This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

Domain Tailored Large Language Models for Log Mask Prediction in Cellular Network Diagnostics

Sayed Taheri*, Achintha Ihalage*, Prateek Mishra, Sean Coaker, Faris Muhammad, Hamed Al-Raweshidy

***Abstract*—Software logs generated by dedicated network testing hardware are often complex and bear minimal similarity to natural language, requiring the expertise of engineers to understand and capture defects recorded in these logs. This manual process is inefficient and expensive for both service providers and their clients. In this study, we demonstrate the transformative potential of Artificial Intelligence (AI), specifically through domain-tailoring of Large Language Models (LLMs) like RoBERTa, BigBird, and Flan-T5, to streamline the process of defect diagnostics. Particularly, we pre-train these models ground up on a real industrial telecommunications log corpus, and perform finetuning on a multi-label classification objective. This facilitates identifying a correct set of log points to be enabled for rapid detection of defects that arise during network testing. Despite encountering several challenges such as intricate text structures, heavily skewed label distribution, and inconsistencies in historical data labelling, our tailored LLMs achieve commendable performance on previously unseen defect cases, significantly reducing the turnaround times. This research not only serves as an exemplar for adapting LLMs in telecommunications industry for automated defect diagnostics, but also has wide implications for software log analysis across various industries.**

***Index Terms*—telecommunications, machine learning, LLM, log analysis, network diagnostics**

I. Introduction

The ever-evolving landscape of telecommunications industry demands thorough network testing and diagnostics that encompass the entire network protocol stack, ensuring the network reliability, performance and compliance with industry standards before real-world deployment. These testing solutions inevitably involve sophisticated hardware and massive software codebases for granular emulation of diverse network infrastructures and user equipment, closely replicating practical conditions. It becomes crucial for network operators to capture any defects or unexpected behaviour in their network under these dynamic environments and rectify them promptly to improve the overall quality of service. Software logs generated by a network testing platform, although highly complicated and barely related to natural language, are pivotal in identifying and troubleshooting issues occurred in a network test run. Nevertheless, this is generally a tedious manual process that demands the expertise of engineers with vast experience in the field. Undoubtedly, manual analysis of software logs for defect detection, whether performed internally by the service provider, or in the field by field application engineers hired by the clients, incurs substantial costs to both parties. Despite these costs, it is imperative not to underestimate the significance of defect resolution. Overlooking or delaying this process could have detrimental effects on business relationships and potentially lead to customer attrition.

While log classification and anomaly detection has been extensively studied especially in domains such as cybersecurity and system monitoring, the applications of machine learning (ML) algorithms for classifying logs generated by advanced testing hardware in telecommunications industry, such as VIAVI TM500, remains largely uncharted. Exploring this domain poses unique challenges owing to the voluminous and intricate nature of telecom logs. For instance, logs generated by VIAVI TM500 contain information related to hardware configuration, network and user equipment (UE) interactions across all protocol layers, running commands and responses among other details, all recorded with a precision down to the microsecond level. Inevitably, this results in software logs often spanning tens of thousands of lines of text and sometimes several millions of text lines for complicated test cases. When issues do occur in a network simulation, capturing only the errors or assert indications is typically insufficient to locate the defective component for debugging purposes. One must acquire a combination of log messages relating to system configuration, dynamic parameters, errors, warnings and asserts for isolating the defect. Furthermore, the default log points (commonly referred to as log bases (LBs)) enabled in a test script that gets executed on the TM500 may write limited number of log messages, sometimes resulting in the failure to record critical incidents. In such cases, engineers rely on other information specified above to commence the defect resolution process.

Dedicated network testers like VIAVI TM500 support over thousand log bases for testing very specific individual components in a network infrastructure. However, enabling all of these LBs in a test script may lead to catastrophic consequences, such as generating logs in Terabyte (TB) range, message buffer overflow and system crash. Therefore, it is crucial to activate only a small subset of log bases that are most likely to assist in pinpointing the issue. A hexadecimal representation of a relevant set of LBs is known as a *log mask*

Sayed Taheri, is with the Department of Electronic and Electrical Engineering (EEE), College of Engineering, Design and Physical Sciences, Brunel University of London, London, United Kingdom. Since 2022, he has also been with VIAVI Solutions Inc., United Kingdom. (e-mail: Sayed.Taheri@Brunel.ac.uk, and, Sayed.Taheri@viavisolutions.com).

Achintha Ihalage, Prateek Mishra, Sean Coaker, and Faris Muhammad are with the Wireless Business Unit of VIAVI Solutions Inc., United Kingdom. (e-mail: {achintha.ihalage, prateek.mishra, sean.coaker, and faris.muhammad}@viavisolutions.com).

Hamed Al-Raweshidy *(Corresponding author)* is with the Wireless Networks and Communications Group, Department of Electronics and Electrical Engineering, Brunel University of London, London, United Kingdom. (e-mail: Hamed.Al-Raweshidy@Brunel.ac.uk).

*These authors contributed equally.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

or a *log filter* which makes the transfer and application of a set of LBs to the customer easier. Selecting the correct set of LBs is a challenging task. Service providers (e.g., VIAVI) and their clients (e.g., cellular network operators and network equipment manufacturers) often find themselves in a time-consuming cycle of re-executing tests with new logging points in hopes of capturing the elusive but necessary data. On average, about eight iterations of re-running the tests with modified LBs, collecting new sets of logs and sending them back to the service provider for further analysis are required before the logs are deemed sufficient for debugging the defects. Understandably, this manual inefficient process adversely affect the turnaround times, service level agreements, and ultimately, revenue.

In this paper, we focus on applying ML to select a more accurate set of LBs from the outset, with the goal of reducing turnaround time and improving efficiency of VIAVI TM500 based 4G, 5G and beyond network diagnostics. The challenge of this task stems from the vast array of LBs to choose from, and the need for multi-label classification (MLC) as opposed to multi-class classification. In particular, we apply recent LLMs, domain-adapted from scratch for telecom software log understanding, to perform massive MLC of software logs into their corresponding set of LBs. Furthermore, to circumvent the problem of hugely skewed LB distribution, we propose an independent binary classification of logs into individual LBs by extracting software log embeddings from various domain-adapted LLMs, including RoBERTa [36], BigBird [37] and Flan-T5 [39]. Classical ML binary classifiers are applied on top of these embeddings to evaluate the applicability of individual LBs for a given piece of log text. Furthermore, we utilize a term frequency-inverse document frequency (TF-IDF) based embeddings paired with ML models as a baseline. The former strategy of end-to-end (E2E) finetuning of LLMs for MLC is benchmarked against the latter binary classification approach.

Off-the-shelf LLMs are primarily trained on natural language. Their applicability in specialised domains such as telecom log classification is rather limited, particularly when dealing with text formats that substantially deviate from any known natural language. While LLMs have been trained ground up for various disciplines, including biomedical [40], finance [41], climate [42] and law [43], no prior peer-reviewed studies have depicted the process of building domain-tailored LLMs pre-trained on intricate and voluminous software logs in telecommunications industry for downstream log classification. In fact, the corpora in the aforementioned industries still consist of natural language, predominantly, English. On the other hand, the text data prevalent in telecommunications logs has minimal similarity to both natural language and code. Consequently, the need arises to undertake domain-specific data pre-processing, along with the training of tokenizers and models *de novo*, in order to attain optimal performance in the downstream log classification task. In particular, here the LLMs and classical ML models are trained to predict a successful set of LBs (i.e., log mask) aimed at capturing only the essential information required for diagnosis, based on limited information available in software logs generated through default LBs.

Novel contributions in this research are summarized below.

1) We pre-train recent LLMs, namely RoBERTa and BigBird on our own industrial software log corpus to develop a deep understanding of the syntactic, semantic and contextual aspects within log data structures. Moreover, Flan-T5 is domain adapted on the same corpus using the low-rank adaptation (LoRA) technique [44]. Model-specific tokenizers are also individually trained to construct a software log vocabulary of appropriate size. Present study serves as an exemplar showcasing the process of building LLMs tailored for telecommunications network testing solutions, with wide applicability for software log understanding across various industries.
2) The pre-trained LLMs are finetuned on a downstream multi-label classification task for identifying a correct set of log bases to pinpoint the defects happened in a network emulation. The models are trained on real industrial customer-reported data from past three years, achieving acceptable performance given the sheer number of labels and human bias in the historical label sets. The best performing model is productionized and currently estimated to reduce the turnaround time (TAT) by $\sim$80% in defect resolution.
3) We extract embedddings of software log samples from several pre-trained LLMs and train independent binary classifiers on these embeddings to predict the presence of individual LBs. These predictions are aggregated over all the LBs and are used as the baseline for comparison against the performance of end-to-end finetuned LLMs.

