KnowLog: Knowledge Enhanced Pre-trained Language Model for Log Understanding

Lipeng Ma
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China
lpma21@m.fudan.edu.cn

Weidong Yang*†
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China
wdyang@fudan.edu.cn

Bo Xu†
School of Computer Science and Technology, Donghua University
Shanghai, China
xubo@dhu.edu.cn

Sihang Jiang
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China
tedsihangjiang@gmail.com

Ben Fei
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China
bfei21@m.fudan.edu.cn

Jiaqing Liang
School of Data Science
Fudan University
Shanghai, China
liangjiaqing@fudan.edu.cn

Mingjie Zhou
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China
mjzhou19@fudan.edu.cn

Yanghua Xiao
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China
shawyh@fudan.edu.cn

ABSTRACT

Logs as semi-structured text are rich in semantic information, making their comprehensive understanding crucial for automated log analysis. With the recent success of pre-trained language models in natural language processing, many studies have leveraged these models to understand logs. Despite their successes, existing pre-trained language models still suffer from three weaknesses. Firstly, these models fail to understand domain-specific terminology, especially abbreviations. Secondly, these models struggle to adequately capture the complete log context information. Thirdly, these models have difficulty in obtaining universal representations of different styles of the same logs. To address these challenges, we introduce KnowLog, a knowledge-enhanced pre-trained language model for log understanding. Specifically, to solve the previous two challenges, we exploit abbreviations and natural language descriptions of logs from public documentation as local and global knowledge, respectively, and leverage this knowledge by designing novel pre-training tasks for enhancing the model. To solve the last challenge, we design a contrastive learning-based pre-training task to obtain universal representations. We evaluate KnowLog by fine-tuning it on six different log understanding tasks. Extensive experiments demonstrate that KnowLog significantly enhances log understanding and achieves state-of-the-art results compared to existing pre-trained language models without knowledge enhancement. Moreover, we conduct additional experiments in transfer learning and low-resource scenarios, showcasing the substantial advantages of KnowLog. Our source code and detailed experimental data are available at **https://github.com/LeaperOvO/KnowLog**.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; • **Software and its engineering** → **Software maintenance tools**.

KEYWORDS

pre-trained language model, knowledge enhancement, log understanding

ACM Reference Format:
Lipeng Ma, Weidong Yang, Bo Xu, Sihang Jiang, Ben Fei, Jiaqing Liang, Mingjie Zhou, and Yanghua Xiao. 2024. KnowLog: Knowledge Enhanced Pre-trained Language Model for Log Understanding. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3623304

1 INTRODUCTION

With the growing complexity and scale of IT systems, the demand for high availability and stability has become increasingly critical, causing more challenges in maintaining complicated systems. These

*Also with Zhuhai Fudan Innovation Institute, Zhuhai, China.
†Corresponding authors.

ICSE '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0217-4/24/04…$15.00
https://doi.org/10.1145/3597503.3623304

systems often record the running status of the system or equipment by outputting logs. As a semi-structured text [18, 35, 61], logs consist of static templates and dynamic parameters, which contain a wealth of semantic information. Once a failure occurs, the most direct way for the maintainers is to analyze logs. However, due to the rapid growth of log size, log analysis by experienced experts becomes more challenging [62]. To solve this problem, automated log analysis has been proposed with the goal of automatically analyzing logs based on machine learning (ML) or deep learning (DL) techniques. Typical applications include log parsing [9, 27, 35, 58, 64], anomaly detection [11, 15, 26, 59, 61, 63], fault diagnosis [19, 62], failure prediction [34, 37, 38], etc. The key to these automated analysis approaches lies in the ability of machines to understand logs comprehensively.

Recently, many DL-based approaches have been adopted to understand logs. In comparison to ML-based approaches, DL-based approaches have garnered more interest due to their capacity to analyze data end-to-end, avoiding inflexible features and providing higher efficiency and adaptability [26, 61]. The DL-based approaches can be divided into two main categories. The first category involves traditional deep learning methods [7, 14, 17, 40, 60, 61], which utilize word embedding models to represent logs. These models are then fed into various neural network modules for training and prediction. The second category involves large pre-trained language models (PLMs) for log understanding. In recent years, pre-trained language models such as BERT [23], RoBERTa [33], GPT3 [3], BART [31] and T5 [48] have achieved remarkable results on various tasks in NLP, with significant performance improvement compared to traditional word embedding models. These models efficiently capture contextual information in natural language through self-supervised learning on a large natural language corpus and have established a *pretrain-then-finetune* paradigm. This paradigm involves achieving significant results by first pre-training on a large corpus and then fine-tuning on a small and specific dataset. Inspired by the success of these pre-trained models in NLP, many works [12, 16, 20, 25, 27, 32, 65] based on pre-trained language models have been proposed to understand logs, improving the performance of a variety of downstream tasks.

Although these pre-trained language models for log understanding have shown promising performance, they still suffer from three weaknesses.

