2020:1–14. https://doi.org/10.1155/2020/6076272
25 Thabtah F, Kamalov F, Hammoud S, Shahamiri SR (2020) Least Loss: A simplified filter method for feature selection. In Information Sciences. Elsevier BV 534:1–15. https://doi.org/10.1016/j.ins.2020.05.017
26. Gumilar A, Prasetiyowati SS, Sibaroni Y (2022) Performance analysis of hybrid machine learning methods on imbalanced data (rainfall classification). Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi) 6(3):481–490
27. Wang Z, Lin Z (2019) Optimal Feature Selection for Learning-Based Algorithms for Sentiment Classification. In Cognitive Computation (Vol 12, Issue 1, pp 238–248). Springer Science and Business Media LLC. https://doi.org/10.1007/s12559-019-09669-5
28. Vangara RVB, Thirupathur K, Vangara SP (2020) Opinion Mining Classification using Naive Bayes Algorithm. In International Journal of Innovative Technology and Exploring Engineering (Vol 9, Issue 5, pp 495–498). Blue Eyes Intelligence Engineering and Sciences Engineering and Sciences Publication - BEIESP. https://doi.org/10.35940/ijitee.e2402.039520
29. ThakkarA, Lohiya R (2020) Attack classification using feature selection techniques: a comparative study. In Journal of Ambient Intelligence and Humanized Computing (Vol 12, Issue 1, pp 1249–1266). Springer Science and Business Media LLC. https://doi.org/10.1007/s12652-020-02167-9
30. Ismail Z, Jantan A, Yusoff Mohd N, Kiru MU (2020) The effects of feature selection on the classification of encrypted botnet. In Journal of Computer Virology and Hacking Techniques (Vol 17, Issue 1, pp 61–74). Springer Science and Business Media LLC. https://doi.org/10.1007/s11416-020-00367-7
31. Bird JJ, Ekárt A, Buckingham CD, Faria DR (2019) High resolution sentiment analysis by ensemble classification. In: Intelligent Computing: Proceedings of the 2019 Computing Conference, vol 1. Springer International Publishing, pp 593–606
32. Schroff F, Kalenichenko D, Philbin J (2015) FaceNet: A unified embedding for face recognition and clustering. IEEE Conf Comput Vision Pattern Recog (CVPR) 2015:815–823. https://doi.org/10.1109/CVPR.2015.7298682
33. Wang Z, Tian J, Fang H, Chen L, Qin J (2022) LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge. In Computer Networks (Vol 203, p 108616). Elsevier BV. https://doi.org/10.1016/j.comnet.2021.108616
34. Farzad A, Gulliver TA (2020) Unsupervised log message anomaly detection. In ICT Express (Vol 6, Issue 3, pp 229–237). Elsevier BV. https://doi.org/10.1016/j.icte.2020.06.003
35. Oliner A, Stearley J (2007) What supercomputers say: A study of five system logs. In: 37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07). IEEE, pp 575–584
36. Xu W, Huang L, Fox A, Patterson D, Jordan MI (2009) Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pp 117–132

M. Hariharan is a Software Engineer at Cisco, primarily focused on research in machine learning for software productivity, security, and log incident analytics. He holds a Bachelor's degree in Computer Science from Vellore Institute of Technology, India. He has published deep learning papers dealing with the analysis of medical imaging datasets. His research interests revolve around learning theory, robustness and out-of-distribution generalization in machine learning.

Abhinesh Mishra is a Software Engineering Manager at Cisco. He has done Master of Science from Liverpool John Moore's University and is current pursuing Doctor of Business Administration from Swiss School of Business Management. His research interest includes Artificial Intelligence, Computer Networks, with emphasis on Software Defined WAN, Network Security and Network Analytics.

Sriram Ravi is a Project Manager at Cisco Systems, India. He has more than 16 years of professional experience. He joined Cisco as a tools engineer and has developed many tools. He took over the role of a project manager and is now leading 3 projects with expertise in Network operating systems, compiler, and coverage utilities. In the last few years, he has been instrumental in developing many ML-related tracks which would increase the productivity of the Cisco engineers. One of his key achievements is the Pioneer Award (2018). He was part of a defensive publication in predicting vulnerabilities.

Ankita Sharma is a seasoned software engineer with a focus on Artificial Intelligence and Machine Learning. Currently employed as a Software Engineer 3 at Cisco Systems India Pvt Ltd, Sharma brings 6 years of industry experience, with a dedicated focus on AI/ML for the past 4 years. Their expertise lies in designing and implementing cutting-edge AI/ML solutions, leveraging their strong programming skills and in-depth understanding of algorithms. Through their research, she aims to contribute to the advancement of AI/ML and its practical applications, bridging the gap between industry and academia.

Anshul Tanwar is a Principal Engineer at Cisco Systems. He has more than 20 years of network design and implementation experience. Over these years, he has architected many routing and switching products used by large tier 1 mobile and Metro Ethernet service providers across the world. He has led the SyncE and PTP architecture definition and

implementation for multiple access and pre-aggregation routers in Cisco. In his most recent role, Anshul was responsible for defining the deployment architecture of phase timing synchronization for one of the world's largest service provider LTE/LTE-A networks. He is also a co-author of the book, "Synchronizing 5G Mobile Networks" and co-inventor on four patents.

Krishna Sundaresan is Vice President, Engineering, leads the Cisco Intent-Based Networking Group, a team of about 1500+ engineers responsible for switching, routing, and wireless and controller product groups in India. He is the sponsor for Cisco Launchpad which is the accelerator for early-stage startups. He is closely engaged with N/core for fostering social entrepreneurship. He represents Cisco India engineering in many conferences and events, including Your Story, NASSCOM, and Tie Global Summit. He has Bachelor's in Computer Science from MNIT Allahabad and a Master's in Business Administration from Leavy Business School in Santa Clara, US. He has 16 patents in his name.

Prasanna Ganesan is Director of Software Development Engineering at Cisco Systems India. He has 16 years of experience in leading engineering teams for Development and Test Tooling, Internet, Enterprise Architecture, Desktop Integration, Mobile, and Cloud. He is a Technology Professional specialized in enterprise software architectural direction, tools and technology evaluations, and process improvements. He holds the reputation of being a Change Champion, providing world-class technology solutions to the business by leading change, managing budget, planning resources & growth, retaining talents, and managing the portfolio for globally dispersed teams.

R. Karthik obtained his Doctoral degree from Vellore Institute of Technology, India, and Master's degree from Anna University, India. Currently, he serves as Senior Assistant Professor in the Research Center for Cyber Physical Systems, Vellore Institute of Technology, Chennai. His research interest includes Deep Learning, Computer Vision, Digital Image Processing, and Medical Image Analysis. He has published around 32 papers in peer-reviewed journals and conferences.