Next, we discuss related work in the field of log classification and log anomaly detection including their strengths and weaknesses. We then illustrate our data acquisition, data preprocessing and model training pipelines in detail in section III. Section IV presents the results of LLM-finetuning and classical ML model training with a comparative analysis and discussion. In Section V, we delve into the specifics of our experiments, including model parameter settings, software packages employed, and the underlying computing infrastructure. Finally, this article is concluded in Section VI.

II. Related Work

The analysis of software logs has been extensively studied across various industries [1], [2], [4], [3], [5], [12], [13], including telecommunications [14], with applications ranging from failure diagnosis to anomaly detection. These tasks often leverage both supervised and unsupervised learning methods. Various ML architectures including classical models, deep learning approaches, and more recently, Transformer-based language models have been utilized for automated log analysis.

In the supervised learning domain, traditional machine learning (ML) techniques have been widely applied to software log classification. Decision trees (DT), support vector machines (SVM), and rule-based classifiers have long been utilized due to their interpretability and straightforward applicability. For instance, Chen et al. [6] used DTs to diagnose failures in large-scale internet services, showing that these methods can successfully identify the true causes of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

failure in production environments. Similarly, Liang et al. [7] developed a methodology to predict failures in the IBM BlueGene/L supercomputer using event logs. Their customized nearest-neighbor approach outperformed other classical ML techniques in precision and coverage, indicating its robustness for large-scale systems.

The rise of deep learning has brought more advanced architectures into log analysis. Recurrent neural networks (RNNs), particularly LSTM-based models, have shown great promise in capturing the sequential nature of log data. Du et al. [8] proposed DeepLog, an LSTM-based model that detects anomalies by predicting future log sequences and flagging deviations. This model highlights the capacity of LSTM to model long-term dependencies in log sequences, making it suitable for systems where log events occur in sequences over time. On the other hand, convolutional neural networks (CNNs) have also been explored for log anomaly detection, particularly when the relationships between log entries are not sequential but involve detecting specific critical events. Lu et al. [19] demonstrated that a one-dimensional CNN can outperform general multi-layer perceptrons (MLP) and LSTM-based models for Hadoop Distributed File System (HDFS) log analysis, providing a faster and computationally efficient alternative. Fotiadou et al. [20] compared LSTM-based and 1D-CNN architectures for log event classification and network traffic anomaly detection. Both architectures demonstrated similar performance.

With the advent of attention mechanism, Transformer-based architectures have revolutionized many NLP tasks, including log analysis. Several recent studies have adopted transformer models to classify logs or detect anomalies. Lee et al. [28] proposed using BERT [29] for unsupervised log anomaly detection, leveraging the model's masked language modeling (MLM) pretraining objective to detect anomalies based on the probability distributions of masked log tokens. They operate on the assumption that the context of a normal system log significantly differs from that of an abnormal one. Based on this, a normal log is expected to have a low prediction error and a high probability for masked tokens, while an abnormal log is likely to show a higher error and a more uniform probability distribution, enabling the identification of anomalies.

Further expanding on this approach, Almodovar et al. [11] presented LogFiT, a finetuned BERT model specifically for log anomaly detection. By training on normal logs with a masked sentence prediction objective, LogFiT effectively distinguishes anomalous logs from normal ones based on top-k token prediction accuracy. Other transformer models, such as LogBERT [30] and Cross-attention-based Transformers (CCT) [31], have also been proposed, emphasizing the growing relevance of self-attention and large language models (LLMs) in log analysis.

While general log anomaly detection and classification are well-studied, specific research focusing on telecommunications logs remains sparse. However, as telecommunications networks generate vast amounts of log data across various layers of the protocol stack, identifying relevant logging points that facilitates defect diagnosis becomes crucial. Ulku et al. [14] provided one of the few comprehensive studies on log classification within the telecommunications industry, demonstrating the applicability of ML approaches for detecting anomalies in network operations. Log analysis in the context of multi-label classification (MLC) is not explored extensively in literature. Dhakal K. [38] proposed a MLC approach for anomaly detection from real world system logs generated in industrial settings. They train custom word and sub-word based tokenizers and obtain token embeddings using static embedding approaches such as Word2Vec as well as contextual embedding methods based on LLMs such as pretrained BERT and DistilBERT models. These embeddings are then sent through a separate LSTM-based classifier. They achieve a high accuracy of 99% and a macro F1-score of 0.94 for multi-label classification of logs generated from patient monitoring devices.

In this study, we propose a ground-up approach for multi-label classification of complex log files for defect diagnosis in telecommunications industry. We pretrain and finetune LLMs end-to-end for log text understanding and logging point detection, respectively. Additionally, we benchmark our end-to-end finetuning approach of LLMs for MLC against extracting LLM-based embeddings and applying separate classifiers, similar to the technique proposed in [38]. For our use case, directly finetuning LLMs for MLC demonstrated higher performance.

III. Methods

Fig. 1 demonstrates the full-scale workflow of log mask prediction using LLMs, encompassing four integral components: data extraction and processing, pre-training, fine-tuning, and inference. This section provides a comprehensive overview of the methodologies employed in the data processing and pre-training phases of the entire pipeline.

A. Data Extraction and Processing

The essential interactions between the customer and service provider's triage and engineering teams for identifying the source of the issues and then resolving the defects, are raised, conducted, and systematically documented under a defect ticketing system. Most often, this record-keeping and communication take place on platforms such as Microsoft Dynamics (MSD) and ClearQuest (CQ) which are widely used for tracking and managing such issues. First, we navigate to the case stores of these platforms and extract all relevant cases. Each case is tagged with a unique sample identifier, and contains the log attachment generated through the default LBs along with other information as shown in Fig. 2. We then employ a parser script to extract related information, such as the network tester configuration (which we refer to as *config*), and any errors, warnings, asserts and other critical information (e.g., captured via keywords such as fail, exceed, invalid, violate, discard, timed out, etc.) recorded in the log file. We use the term *err_warn* to describe information that signifies errors, warnings, or failures. For every defect case, we then retrieve the associated final set of LBs recommended by engineers before the case was closed. This set serves as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