First, these models fail to understand domain-specific terminology, especially abbreviations. As specific-domain texts, logs contain numerous domain-specific terminology and rarely appear in general natural language corpus. Consequently, most general pre-trained language models hardly understand the accurate semantics of domain terminology. In particular, abbreviations frequently appear in the log. Abbreviations usually contain the background knowledge of logs (for example, CM represents Configuration Management, and its appearance usually indicates the information related to configuration), ordinary people can only comprehend them according to their context, while experts are able to make inferences with relevant domain knowledge. Similarly, these existing pre-trained language models act just like ordinary people.

Second, these models struggle to adequately capture the complete log context information. Logs are typically concise and cannot provide enough background information, which presents a significant obstacle for models to gain an adequate understanding of them. For example, "NFM-4-PSS_VERSION_MISMATCH: PSS [dec] Code [dec]."[1] is a log for Cisco switches, and it is difficult to understand its content and intent without supplementary information.

Third, these models have difficulty in obtaining universal representations of different styles of the same logs. Learning universal representations can capture the commonality of logs and efficiently adapt to different tasks. Obviously, there are differences in log styles among different vendors, for example, the switches logs of Cisco and Huawei indicating the same *LinkDown* event are: "PORT-5-IF_DOWN_LINK_FAILURE: Interface [chars] is down (Link failure [chars]) [chars] [chars]"[1] and "IFNET/2/linkDown _active: The interface status changes. (ifName= [ifName], AdminStatus= [ifAdminStatus], OperStatus= [ifOperStatus], Reason=[Reason], mainIfname=[mainIf])"[2], we can find significant differences in structure and content.

To alleviate the above problems, we propose KnowLog, a knowledge-enhanced pre-trained language model for log understanding, which emphasizes leveraging domain knowledge to enhance pre-training on log corpus. KnowLog can comprehensively understand logs by exploiting abbreviations and natural language descriptions from documentation as local and global knowledge, respectively, which can be easily accessed from public documentation. In addition, it is consistent with human cognitive processes. When experts encounter an unfamiliar log, the expert likewise need to consult the documentation to understand the log.

In this paper, we design three novel pre-training tasks to effectively utilize domain knowledge to improve the understanding of logs. Specifically, to solve the first issue, we propose an abbreviation prediction (AP) task by exploiting the abbreviations information as local knowledge to enable the model to accurately understand these domain abbreviations. To solve the second issue, we propose a description discrimination (DD) task by exploiting the natural language description knowledge in the documentation as global knowledge to complement the underlying knowledge of logs and adequately capture the complete log context information. To solve the last issue, we propose a log contrastive learning (LCL) task to capture commonality in logs and make the representations of logs more universal.

After pre-training, we fine-tune and evaluate KnowLog on six different downstream tasks. The experimental results show that KnowLog obtains state-of-the-art results on all tasks compared to other pre-trained language models without knowledge enhancement, demonstrating that KnowLog can significantly improve log understanding with knowledge enhancement. Moreover, KnowLog is especially outstanding in transfer learning and low-resource scenarios.

In sum, our main contributions can be summarized as follows:

(1) To our best knowledge, we are the first to introduce domain knowledge to enhance the pre-trained language model, dubbed KnowLog, which is able to effectively improve the log understanding ability of the pre-trained model.

[1]https://www.cisco.com/c/en/us/td/docs/switches/datacenter/sw/system_messages/reference/sys_Book/sl_7K_MDS_m_to_r.html
[2]https://support.huawei.com/enterprise/en/doc/EDOC1100262538/8a642769

(2) We propose three pre-training tasks to improve the understanding of logs, where two tasks exploit local and global knowledge to enhance the model, and the third task is based on contrastive learning to obtain more universal representations.
(3) KnowLog achieves state-of-the-art results on all six different log understanding tasks, with significant performance improvement compared to the existing pre-trained language models without knowledge enhancement. In addition, KnowLog still has significant advantages in transfer learning and low-resource scenarios.

2 RELATED WORK

2.1 Pre-trained Language Models

Pre-trained language models (PLMs) have received much concern in recent years, especially after the emergence of ChatGPT, which follows the training of Instruct-GPT [43]. These PLMs based on Transformer structure [53] have revolutionized the NLP field. Compared to traditional DL-based models, these PLMs trained on self-supervised tasks aim to learn accurate latent representations of input text, and they have achieved state-of-the-art performance on a wide range of downstream tasks. Generally, these models can be divided into three categories: masked language models based on encoder structure (e.g., BERT [23] and RoBERTa [33]), autoregressive language models based on decoder structure (e.g., GPT3 [3]), and encoder-decoder models (e.g., BART [31] and T5 [48]). In addition, due to the lack of knowledge in PLMs when dealing with knowledge-driven tasks, many works have proposed to enhance PLMs with knowledge of linguistic [54], commonsense [10], factual [55], which focus on how to enhance the model's understanding of natural language. These models have a certain knowledge breadth and contain rich general cognitive knowledge with excellent natural language understanding capabilities [45], while the understanding of logs requires focusing on domain knowledge based on this general knowledge. This inspires us to leverage domain knowledge to improve the log understanding in the field of software engineering.

Apart from general pre-trained language models, many studies propose domain pre-trained language models, which pre-train on domain corpus based on BERT, such as BioBERT [28], SciBERT [2], LEGAL-BERT [5], BERTweet [42]. The success of these works inspires us to focus on domain data. Unlike these works, we focus on understanding logs, which are highly concise and do not conform to the syntax of natural language, making logs more challenging to understand. Apart from pre-training on domain corpus, we introduce additional domain knowledge as well as propose new pre-training tasks.

2.2 Log Understanding with Pre-trained Language Models

Inspired by the success of PLMs, many studies have proposed using pre-trained language models for log understanding. These works can be divided into two categories. One type of work is not fine-tuning the pre-trained model. This means that the parameters of the pre-trained model are not updated, and only obtain the representation vectors of logs and feed them into other deep models to complete specific tasks. HitAnomaly [20], SwissLog [32], NeuralLog [25] and ClusterLog [12] utilize the pre-trained model for log-based anomaly detection. Another type of work is to fine-tune the pre-trained model. A specific task model can be obtained by fine-tuning the entire pre-trained model on a small task dataset. NuLog [41], Logstamp [52] and LogPPT [27] are proposed to fine-tune the pre-trained model for log parsing. Also, Translog [16] and LogEncoder [47] perform fine-tuning based on log-based anomaly detection. Although these methods have achieved promising performance, we argue that these pre-trained language models for natural language still suffer from weaknesses that cannot comprehensively understand logs due to the gap between natural language and logs. In addition, UniLog [65] first proposes to pre-train on logs with the goal of reconstructing masked tokens and then fine-tune on multiple downstream tasks. We argue that the simple pre-training task of UniLog and the lack of domain knowledge cannot address our proposed challenges. Based on these works, it can be found that there is no in-depth research addressing the challenges of log understanding. In this paper, we propose a new perspective to enhance the pre-training of logs with domain knowledge and design relevant pre-training tasks to improve log understanding.

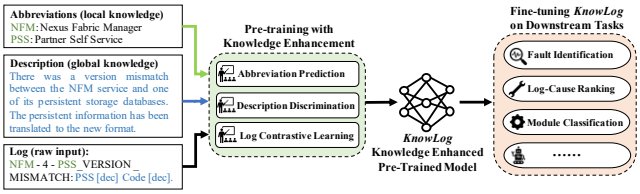

Figure 1: Conceptual overview of KnowLog.

3 METHOD

3.1 Overview

The conceptual overview of KnowLog is shown in Figure 1, KnowLog follows a pre-train and fine-tune paradigm, where it is pre-trained on log data with knowledge enhancement and fine-tuned on different downstream tasks. Specifically, we use logs as the pre-training corpus and introduce the abbreviations and the natural language descriptions of logs as local and global knowledge respectively to enhance the model via pre-training on three tasks. Finally, the knowledge-enhanced pre-trained language model for logs can be fine-tuned on various log analysis tasks. In the following sections, we describe the details of KnowLog, including input representation (Section 3.2), model architecture (Section 3.3), the pre-training tasks for training KnowLog (Section 3.4) and how to fine-tune it on downstream tasks (Section 3.5).

3.2 Input Representation

Logs as text strings must be preprocessed to be used as input to the model. We notice several interesting phenomena about tokens in logs, which differ from natural language. The tokens in logs contain not only some words in natural language but also some domain abbreviations such as OSPF and TCP. In addition, there are more tokens constructed by experts according to their own customs. This is close to the style of variables in the code [6], such as *TransferFail*

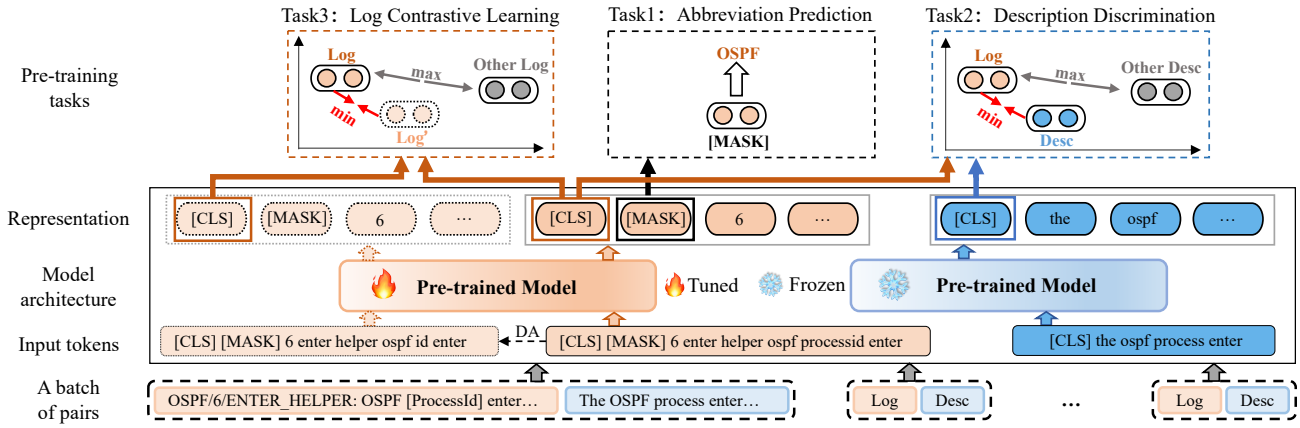

Figure 2: Framework of KnowLog pre-training. We pre-train on log corpus and take abbreviations and natural language descriptions as local and global knowledge. KnowLog performs three novel pre-training tasks: abbreviation prediction, description discrimination and log contrastive learning.

based on camel nomenclature. These phenomena pose a challenge for how to effectively split the logs.