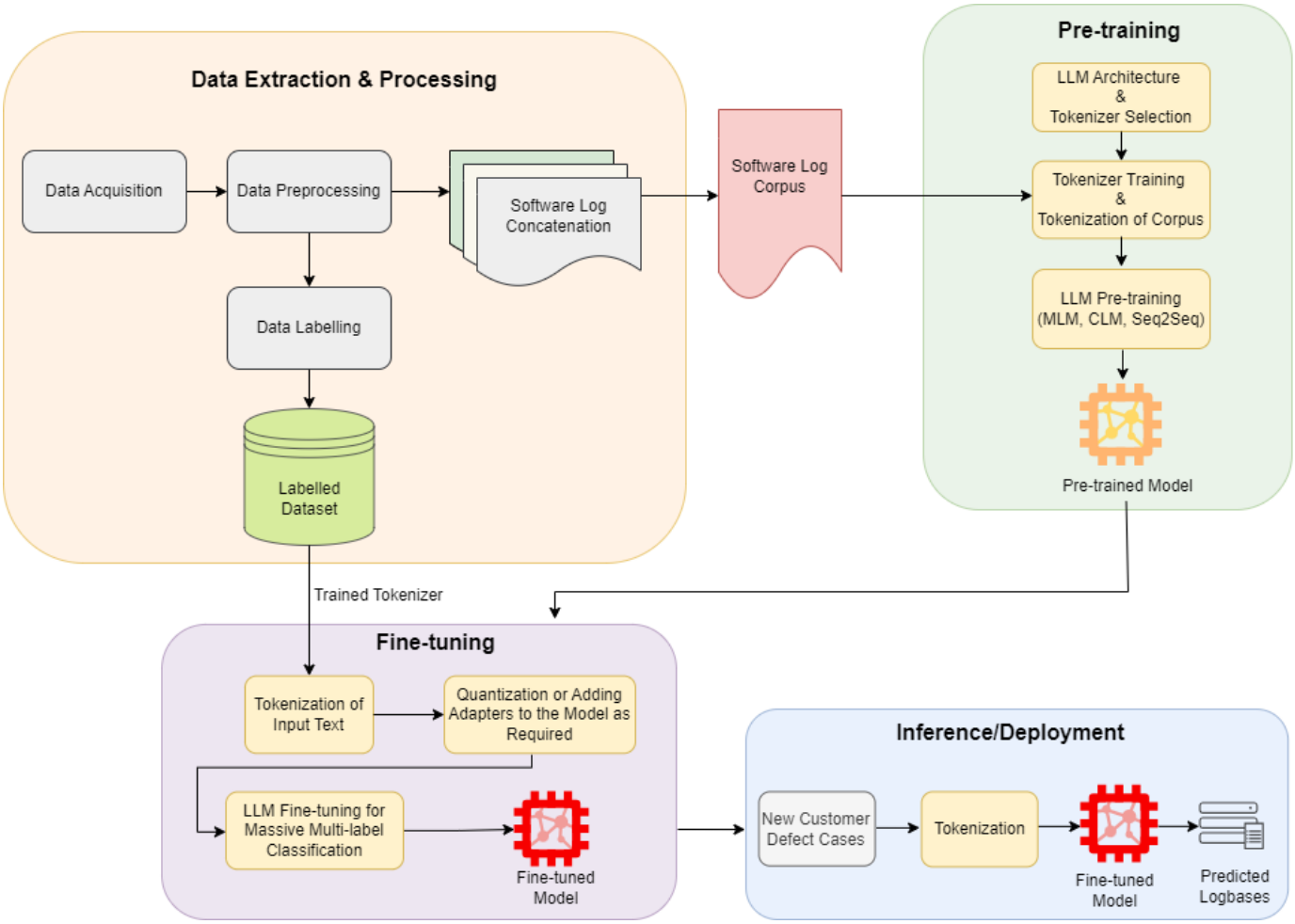


Fig. 1. Complete workflow of LLM-based logmask prediction.

the target label, while both the *config* and *err_warn* text data features are used as inputs to our classification models.

We then detect and remove outlier samples from the database. All relevant lines of text in our software logs, namely, the indications (tagged with `I:`) that contain data transmitted or received by the network and confirmations (tagged with `C:`) that record the execution status of system commands, along with *config* and *err_warn* are captured and the character length of this combined text is calculated for every sample. Subsequently, we employ the interquartile range (IQR) to detect outliers, particularly those featuring excessively long text sequences. IQR is defined and computed as the difference between the 25th and the 75th percentiles of a data distribution, the character length distribution of samples in this case. Detected outliers are then removed, ensuring that the majority of text samples remain within the typical context window of LLMs. Our final dataset contains 7958 industrial data samples.

Next, we proceed by identifying and removing specific custom stop words, time stamps and some very long jargon words in our logs that are generally irrelevant for defect resolution. The actual values of numbers, hexadecimal strings and IP addresses in our logs are typically unrelated for root cause analysis. Nonetheless, including a placeholder to signify their presence can be beneficial for maintaining the syntactic coherence of the logs. As a result, we use special tokens like `<num>`, `<hex>`, and `<ipaddr>` to mask such text. It should be noted that only distinctive numbers are masked with `<num>` token, and not the numbers in some alphanumeric words, as they may carry information. The objective of our data pre-processing phase is twofold: to construct an appropriately formatted text corpus suitable for LLM pre-training and to extract relevant text snippets and target labels for establishing a labelled dataset for the downstream log classification task. Before creating the corpus, we replace the newline characters (`\n`) with `<newline>` and demarcate the end of each software log with an `<endsample>` token. This procedure enables the generation of an extensive and systematically structured text corpus through the concatenation of pre-processed individual software logs. Furthermore, deduplication is performed to remove recurring identical lines of text from the software log corpus.

After performing data cleaning and incorporating the special tokens as previously detailed, we proceed to construct our labeled dataset. This dataset primarily consists of *config* and *err_warn* text columns along with the LBs extracted at the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

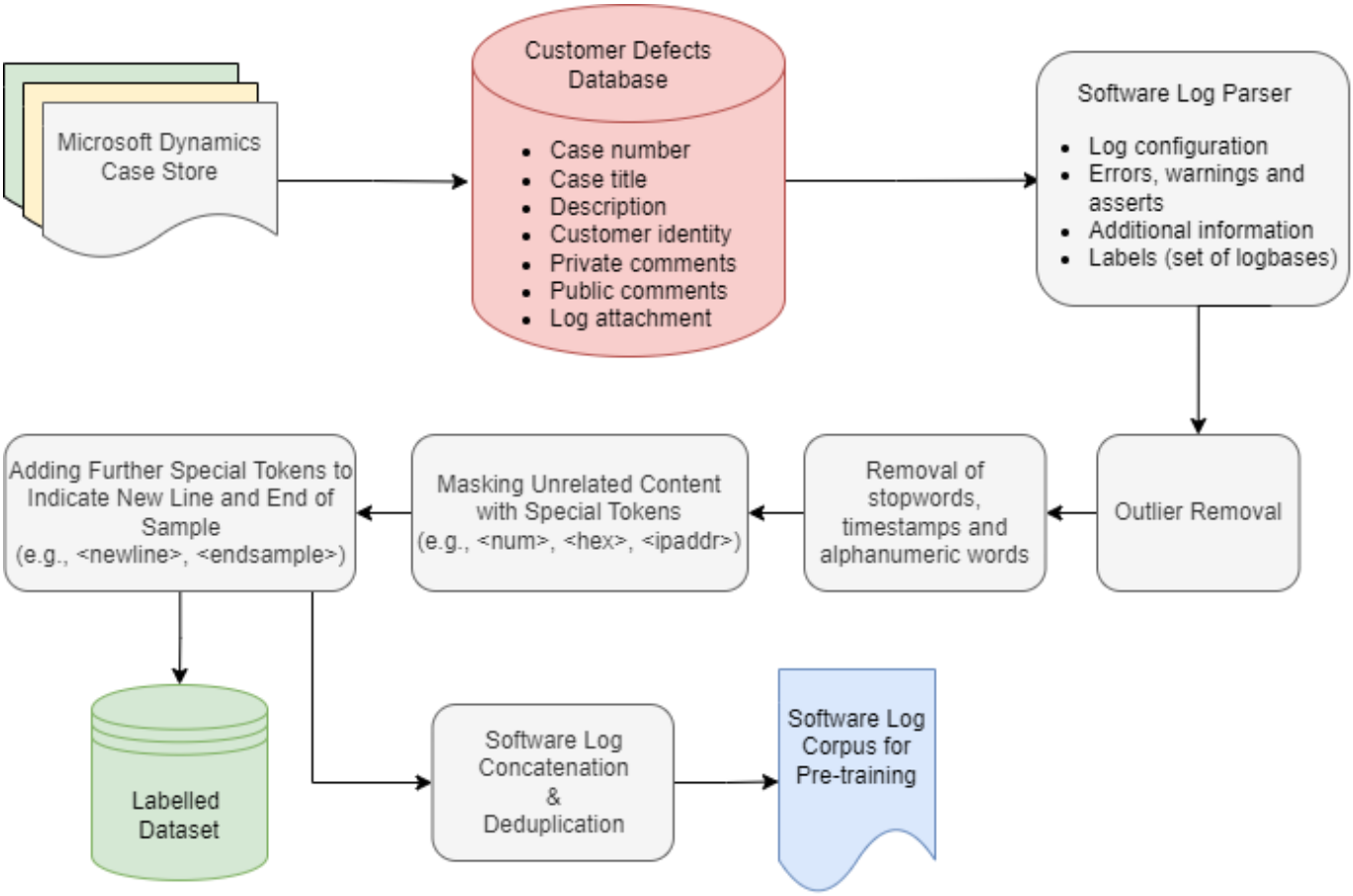


Fig. 2. Data extraction and processing pipeline.

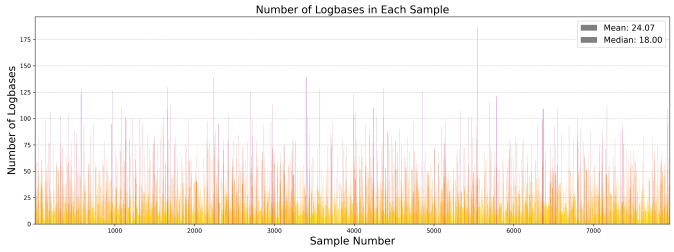


Fig. 3. Number of target log bases in every sample of the labelled dataset.

parsing phase for every customer defect case. However, we observed that the total number of unique LBs in the target label column is too large (> 1500), given the size of our dataset. Moreover, historical data suggests that many LBs rarely need to be enabled to identify defects. Consequently, we consider only those LBs that have appeared over 100 times across all defect cases reported in the past three years. This results in a total of 257 unique LBs available for selection. Fig. 3 shows the number of LBs associated with every defect case. As observed, debugging different issues requires enabling diverse number of LBs, indicating significant variability.