To avoid the Out-Of-Vocabulary (OOV) problem and retain token information effectively, we use a tokenizer of the pre-trained language model for word segmentation. The tokenizer is a WordPiece [56] based method for transforming each sentence into a set of words and subwords, which maintains a vocabulary table by training on a vast corpus and is widely studied in language models [24, 46, 50]. For example, *TransferFail* is sliced into ["Transfer", "##Fail"], where "##" indicates that this token is a suffix of the previous word. However, there is also a drawback of this kind of segmentation, the integrity of the abbreviation will be destroyed. To avoid this issue, we propose to add abbreviations to the vocabulary of the tokenizer, which is described in detail in Section 3.4.2.

For a given piece of input text L, we use the tokenizer to obtain the sequence of tokens $\{[CLS], l_1, l_2, ..., l_n, [SEP]\}$ for input to the encoder. As a special token, $[CLS]$ is added to the front of the whole sequence, and its final hidden representation is considered to integrate the representation of the whole sequence.

3.3 Model Architecture

To enhance the understanding of logs through the corresponding natural language descriptions, we build Log-Description pairs $P = \{(l, d)\}$ as input and adopt a bi-encoder architecture in which two encoders process each type of data separately. Due to the powerful performance of pre-trained language models in natural language understanding, we use the pre-trained model based on Transformer encoder-only structure as the backbone to encode logs, such as BERT [23] and RoBERTa [33]. We will not review the ubiquitous Transformer structure in detail here. It is worth noting that our approach is not limited to a specific pre-trained model. That is, given a sequence of tokens after tokenization, an encoder outputs the hidden representation $h \in R^D$ of that sequence:

$$h_l = f(l; \theta), h_d = f(d; \theta^*), \tag{1}$$

f denotes the Transformer-based encoder, θ and θ^* represent parameters that can be learned and frozen, respectively, and D is the dimension of the hidden representation. Here, the encoder of natural language is only used for providing the description semantic representation as external knowledge, thus the parameters of this encoder do not need to be learned. In addition, to guarantee the representations of log and description in the same semantic space, two encoders are initialized with the same pre-trained model but do not share parameters.

3.4 Pre-Training KnowLog

3.4.1 Framework. Figure 2 illustrates the pre-training framework of KnowLog. KnowLog receives Log-Description pairs as input. First, we mask the abbreviations in logs and perform data augmentation (DA) on logs. Then, two pre-trained language models respectively encode logs and descriptions to obtain the corresponding representations. Finally, these representations are used in three novel pre-training tasks. The first task is abbreviation prediction, which leverages the abbreviations as local knowledge to understand these abbreviations. The second task is description discrimination, which leverages the natural language descriptions of logs as global knowledge to complement the underlying knowledge of logs and enable the model to adequately capture the complete log context information. The third task is log contrastive learning, which is based on contrastive learning to capture commonality in logs and obtain more universal representations of logs.

3.4.2 Abbreviation Prediction Task. Logs contain many abbreviations, which are usually domain terminology. These abbreviations are the basis for domain experts to understand logs, and we expect the model to acquire this capability as well. For example, in the log "SNMP/4/SNMP_BLACKLIST_UNBLOCK: User [UserOrIP] = [IPAddr_UserName] unblocked."[3], "SNMP" is an abbreviation, and its full name is Simple Network Management Protocol[4], and we can further understand that this log indicates events at the application layer. However, the current models cannot understand the abbreviations effectively for the reason that these abbreviations are domain

[3]https://support.huawei.com/enterprise/en/doc/EDOC1100137884/7d07f164
[4]https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

words, which rarely occur in natural language corpus. In addition, we also find that these abbreviations are split into subword tokens when logs are processed with the tokenizer, which breaks the semantics of the abbreviations and is detrimental to understanding logs. To capture the semantics of domain terminology in logs effectively, we exploit abbreviations information as local knowledge to enhance the model. First, we collect 1,711 abbreviations from the glossary in the public documentation of multiple vendors. Second, to retain the integrity of the abbreviations, we add all abbreviations to the tokenizer's vocabulary to ensure the abbreviations will not be sliced. Finally, we propose a novel abbreviation prediction (AP) task in pre-training to enhance the model for effectively understanding the abbreviations.

Specifically, given an input sequence of log $l = \{l_1, l_2, ..., l_n\}$, M abbreviations are included in l, each abbreviation $l_{ABBR} \in l$ (where $|l_{ABBR}| = M$) is randomly masked with a special $[MASK]$ symbol with a certain probability p, otherwise it remains unchanged. In this paper empirically we set p to 0.5 by default. The AP objective is to predict the original abbreviations which are masked out, and formulated as follows:

$$l^{masked} = \begin{cases} REPLACE(l_{ABBR}, [MASK]), & \text{if } p >= 0.5, \\ l, & otherwise. \end{cases} \quad (2)$$

$$\mathcal{L}_{AP}(\theta) = \sum_{i=1}^{M} -logp(l_i|l^{masked}) \quad (3)$$

3.4.3 Description Discrimination Task. Obviously, logs do not conform to the syntax of natural language and are highly concise, it is difficult to effectively capture the underlying information of logs. To better understand logs, we exploit natural language descriptions of logs as global knowledge to enhance the model. We expect the model to pull closer the distance between logs and the corresponding natural language descriptions in the semantic space, thus reducing the misunderstanding of log semantics.

To implement our idea, we propose a description discrimination (DD) pre-training task based on contrastive learning. Learning to distinguish whether the semantics of logs and descriptions are similar helps the model to better understand log semantics effectively. Conceptually, the aim of contrastive learning is to pull similar instances closer while pushing different instances farther apart. Specifically, given a batch of log and description pairs $\{(l_i, d_i)|i = 1, 2, ..., N\}$, we first separately encode them into hidden representations $\boldsymbol{h_l^i} = f(l_i; \theta)$, $\boldsymbol{h_d^i} = f(d_i; \theta^*)$. Then, we construct $(\boldsymbol{h_l^i}, \boldsymbol{h_d^i})$ as a positive pair and $\boldsymbol{h_l^i}$ with other descriptions within the same batch as negative pairs. To pull $\boldsymbol{h_l^i}$ and $\boldsymbol{h_d^i}$ closer, and push away $\boldsymbol{h_l^i}$ and $\boldsymbol{h_d^j}$, we follow the contrastive framework in [8], and the training objective of DD is:

$$\mathcal{L}_{DD}(\theta) = -log\frac{exp(sim(\boldsymbol{h_l^i}, \boldsymbol{h_d^i})/\tau)}{\sum_{j=1}^{N} exp(sim(\boldsymbol{h_l^i}, \boldsymbol{h_d^j})/\tau)}, \quad (4)$$

where τ is a temperature hyperparameter introduced by [57], which is set to 0.05 empirically in this paper, N is batch size and $sim(\boldsymbol{h_l^i}, \boldsymbol{h_d^i})$ is the cosine similarity.

3.4.4 Log Contrastive Learning Task. Logs as semi-structured texts composed of templates and variables, on the one hand, there are some differences in log templates conveying the same semantics on different vendors. On the other hand, the semantics of log instances with concrete parameters are essentially similar to log templates. The semantics should not be deviated by parameters when using the model directly to represent the log instances. Therefore, it is anticipated that the model should capture the commonality of logs and the semantic representations of logs should be more universal.

Inspired by the research on sentence representation [13] in NLP, we introduce a log contrastive learning (LCL) task. We employ data augmentation to generate additional positive samples of logs, and then we pull closer the semantic distance between logs and positive samples with contrastive learning. For data augmentation, we argue that parameter changes in the log template do not affect the log semantics. Thus, we perform synonymous rewriting to the parameter token in the log. For example, the "[dec]" in the log template as a parameter token in Figure 1 is modified to "[decimal]" for achieving an augmentation sample. Following Simcse [13], we still adopt contrastive learning to achieve this goal. Specifically, given a batch of log and description pairs $\{(l_i, d_i) \mid i = 1, 2, ..., N\}$, we perform data augmentation on the log l_i to get l'_i. Then we encode them into hidden representation using the same log encoder $\boldsymbol{h_l^i} = f(l_i; \theta)$, $\boldsymbol{h_l'^i} = f(l'_i; \theta)$. Further, $(\boldsymbol{h_l^i}, \boldsymbol{h_l'^i})$ as a positive pair and $\boldsymbol{h_l^i}$ with other logs within the same batch as negative pairs. Eventually, like the DD objective, the training objective of LCL is:

$$\mathcal{L}_{LCL}(\theta) = -log\frac{exp(sim(\boldsymbol{h_l^i}, \boldsymbol{h_l'^i})/\tau)}{\sum_{j=1}^{N} exp(sim(\boldsymbol{h_l^i}, \boldsymbol{h_l^j})/\tau)}, \quad (5)$$

where the hyperparameter is the same as in Equation 4.

Now we present the overall pre-training objective of KnowLog. To avoid catastrophic forgetting [39], we train three tasks together. Hence, the overall learning objectives are given below.

$$arg\min_{\theta} \mathcal{L}_{AP}(\theta) + \mathcal{L}_{DD}(\theta) + \mathcal{L}_{LCL}(\theta) \quad (6)$$

3.5 Fine-Tuning KnowLog

After pre-training, we fine-tune KnowLog on different log-related downstream tasks. We feed the input into the model as in the pre-training phase and use the representation of $[CLS]$ to represent the whole input. In the downstream tasks, we experiment with two categories of input tasks, where one type of input is a log-single task, such as fault phenomenon identification, and the other type is a log-pair task, such as log and possible cause ranking. For a single-log task, we pass the representation vector to a classifier. For log-pair task, following sentence-bert [49], we encode inputs independently to produce representation vectors $\boldsymbol{l}, \boldsymbol{d}$, so we concatenate $\boldsymbol{l}$ and $\boldsymbol{d}$ with the element-wise difference $\boldsymbol{\{l - d\}}$ and pass $\boldsymbol{[l; d; \{l - d\}]}$ to a classifier. For each task, we simply feed the input into the model, then all parameters are fine-tuned end-to-end, and finally, the fine-tuned KnowLog can be used to make inferences on the specific task. Details are given in the next section.