B. Pre-training

Once a sufficiently sized and properly structured software log corpus is established, we progress to the LLM pre-training stage. One of the primary factors to consider at this point is the selection of the LLM architecture, in conjunction with its associated tokenizer. Given that our corpus contains approximately 4 billion tokens, the choice of model size should align accordingly [45]. Here, we opt for RoBERTa, BigBird and Flan-T5 architectures for domain-specific pre-training. In particular, RoBERTa and BigBird architectures are pre-trained from scratch, including their tokenizers, while Flan-T5 is domain-adapted starting from publicly released `flan-t5-large` checkpoint. LLMs designed for specific objectives or trained on heterogeneous morphological text corpora may require different tokenizers that capture relevant linguistic features more effectively. Below we delve into the details of tokenizer training and the creation of domain-specific vocabularies.

Tokenization is the process of converting raw text into a numerical representation (i.e., a sequence of tokens) that models can interpret and learn from. Tokenizers play a pivotal role in constructing the vocabulary of a model—a finite set of tokens that the model recognizes. This vocabulary includes words, subwords, or characters, depending on the granularity of the chosen tokenization technique. The choice

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

of vocabulary size and tokenization strategy directly impacts the model's ability to learn and generalize across different text structures. Subword tokenization is particularly popular because it strikes a good balance between representing common words as they are and decomposing less common words into understandable sub-units. As the name implies, subwords are typically more granular than whole words but larger than individual characters. This is an effective strategy in handling morphologically rich text formats, out-of-vocabulary words and optimising the model's vocabulary size for computational efficiency. Two tokenizing strategies that we discuss here - byte pair encoding (BPE) and SentencePiece, both belong to the subword tokenization category. Readers are directed to the original studies for more details on BPE [46] and SentencePiece [48] algorithms. Among the LLMs examined in this study, BigBird and Flan-T5 employ the SentencePiece tokenizer, while RoBERTa uses the BPE tokenizer.

We then proceed in training these tokenizers specifically on our log corpus to better represent and utilize the inherent semantic information within these logs. Following empirical experimentation with various vocabulary sizes, we determined that a vocabulary size of approximately 1600 tokens is most suitable for our specific application. This is because the number of unique space-separated words in our logs is not as high compared to a natural language. This conclusion was reinforced by telecom experts who, upon examining the tokens in the dictionary, confirmed their relevance and applicability. Once an appropriate vocabulary size and a set of special tokens are identified as mentioned previously, we train BPE and SentencePiece tokenizers independently. While it is possible to train tokenizers using only a portion of the corpus, particularly with extremely large corpora, here, we used the full content of our corpus for tokenizer training owing to its relatively smaller size. This process results in the creation of a vocabulary file for each tokenizer, mapping every token to a unique index.

The software log corpus is tokenized accordingly, enabling the launch of LLMs. We train RoBERTa and BigBird architectures, each consisting of 6 hidden blocks and 12 attention heads, with a masked language modelling (MLM) objective. MLM involves randomly masking out some of the tokens in the input text and then training the model to predict these masked tokens based solely on their context. This allows the model learn a deep understanding of software log structure and context. Here, we mask about 30% of total tokens and train the models for five epochs. Fig. 4 shows the attention matrices of the pre-trained RoBERTa model for a given input text sample. As observed, the model calculates appropriate attention weights for different tokens based on the context, helping to construct a learned embedding of a given piece of text.

Flan-T5 is trained with a warm start for domain adaptation by tokenizing the corpus with its own pre-trained tokenizer. Training our own tokenizer is infeasible here because the embedding matrix of Flan-T5 checkpoint consists of learned embedding vectors for tokens in its original tokenizer with a fixed vocabulary size. Nevertheless, we add our special tokens to the pre-trained tokenizer of Flan-T5 and extend the model embedding matrix to accomodate these new tokens. Because the `flan-t5-large` checkpoint is relatively large (780M parameters), and with the hope of preserving the original checkpoint weights, we employ LoRA method for domain-specific training of Flan-T5. LoRA focuses on updating a small set of trainable parameters that modify the original weights in a low-rank subspace, rather than updating all the parameters of the model directly. Flan-T5 is an encoder-decoder architecture and we train it with a sequence-to-sequence objective where the target sequences are created by shifting the input sequences by k ($\geq$1) number of tokens. Similar to before, Flan-T5 is trained for five epochs. All pre-trained models along with their tokenizers are stored for further finetuning and/or text embedding extraction. Note that it is crucial to maintain consistency in the use of the tokenizer throughout the entire process. Once the tokenizer is trained and a vocabulary is established, it is essential to use the same tokenizer consistently during the LLM pre-training, finetuning, and in the inference pipeline.

IV. Results and Discussion

We explore the use case of telecom log classification via two avenues: 1) E2E finetuning of pre-trained LLMs for multi-label classification. 2) Independent binary classification to detect the presence of individual LBs by applying classical ML models on log text embeddings extracted from pre-trained LLMs. In both cases, our emphasis lies in accurately identifying a relevant set of LBs from the concatenated `err_warn` and `config` input text features. The character length distributions of `config` and `err_warn` across the dataset are shown in Fig. 5.

A. E2E Finetuning

We finetune the pre-trained RoBERTa and BigBird model for MLC on our labelled dataset. MLC approach faces several challenges, including heavily skewed distribution of log bases as depicted in Fig. 6, limitations in the available data, and the overall high number of log bases. As depicted in the finetuning block of Fig. 1, the input text is first tokenized with the respective tokenizer. Initial dataset containing 7958 entries is split into 80% training and 20% test sets. Both models are trained on the training set for up to 400 epochs and their MLC performance is evaluated on the test set. Model quantization or the utilization of LoRA was not required in our computing infrastructure for RoBERTa and BigBird finetuning. Likewise, all model weights were updated during the finetuning stage. Table I summarizes the performance metrics for RoBERTa and BigBird on the test set. Notably, both models display similar performance, with BigBird performing marginally better. It is worth noting that the performance metrics reported in Table I refer to the micro average, where the total true positives, false negatives and false positives across all classes are considered when calculating the metrics.

In Fig. 7, we display the the distribution of F1-score per log base. As observed, there exists significant variation in the F1-score per LB. This may be attributed to the largely skewed distribution of LBs in our dataset. More prominent LBs are easily detectable whereas those with lower frequency of occurrence are generally associated with a lower F1-score.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

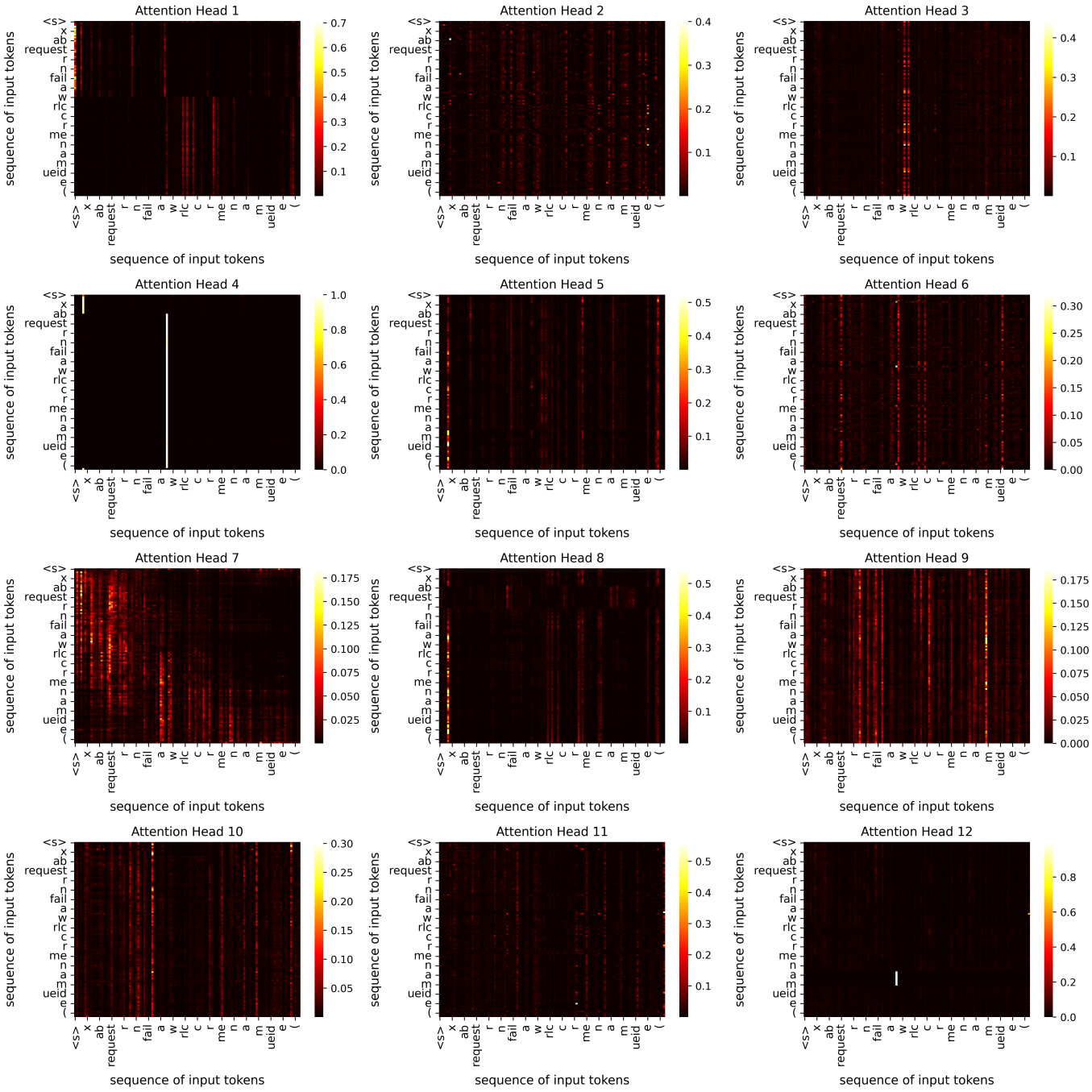


Fig. 4. Visualization of the attention matrices of hidden layer block #2 of pretrained RoBERTa model for a given text sample of length 425 tokens. The color indicates the attention score between the two associated tokens. Only a subset of tokens are shown at the corresponding tick index.

While the final performance metrics are similar in RoBERTa and BigBird, their training progressions exhibit unique characteristics, as illustrated in Fig. 8. BigBird enters within 5% of its optimal F1-score in about 60 epochs whereas RoBERTa takes about 120 epochs to accomplish the same level of performance. It appears that the training progress of RoBERTa is more stable than that of BigBird. Moreover, finetuning the BigBird model on our infrastructure requires approximately 13 hours, significantly longer than the training time of RoBERTa, which takes just under an hour. This discrepancy arises due to the large context window size of BigBird, leading to a quadratic increase in computational complexity. Nevertheless, once trained, both models are capable of performing inference without the requirement of graphics processing units (GPUs) and are deployable on a typical personal computer.

There may be several possible reasons contributing to the difficulty in achieving higher performance in the present downstream task. Mainly, our labels extracted from historic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

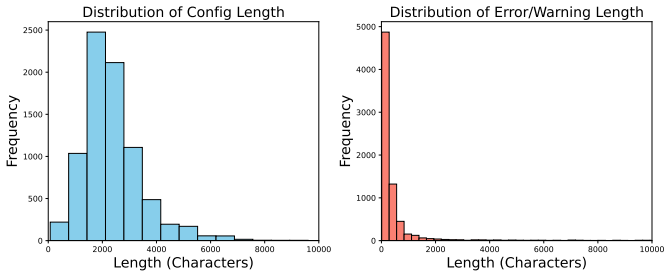


Fig. 5. Distribution of the character length of *configuration* and *errors and warnings* features.

Fig. 6. Logbase distribution across dataset. Heavily skewed distribution of logbases makes the multi-label classification more challenging.

data might reflect human bias - as the selection of LBs may differ from engineer to engineer. Also, the engineers are tasked to find the necessary set of LBs, which in most cases contain additional LBs that may not be necessary for locating the defects. Therefore, the target LBs are not consistent in our dataset, and it is extremely difficult to rectify this without manually going through each defect case with the help of a domain expert. Furthermore, the sheer number of available classes makes the MLC problem more challenging. As such, we believe that our finetuned LLMs remain highly useful, especially in generating an initial, reasonably accurate set of log bases. This aids in minimizing turnaround times and iterative communications between service providers and clients for defect identification and troubleshooting.

B. Independent Binary Classification

In this approach, we aim to mitigate the impact of imbalanced log base distribution on the classification performance. To achieve this, we establish a balanced database for each log base by randomly selecting negative class samples to match the number of positive ones. Here, the positive samples indicate those containing the LB of interest. Consequently, individual binary classifiers are trained for each LB to ascertain its relevance within a given software log. Variable-length text sequences need to be transformed into fixed-size representations for compatibility with traditional ML algorithms. We extract fixed-length embeddings of text samples in our labelled dataset using the pre-trained LLMs: RoBERTa, BigBird and Flan-T5. Specifically, the `err_warn` and `config` text are concatenated and passed through the pre-trained models and token embeddings are capture from the final hidden layer. These embeddings are then aggregated using a mean-pooling function to retrieve a single vector representation for each sample.Fig. 9 depicts the overlapping sliding window approach of extracting embeddings of text samples longer than the context length of the model that we proposed in our previous study [47]. Such a strategy is required especially when the length of the text sequence exceeds the maximum sequence length of the model. Concretely, consider a long text document with a total length of L tokens and a context window of l_c tokens, where l_c is smaller than L. Using an adaptable overlapping window of size w tokens, where w is less than l_c, the number of overlapping chunks required to cover the entire document, denoted by M, is given by the following formula:

$$M = \frac{L - w}{l_c - w} \tag{1}$$

In this approach, the global document embedding, E_g, is calculated as follows:

$$E_g = \frac{1}{M}\frac{1}{l_c}\sum_{k=1}^{M}\sum_{i=1}^{l_c} TE_{k,i} \tag{2}$$

Here, $TE_{k,i}$ represents the token embedding for the i^{th} token in the k^{th} chunk. These token embeddings are extracted from the last hidden layer of the model, with each embedding having a dimension d_{TE}, which corresponds to the number of units in the model's last hidden state. To compute the chunk embedding, a mean-pooling operation is applied across all token embeddings in a chunk, and this process is repeated across all chunks to obtain the final document embedding.

Model	Context Window	Precision	Recall	F1-score	Hamming Distance
RoBERTa	512	0.583	0.46	0.514	0.08
BigBird	4096	0.593	0.455	**0.515**	0.079

TABLE I
MULTI-LABEL CLASSIFICATION METRICS (MICRO AVERAGE) FOR THE TEST SET.

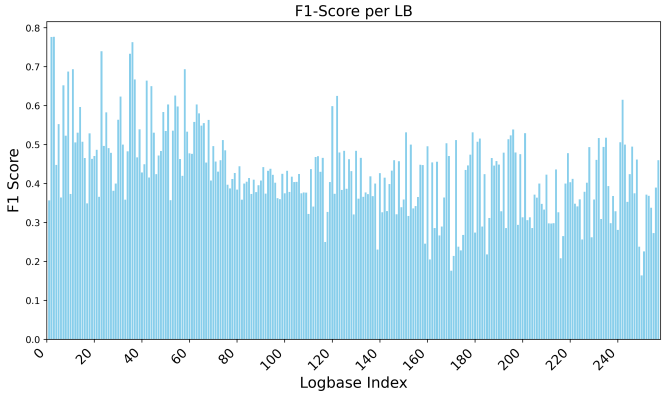


Fig. 7. F1-score per LB on the test set for BigBird.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