4 EXPERIMENTS

In this section, we first introduce the pre-training data (Section 4.1), baselines (Section 4.2) and model settings (Section 4.3). Then, we introduce six different downstream tasks (Section 4.4) and answer five research questions through experiments (Section 4.5). Finally,

we perform the ablation studies (Section 4.6) and qualitative analysis (Section 4.7) of KnowLog.

4.1 Pre-Training Dataset

In this paper, the log analysis scenario studied is oriented toward network devices, such as switches and routers. We train KnowLog with log templates rather than log instances that log with concrete parameters. Pre-training based on log templates has several advantages: First, log templates can be easily obtained from public documentation. Second, it avoids data privacy leakage [4, 21, 30]. Log instances usually record the resource information of the environment, and the model may unintentionally leak this privacy. Third, it can reduce the data governance burden. Log instances have a large number of repeated logs of the same template, making the corpus lack diversity. Some studies [1, 29] have shown that repetitive data leads to the model memorizing specific data making the model lack generalization. In our experiments, we also verify the generalization capability of the log template-based pre-trained model on log instances, demonstrating its effectiveness.

We collected 96,060 log templates and the corresponding natural language descriptions from Cisco[5], Huawei[6] and H3C[7] public product documentation for four products in switches, routers, security and WLAN. Additionally, we also collected 1,711 abbreviations from the documentation. Since log templates come from publicly available documentation, we find that there are different versions of the same product from one vendor, resulting in many identical log templates, and we filter the duplicate log templates. The final data statistics are shown in Table 1.

Table 1: Statistics of the dataset used for pre-training.

	Switches	Routers	Security	WLAN	All
Cisco	41,628	22,479	1,578	6,591	72,276
Huawei	6,418	4,980	3,737	1,001	16,136
H3C	2,171	2,364	1,852	1,261	7,648
All	50,217	29,823	7,167	8,853	96,060

4.2 Baselines

We categorize these baselines for log understanding into two groups: traditional deep-learning methods and pre-trained language models. Traditional deep learning models rely on static word embedding models like Glove [44] to acquire semantic representations of logs. These representations are then fed into various neural network models for log analysis. On the other hand, pre-trained language models are used to extract semantic information or directly fine-tuned on various tasks with powerful language comprehension. In our comparison, we evaluate KnowLog against two commonly used traditional deep-learning methods, two popular general pre-trained models, and a specific pre-trained model.

- **CNN.** Following [36], we adopt a Word2vec model [44] to learn the representation vectors of logs. Then these vectors input a convolutional neural network (CNN) to support downstream tasks.
- **Bi-LSTM (Attention).** Similar to CNN, following [61], we also adopt the Word2vec model [44] to represent logs. Then these representation vectors input an Attention-based Bi-LSTM model to support downstream tasks.
- **BERT.** BERT [23] is a pre-trained language model for natural language with the goal of masked language model (MLM). Recently many BERT-based log analysis works have been proposed, where BERT is used to encode the semantic meaning of logs. For a fair comparison with KnowLog, we also fine-tune the BERT on downstream tasks.
- **RoBERTa.** RoBERTa [33] is also a pre-trained language model for natural language, it improves the training method of BERT. We also fine-tune RoBERTa on downstream tasks.
- **UniLog.** UniLog [65] is a specific pre-trained model for logs, which is the first model to introduce pre-training on log corpus with the goal of reconstructing masked tokens and fine-tuning for different downstream tasks. UniLog verifies the effectiveness of log pre-training for log understanding and as a state-of-the-art model for log analysis. For fairness, we reproduce UniLog based on BERT and pre-train it on KnowLog's identical pre-training log corpus. The primary distinction between UniLog and KnowLog lies in the absence of domain knowledge enhancement in UniLog.

4.3 Parameter Settings

For pre-training, we use the encoder structure model (BERT-base-uncased and RoBERTa-base) as the backbone and BERT-base-uncased is mainly used for ablation studies and qualitative analysis. The optimizer KnowLog used is Adam with a learning rate of 2e-5, a weight decay of 0.01, learning rate warmup for 10,000 steps and linear decay of the learning rate after. The sequence length was limited to 512 tokens and we train for 50 epochs with a batch size of 16. For fine-tuning, the cross-entropy loss is employed as the loss function in the downstream tasks and we set epoch to 20 and 10 on log-single tasks and log-pair tasks, respectively. We conduct all the experiments on NVIDIA RTX 3090 GPUs with PyTorch 1.10.1.