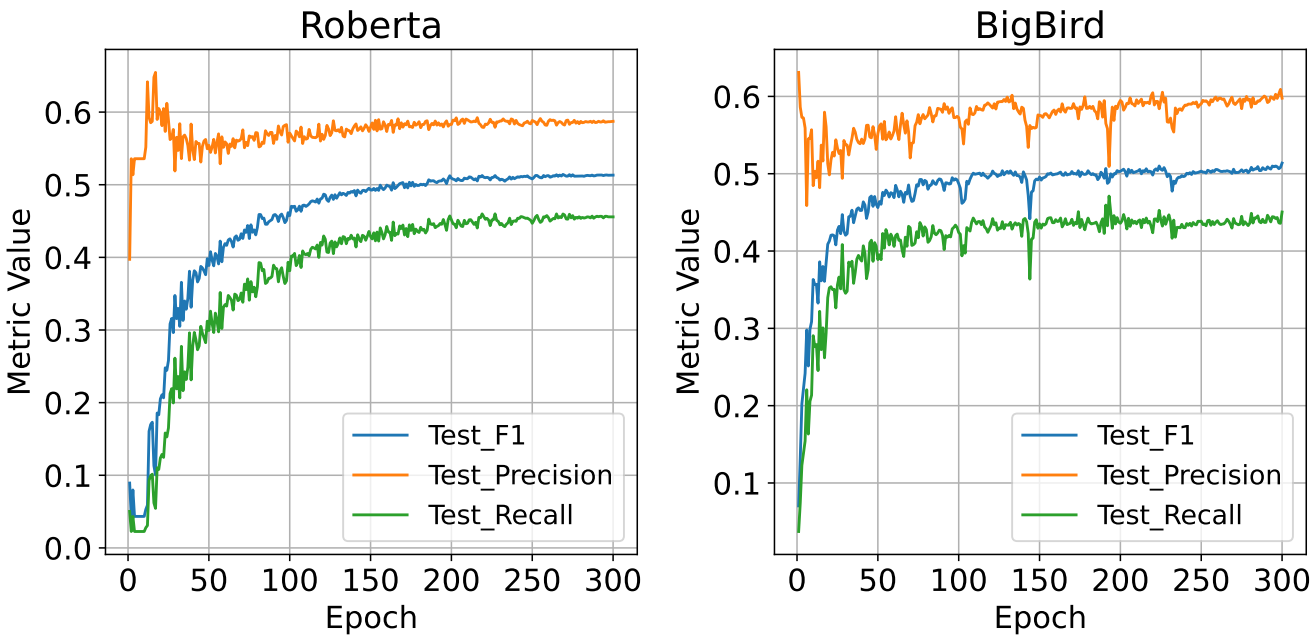


Fig. 8. Progression of performance metrics for the test set as (A) RoBERTa and (B) BigBird models are trained.

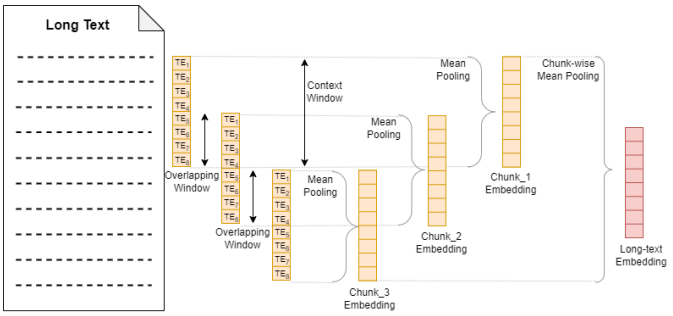


Fig. 9. Overlapping sliding window approach to extract embeddings of long documents using LLMs. TE_i refers to the token embedding of i^{th} token at the final hidden layer of the LLM. Token embedding dimension typically ranges from 512 to 4096 for larger models.
[47]

While performing this pooling, the attention mask is taken into account, ensuring that padding tokens are ignored. The text chunks overlap by w tokens, allowing some context and information from previous windows to be carried forward as the model processes the document.

With these embeddings, we then create a bespoke dataset for each log base as specified above. Subsequently, we train separate classical ML classifiers on an 80% randomly split subset of this dataset and assess model performance on the remaining 20% held-out test set. Models such as XGBoost, Random Forest, Gaussian Process and Extra Trees are investigated in this section. Please refer to the Experimental Details section for more details on hyperparameter optimization of ML models. Additionally, we employ TFIDF-based embeddings as a baseline approach and compare the performance of ML models on these embeddings against those using LLM embeddings.

Table II presents the binary classification outcomes achieved using various LLM embeddings and classical ML models. The performance metrics are averaged across all LBs. Notably, BigBird embeddings coupled with Gaussian Process classifiers demonstrate the highest performance, closely followed by ML models paired with Flan-T5 embeddings. Across all experiments, consistent training and test datasets are utilized, with fixed ML model states. Therefore, we may conclude that the superior quality of LLM embeddings is what contributes to the improved performance. Interestingly, our TFIDF baseline approach yields results comparable to LLM-based methods. This may suggest that the presence of particular keywords (e.g., error or assert indicators) may be more relevant in enabling certain LBs, rather than the actual semantics of software logs. Nonetheless, our results suggest that the LB classification remains a challenging task, despite employing datasets with balanced class distributions.

In the final phase of our analysis, we focus on obtaining a predicted set of LBs for each entry in a newly compiled dataset, comprising 189 recently resolved defect cases previously unseen by our models. This objective is achieved by deploying our trained binary classifiers, each dedicated to assessing the applicability of a specific LB to the software log samples under consideration. Through this approach, we determine the relevance of individual LBs and acquire a set of predicted LBs for every sample in the new dataset. This process, however, culminated in suboptimal outcomes, as evidenced by a micro F1-score of 0.214. A notably low precision rate of 0.13 was observed, primarily attributed to an elevated incidence of false positives. This phenomenon

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

stems from our approach to artificially balance the dataset for the purpose of training binary classifiers. By altering the dataset to simulate a uniform distribution of LBs—a scenario that contrasts with the actual, highly skewed distribution of LBs in practice—we inadvertently influenced the model's tendency to make predictions based on real label distribution. Consequently, this led to a frequent overprediction of rare LBs, thereby inflating the rate of false positives. Therefore, based on our findings, we assert that E2E finetuning of LLMs for multi-label classification represents the most promising strategy for *log mask* prediction.

V. Experimental Details

The experiments conducted in this research were performed on an Nvidia DGX server with 8xA100 80GB GPUs. We utilized the KubeFlow user interface (UI) for managing ML workflows, that leverages Kubernetes as its underlying platform for orchestration of hardware resources and efficient management of containerized applications across the available hardware. MLFlow [50], coupled with the KubeFlow UI was used for experiment tracking.

LLMs investigated in this research were adapted from the Hugging Face ecosystem [51]. In the pre-training stage, RoBERTa and BigBird were both trained for 5 epochs, with a batch size of 64 and 8 respectively. The architectures of both models consisted of six hidden layer blocks. The rest of the hyperparameters remained consistent with default values as provided by the `transformers` package. RoBERTa took about 50 hours to (pre)-train on our hardware, whereas BigBird was trained in 72 hours. LoRA adapters, with a rank of 16 and a scaling factor of 32 were applied to the Flan-T5 model checkpoint for domain adaptation. Flan-T5 was domain adapted for 5 epcohs with a batch size of 8, which resulted in about 160 hours of training time. At the finetuning stage, both RoBERTa and BigBird were trained for up to 400 epochs, with an initial learning rate of 10^{-4} for multi-label classification. Micro average of MLC metrics were calculated using the `torchmetrics` package. Fine-tuning of RoBERTa and BigBird took approximately 1 hour and 13 hours, respectively.

All classical models examined in this work were adapted from the `scikit-learn` package. ML model hyperparameters were tuned employing a 5-fold cross validation strategy. Binary classification metrics are calculated using the `scikit-learn` package and averaged across LBs. The training of a single ML model, depending on the dataset size and model type, ranges from less than a second up to a minute. LLM sample embeddings were stored offline, allowing for swift execution of the series of ML classifiers.

VI. Conclusion

In this paper, we presented a comprehensive end-to-end workflow of pre-training LLMs ground up on an industrial software log corpus, followed by the finetuning of these models on a multi-label classification task. This downstream task is aimed at accurately identifying the appropriate set of log bases for defect resolution, significantly reducing the turnaround times and the resource consumption of both service providers and clients. Despite the formidable challenges posed by complex and voluminous log data with little resemblance to natural language, the imbalanced distribution of labels, and the presence of human bias in historical data — a scenario frequently encountered in industrial datasets — our approach demonstrates a promising avenue for leveraging AI to minimize manual intervention. This research lays the groundwork for future advancements in AI-driven diagnostics in telecommunications, underscoring the importance of systematic data curation and the infusion of domain-specific knowledge into AI models to enhance efficiency and accuracy in defect resolution.

Acknowledgment

We acknowledge the support of VIAVI Solutions Inc. for their provision of data, funding, and MLOps infrastructure, including GPUs, which contributed significantly to the completion of this project.

References

[1] Aljabri, Malak, Amal A. Alahmadi, Rami Mustafa A. Mohammad, Menna Aboulnour, Dorieh M. Alomari, and Sultan H. Almotiri. Classification of firewall log data using multiclass machine learning models. *Electronics* 11, no. 12 (2022): 1851.

[2] Cândido, Jeanderson, Maurício Aniche, and Arie van Deursen. "Log-based software monitoring: a systematic mapping study." *PeerJ Computer Science* 7 (2021): e489.

[3] Pirscoveanu, Radu S., Steven S. Hansen, Thor MT Larsen, Matija Stevanovic, Jens Myrup Pedersen, and Alexandre Czech. "Analysis of malware behavior: Type classification using machine learning." In 2015 International conference on cyber situational awareness, data analytics and assessment (CyberSA), pp. 1-7. IEEE, 2015.

[4] Catovic, A., Cartwright, C., Gebreyesus, Y.T. and Ferlin, S., 2021, May. Linnaeus: A highly reusable and adaptable ML based log classification pipeline. In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)* (pp. 11-18). IEEE.

[5] Maxwell, Kane, Mojtaba Rajabi, and Joan Esterle. "Automated classification of metamorphosed coal from geophysical log data using supervised machine learning techniques." *International Journal of Coal Geology* 214 (2019): 103284.

[6] Chen, Mike, Alice X. Zheng, Jim Lloyd, Michael I. Jordan, and Eric Brewer. "Failure diagnosis using decision trees." In International Conference on Autonomic Computing, 2004. Proceedings., pp. 36-43. IEEE, 2004.

[7] Liang, Yinglung, Yanyong Zhang, Hui Xiong, and Ramendra Sahoo. "Failure prediction in ibm bluegene/l event logs." In Seventh IEEE International Conference on Data Mining (ICDM 2007), pp. 583-588. IEEE, 2007.

[8] Du, Min, Feifei Li, Guineng Zheng, and Vivek Srikumar. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." In Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp. 1285-1298. 2017.

[9] Lin, Qingwei, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. "Log clustering based problem identification for online service systems." In Proceedings of the 38th International Conference on Software Engineering Companion, pp. 102-111. 2016.

[10] Xu, Wei, Ling Huang, Armando Fox, David Patterson, and Michael I. Jordan. "Detecting large-scale system problems by mining console logs." In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pp. 117-132. 2009.

[11] Almodovar, Crispin, Fariza Sabrina, Sarvnaz Karimi, and Salahuddin Azad. "LogFiT: Log anomaly detection using fine-tuned language models." IEEE Transactions on Network and Service Management (2024).

[12] Malik, Varun, Ruchi Mittal, Jaiteg Singh, and Pawan Kumar Chand. "Performance evaluation based on classification of web log data: a machine learning approach." In *Proceedings of International Conference on Communication, Circuits, and Systems*: IC3S 2020, pp. 363-369. Springer Singapore, 2021.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

LLM Embedding Model	Embedding Dimension	Binary Classifier	Average Metrics			
			F1-Score	Precision	Recall	AUROC
RoBERTa	768	XGBoost	0.629	0.633	0.633	0.622
		Random Forest	0.623	0.636	0.621	0.621
		Gaussian Process	**0.640**	0.619	0.671	0.616
BigBird	768	XGBoost	0.650	0.652	0.654	0.643
		Random Forest	0.646	0.662	0.638	0.648
		Gaussian Process	**0.661**	0.638	0.690	0.639
Flan-T5	1024	XGBoost	**0.647**	0.653	0.647	0.644
		Random Forest	0.646	0.659	0.640	0.646
		Gaussian Process	0.555	0.577	0.573	0.566
Classical Embedding Model		Binary Classifier	Average Metrics			
			F1-Score	Precision	Recall	AUROC
TFIDF		XGBoost	0.638	0.633	0.647	0.635
		Random Forest	0.638	0.642	0.640	0.640
		Extra Trees	**0.642**	0.645	0.644	0.644

TABLE II
INDEPENDENT BINARY CLASSIFICATION RESULTS. THE AVERAGE OF PERFORMANCE METRICS PER LB IS SHOWN HERE.

[13] Pitakrat, T., Grunert, J., Kabierschke, O., Keller, F. and Van Hoorn, A., 2015. A framework for system event classification and prediction by means of machine learning. *EAI Endorsed Transactions on Self-Adaptive Systems*, 1(3).

[14] Ülkü, O., Gözüaçik, N., Tanberk, S., Aydin, M.A. and Zaim, A.H., 2021, September. Software Log Classification in Telecommunication Industry. In *2021 6th International Conference on Computer Science and Engineering (UBMK)* (pp. 348-353). IEEE.

[15] Elmrabit, Nebrase, Feixiang Zhou, Fengyin Li, and Huiyu Zhou. "Evaluation of machine learning algorithms for anomaly detection." In 2020 *international conference on cyber security and protection of digital services (cyber security)*, pp. 1-8. IEEE, 2020.

[16] Bertero, Christophe, Matthieu Roy, Carla Sauvanaud, and Gilles Trédan. "Experience report: Log mining using natural language processing and application to anomaly detection." In *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 351-360. IEEE, 2017.

[17] Henriques, João, Filipe Caldeira, Tiago Cruz, and Paulo Simões. "Combining k-means and xgboost models for anomaly detection using log datasets." *Electronics* 9, no. 7 (2020): 1164.

[18] Catillo, Marta, Antonio Pecchia, and Umberto Villano. "AutoLog: Anomaly detection by deep autoencoding of system logs." *Expert Systems with Applications* 191 (2022): 116263.

[19] Lu, Siyang, Xiang Wei, Yandong Li, and Liqiang Wang. "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)*, pp. 151-158. IEEE, 2018.

[20] Fotiadou, Konstantina, Terpsichori-Helen Velivassaki, Artemis Voulkidis, Dimitrios Skias, Sofia Tsekeridou, and Theodore Zahariadis. "Network traffic anomaly detection via deep learning." Information 12, no. 5 (2021): 215.

[21] Yadav, Rakesh Bahadur, P. Santosh Kumar, and Sunita Vikrant Dhavale. "A survey on log anomaly detection using deep learning." In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pp. 1215-1220. IEEE, 2020.

[22] Nedelkoski, Sasho, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. "Self-attentive classification-based anomaly detection in unstructured logs." In 2020 *IEEE International Conference on Data Mining (ICDM)*, pp. 1196-1201. IEEE, 2020.

[23] Le, Van-Hoang, and Hongyu Zhang. "Log-based anomaly detection with deep learning: How far are we?." In *Proceedings of the 44th international conference on software engineering*, pp. 1356-1367. 2022.

[24] Du, Min, Feifei Li, Guineng Zheng, and Vivek Srikumar. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." In Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp. 1285-1298. 2017.

[25] Landauer, Max, Sebastian Onder, Florian Skopik, and Markus Wurzenberger. "Deep learning for anomaly detection in log data: A survey." *Machine Learning with Applications* 12 (2023): 100470.

[26] Huang, Shaohan, Yi Liu, Carol Fung, Rong He, Yining Zhao, Hailong Yang, and Zhongzhi Luan. "Hitanomaly: Hierarchical transformers for anomaly detection in system log." *IEEE transactions on network and service management* 17, no. 4 (2020): 2064-2076.

[27] Almodovar, Crispin, Fariza Sabrina, Sarvnaz Karimi, and Salahuddin Azad. "LogFiT: Log Anomaly Detection using Fine-Tuned Language Models." IEEE Transactions on Network and Service Management (2024).

[28] Lee, Yukyung, Jina Kim, and Pilsung Kang. "Lanobert: System log anomaly detection based on bert masked language model." *Applied Soft Computing* 146 (2023): 110689.

[29] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[30] Guo, Haixuan, Shuhan Yuan, and Xintao Wu. "Logbert: Log anomaly detection via bert." In 2021 international joint conference on neural networks (IJCNN), pp. 1-8. IEEE, 2021.

[31] Larisch, René, Julien Vitay, and Fred H. Hamker. "Detecting anomalies in system logs with a compact convolutional transformer." IEEE Access (2023).

[32] Ott, Harold, Jasmin Bogatinovski, Alexander Acker, Sasho Nedelkoski, and Odej Kao. "Robust and transferable anomaly detection in log data using pre-trained language models." In *2021 IEEE/ACM international workshop on cloud intelligence (CloudIntelligence)*, pp. 19-24. IEEE, 2021.

[33] Chen, Song, and Hai Liao. "Bert-log: Anomaly detection for system logs based on pre-trained language model." *Applied Artificial Intelligence* 36, no. 1 (2022): 2145642.

[34] Farzad, Amir, and T. Aaron Gulliver. "Unsupervised log message anomaly detection." *ICT Express* 6, no. 3 (2020): 229-237.

[35] Meng, Weibin, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen et al. "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." In *IJCAI*, vol. 19, no. 7, pp. 4739-4745. 2019.