4.4 Downstream Tasks and Datasets

Since there is no publicly available dataset on network device logs and to evaluate the capability of KnowLog on these logs, we construct six different log understanding downstream tasks. Depending on the type of input, these downstream tasks can be divided into two categories, log-single tasks, in which the input is a single log, and log-pair tasks, in which the input is a log-log pair or a log-NL pair. For each dataset, we divide the training, validation and testing set by a ratio of 6:2:2. We fine-tune the model with the training set to obtain the optimal results in the validation set, and finally evaluate and report the results on the testing set. Next, we give a brief introduction of each task including task definition, dataset preparation and the evaluation metrics. In Table 2 we provide statistics for different tasks of their datasets.

4.4.1 Module Classification (MC). MC is a log-single type multi-class classification task designed to identify the modules to which the logs belong. The input of this task is one log whose module name is masked and the output is the corresponding module name.

[5]https://www.cisco.com/c/en/us/support/all-products.html
[6]https://support.huawei.com/enterprise/en/index.html
[7]https://www.h3c.com/en/Support/

Table 2: Downstream tasks and their dataset statistics (Training/Validation/Testing Size).

		Switches	Routers	WLAN
Module Classification	Cisco	13,495/4,498/4,498	7,265/2,422/2,421	3,044/1,014/1,014
	Huawei	3,439/1,146/1,146	2,539/846/845	544/181/181
	H3C	1,241/413/413	1,336/445/444	724/241/241
Risk Log Identification	Huawei	788/263/262	502/167/166	379/126/125
Fault Phenomenon Identification	Huawei	362/120/120	-	-
Log and Description Semantic Matching	Cisco	49,954/16,651/16,651	26,975/8,992/8,991	7,910/2,636/2,636
	Huawei	7,702/2,567/2,567	5,977/1,992/1,991	1,202/400/400
	H3C	2,606/868/868	2,837/946/945	1,514/504/504
Log and Possible Cause Ranking	Huawei	3,851/1,283/1,283	3,097/1,032/1,032	602/200/200
Inter-vendor Module Matching	Huawei-Cisco	3,337/1,112/1,111	2,121/707/706	483/161/160
	Huawei-H3C	3,533/1,178/1,177	2,690/896/896	437/146/145
	Cisco-H3C	3,059/1,020/1,019	-	-

Datasets. We use the log template collected in Table 1 as raw data, the module name in the log as ground truth, and then replace the module name with *[MASK]* to avoid label leak, which is a special token.

Metrics. Since this is an unbalanced multi-class classification task and considering the importance of different classes, we use *Accuracy* and *Weighted F1-score* as evaluation metrics.

4.4.2 Risk Log Identification (RLI). RLI with the goal of identifying whether the log has an impact on the system is a log-single type of binary classification task. For example, logs for receiving invalid packets will impact the system, while notification-type logs will not impact the system. The input of this task is one log and the output is True (represents impact) or False (represents no impact).

Datasets. Each Huawei log contains the introduction of *Procedure* in the documentation, which indicates the action to be taken when the log occurs. We parse this field from the documentation and construct ground truth based on whether it contains explicit action steps. Specifically, logs with action steps as True, and logs without action steps as False.

Metrics. As a classical binary classification task, we use *Precision*, *Recall* and *F1-score* as the evaluation metrics to measure the effectiveness of detecting risk logs.

4.4.3 Fault Phenomenon Identification (FPI). The goal of FPI is to identify the fault phenomena to which the logs belong and this task is based on real-world data. The fault phenomenon can be described as a group of logs that regularly co-occur with a system failure, excluding repetitive logs. For example, the phenomenon of $boardfault$ results in two logs occurring together, BoardInvalid_active and BoardFail_active. Note that a single log may indicate different fault phenomena, e.g. $trunklinkdown$ and $physicallinkdown$ both contain the log linkDown_active. Therefore, we model the FPI as a multi-label classification task, where the input is a log and the output is one or more fault phenomena.

Datasets. We collected 602 Huawei switches log instances from the real-world scenario as the dataset, these logs are labeled by experts and cover a total of 43 fault phenomena.

Metrics. Different from the multi-class classification task, we use *Hamming-score* [51] to evaluate this multi-label classification task. Hamming-score is calculated as the **average accuracy** of all samples. The accuracy for each sample here is the number of correctly predicted labels as the proportion of the whole labels.

4.4.4 Log and Description Semantic Matching (LDSM). LDSM is a log-pair type binary classification task to measure whether the semantics of logs l match the corresponding natural language descriptions d. The input of LDSM is a (l, d) pair and the output is True or False.

Datasets. We first obtain descriptions of logs from the documentation, then label the logs with their corresponding descriptions as True, and randomly select one other description with the log labeled as False, where the ratio of True to False is 1:1.

Metrics. As a binary classification task of semantic matching, both positive and negative cases require attention, and we use the *Accuracy* and *Weighed F1-score* as evaluation metrics.

4.4.5 Log and Possible Cause Ranking (LPCR). LPCR is a log-pair type ranking task to identify the most probable underlying cause for a given log based on its semantic context. The input of LPCR is a log l as a query and a candidate set C containing the answer and the output is the ranking result. Ideally, the ground-truth cause should be ranked first.