[36] Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[37] Zaheer, Manzil, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham et al. "Big bird: Transformers for longer sequences." *Advances in neural information processing systems* 33 (2020): 17283-17297.

[38] Dhakal, Kamal, 2023. Log Analysis and Anomaly Detection in Log Files with Natural Language Processing Techniques (Master's thesis). 2023.

[39] Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li et al. "Scaling instruction-finetuned language models." arXiv preprint arXiv:2210.11416 (2022).

[40] Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining." Bioinformatics 36, no. 4 (2020): 1234-1240.

[41] Wu, Shijie, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

Gideon Mann. "Bloomberggpt: A large language model for finance." arXiv preprint arXiv:2303.17564 (2023).
[42] Webersinke, Nicolas, Mathias Kraus, Julia Anna Bingler, and Markus Leippold. "Climatebert: A pretrained language model for climate-related text." arXiv preprint arXiv:2110.12010 (2021).
[43] Chalkidis, Ilias, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. "LEGAL-BERT: The muppets straight out of law school." arXiv preprint arXiv:2010.02559 (2020).
[44] Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).
[45] Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas et al. "Training compute-optimal large language models." arXiv preprint arXiv:2203.15556 (2022).
[46] Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." arXiv preprint arXiv:1508.07909 (2015).
[47] Achintha Ihalage, Sayed Taheri, Faris Muhammad, Hamed Al-Raweshidy, "Convolutional vs Large Language Models for Software Log Classification in Edge-Deployable Cellular Network Testing" arXiv preprint arXiv:2407.03759 (2024).
[48] Kudo, Taku, and John Richardson. "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." arXiv preprint arXiv:1808.06226 (2018).
[49] Singhal, Karan, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales et al. "Large language models encode clinical knowledge." Nature 620, no. 7972 (2023): 172-180.
[50] Zaharia, Matei, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching et al. "Accelerating the machine learning lifecycle with MLflow." IEEE Data Eng. Bull. 41, no. 4 (2018): 39-45.
[51] Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac et al. "Huggingface's transformers: State-of-the-art natural language processing." arXiv preprint arXiv:1910.03771 (2019).

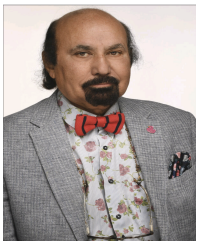
Sayed Taheri (S'09–M'12) received his PhD from University of London, Brunel, and his Master's degree (with Distinction) in Electrical Engineering - Signal Processing and Communications with AI from The University of Edinburgh, United Kingdom. He has over 15 years of experience in numerous large enterprises in the AI and data science fields. Since 2020, he has been working as a Research Fellow of VIAVI Solutions Inc. while holding a position as a Doctoral Researcher at Brunel University of London, working in cutting-edge AI/ML towards his PhD. In 2024, Sayed won the prestigious 'Vice Chancellor's Prize for Doctoral Research' from the University of London, which is awarded *yearly* to the highest-achieving PhD graduate across all fields and colleges. In the same year, he also won the 'Dean's Prize for Doctoral Research Innovation and Impact' from the College of Engineering, Brunel University of London. Sayed is the main inventor of 9 US patents as well as the author and co-author of two books, one book chapter, and several peer-reviewed research papers. He was a distinguished student in the Olympiad of Mathematics in 2004. He has won several scholarships and was the only winner of the USA's SPIE Education Scholarship Award from Asia in 2013. He has been a designated reviewer for several journals in IEEE and Springer. In January 2022, he was selected as the joint Technical Lead and SME in VIAVI's Centre of Excellence (COE) for AI. Sayed has been holding the position of AI/ML Solutions Architect and Team Lead at VIAVI since July 2022, where he leads cutting-edge research AI/ML initiatives and the productization of several PoCs in the Telecom domain.

Achintha Ihalage Dr. Achintha Ihalage received his PhD in Applied Machine Learning from Queen Mary University of London (QMUL) in 2022. His expertise lies in leveraging AI/ML within the antenna and telecommunication industry, with specific focus on transformer-based language models and graph neural networks. Between 2022/23, he worked as Post Doctoral Research Assistant at QMUL, contributing to multidisciplinary research efforts on AI-driven materials discovery for reconfigurable antenna design. He delivered undergraduate modules related to AI/ML as a Teaching Fellow at QMUL. He is the winner of the 2022 Mansel Davies Award for Dielectrics by the Institute of Physics, UK. Currently, he works in the telecommunications industry as an AI/ML Specialist, focusing on building generative AI systems with large language models (LLMs) for automated root cause analysis. He also leads the integration of AI into the Radio Access Network (RAN) infrastructure and RAN intelligent controller (RIC).

Prateek Mishra Prateek Mishra holds a Bachelor's degree in Electronics Engineering in 2007 and a Master's degree in Artificial Intelligence from King's College London in 2020. With over 16 years of experience in the telecommunications industry, he specializes in cutting-edge AI applications. Currently focused on advancing AI in telecoms, Prateek is dedicated to driving innovation and leveraging technology to optimize operations and enhance customer experiences.

Sean Coaker Sean Coaker received his MEng in Computing from Swansea University, UK, in 2022. With over a year of experience working in the software industry, Sean has developed a passion for developing cutting-edge software solutions.

Faris Muhammad Dr. Faris Muhammad received his MSc and PhD from Strathclyde University, a PGCE from MMU, and an MBA from Imperial College London. He is a Chartered Engineer (CEng) and a Fellow of the IET (F'IET). He has over 27 years of professional academic and industrial experience. Following 5 years of academic experience as senior lecturer, Dr. Muhammad progressed his career in the telecoms industry where he has developed expertise in software development, telecommunications, project management, and leadership. He has a distinguished track record in delivering complex projects, leading diverse teams, and driving successful change initiatives. He is also a Certified SAFe Agilist. Faris excels in Project and Programme Management, employing both Agile and Waterfall methodologies. He is skilled in Commercial and Customer Management, Wireless Technologies, Systems Engineering, and Leadership and Team Management. In addition to his engineering and project lifecycle management skills, Dr. Muhammad has directed large and complex development programs encompassing software, systems engineering, DevOps, testing, AI/ML, and, to a lesser extent, hardware. With a focus on customer-facing and commercial skills, he earned his MBA from Imperial College Business School. Dr. Muhammad has authored 1 Book, 4 Theses, 6 Patents, and over 50 Conference and Journal Papers.



Hamed Al-Raweshidy (M'95–SM'03) HAMED S. AL-RAWESHIDY (Senior Member, IEEE) received the Ph.D. degree from Strathclyde University, Glasgow, U.K., in 1991. He is currently a professor in communications engineering. He was with the Space and Astronomy Research Centre, Iraq, PerkinElmer, USA, Carl Zeiss, Germany, British Telecom, U.K., Oxford University, Manchester Metropolitan University, and Kent University. He is also the Group Leader of the Wireless Networks and Communications Group (WNCG) and the Director of PG studies (EEE) with Brunel University London, U.K. He is the Co-Director of the Intelligent Digital Economy and Society (IDEAS); the new research centre which is a part of the Institute of Digital Futures (IDF). He is a course director for the MSc Wireless Communication and Computer Networks. He is an Editor of the first book in Radio over Fibre Technologies for Mobile Communications Networks. He acts as a consultant and involved in projects with several companies and operators, such as Vodafone, U.K.; Ericsson, Sweden; Andrew, USA; NEC, Japan; Nokia, Finland; Siemens, Germany; Franc Telecom, France; Thales, U.K. and France; and Tekmar, Italy, Three, Samsung and VIAVI Solutions, USA and UK—actualizing several projects and publications with them. He is a Principal Investigator for several EPSRC

Citation information: DOI: 10.1109/TNSM.2025.3541384, IEEE Transactions on Network and Service Management

projects and European Project, such as MAGNET EU Project (IP) 2004-2008. He has published more than 500 journals and conference papers and his current research interests include 6G with AI and Quantum and the IoT with AI and Quantum. He is also an External Examiner for the Beijing University for Posts and Telecommunications (BUPT)—Queen Mary University of London. Further, he was an External Examiner for a number of the M.Sc. communications courses with Kings College London, from 2011 to 2016. He has also contributed to several white papers. Specifically, he was an Editor of Communication and Networking (White Paper), which has been utilised by the EU Commission for research. He has been invited to give presentations at the EU workshop and delivered two presentations at Networld2020, and being the Brunel Representative for NetWorld2020 and WWRF (for the last 15 years).