Datasets. Each log in the Huawei documentation contains the introduction of the *Possible Cause*, and we parse this field as ground truth. The ground truth and 5 randomly selected possible causes of other logs as a candidate set, and the model is required to rank the candidate set.

Metrics. We use *Precision@1* and *Mean Reciprocal Rank* (MRR) as evaluation metrics. Precision@1 represents the precision of the first place. MRR is the metric that evaluates search performance by calculating the average of the reciprocal ranks for the given queries.

4.4.6 Inter-Vendor Module Matching (IVMM). IVMM is a log-pair type binary task to align the same modules across vendors, which can measure the semantic representation of logs whether more universal. The input of IVMM is two logs from different vendors and the output is True or False.

Datasets. We construct ground truth between different vendors' logs based on whether the module names are identical, and randomly sample logs from different modules as negative samples, where the ratio of True to False is 1:1. In addition, we replace the module names in the logs with *[MASK]* token to avoid label leakage during training.

Metrics. As a binary classification task, we also use the *Accuracy* and *Weighed F1-score* as evaluation metrics.

4.5 Evaluation

We evaluate KnowLog by answering five research questions.

RQ1: How effective is KnowLog on downstream tasks compared to the current mainstream methods? We conduct experiments on the six tasks mentioned above. Tables 3-5 show the experiment results for MC, RLI, LDSM, LPCR and IVMM respectively, and the results for FPI are shown in Figure 3. Overall, KnowLog outperforms other baselines on each of the datasets. We notice that: (1) Experiment results indicate that pre-trained language models outperform traditional methods, which suggests that the static word embedding models have a limited ability to represent logs. (2) Compared with general pre-trained models, the BERT-based KnowLog outperforms BERT by 9% in F1 value on the RLI task of Huawei (Switches) and 8.55% in average Accuracy of all datasets on the

Table 3: Results on Module Classification and Risk Log Identification.

Methods	MC (Accuracy/Weighted F1)									RLI (Precision/Recall/F1)		
	Cisco			Huawei			H3C			Huawei		
	Switches	Routers	WLAN	Switches	Routers	WLAN	Switches	Routers	WLAN	Switches	Routers	WLAN
CNN	56.89/56.85	57.46/54.92	53.55/51.89	74.52/73.95	72.78/72.23	73.48/71.47	69.49/67.55	70.72/69.71	74.27/72.87	0.63/0.62/0.63	0.62/0.59/0.61	0.68/0.69/0.68
BiLSTM (Attention)	55.74/55.63	57.17/56.76	53.25/52.38	76.52/75.49	73.96/73.30	73.48/72.51	70.21/68.45	71.40/69.93	74.69/73.56	0.68/0.64/0.66	0.57/0.62/0.59	0.67/0.72/0/69
UniLog	63.83/63.45	64.60/63.44	62.13/61.18	83.07/82.11	81.30/79.57	88.95/87.68	81.60/79.80	79.28/77.75	80.50/78.92	0.70/0.67/0.69	0.76/0.72/0.74	0.84/0.80/0.82
BERT	62.67/61.38	62.72/62.60	61.63/60.87	82.37/81.20	81.18/79.20	86.19/84.89	81.11/79.78	77.93/76.05	80.08/76.90	0.69/0.61/0.64	0.70/0.73/0.71	0.80/0.82/0.81
KnowLog (BERT)	**68.03/67.80**	**70.05/69.22**	**66.57/66.29**	**86.13/85.36**	**86.39/85.23**	**88.95/88.49**	**82.57/80.95**	**80.86/79.19**	**81.33/79.18**	**0.75/0.70/0.73**	**0.76/0.78/0.77**	**0.85/0.89/0.87**
RoBERTa	62.72/62.58	63.90/63.08	60.95/60.77	81.50/80.64	81.18/79.20	86.19/85.01	81.35/79.73	78.60/76.81	80.08/77.97	0.70/0.64/0.67	0.70/0.73/0.71	0.80/0.82/0.81
KnowLog (RoBERTa)	**68.32/67.88**	**71.62/70.89**	**67.16/66.72**	**86.39/85.30**	**86.27/84.93**	**88.95/87.68**	**82.08/80.45**	**80.63/79.33**	**80.50/78.17**	**0.81/0.72/0.77**	**0.79/0.82/0.81**	**0.81/0.87/0.84**

Table 4: Results on Log and Description Semantic Matching and Log and Possible Cause Ranking.

Methods	LDSM (Accuracy/Weighted F1)									LPCR (Precision@1/MRR)		
	Cisco			Huawei						Huawei		
	Switches	Routers	WLAN	Switches	Routers	WLAN	Switches	Routers	WLAN	Switches	Routers	WLAN
CNN	84.04/84.04	80.99/80.99	72.15/72.16	86.05/86.05	82.37/82.30	72.75/72.75	83.29/83.19	83.60/83.59	82.54/82.42	-	-	-
BiLSTM (Attention)	89.45/89.44	85.42/85.41	76.86/76.86	87.85/87.85	84.43/84.40	72.75/72.73	80.88/80.83	83.81/83.80	82.74/82.72	-	-	-
UniLog	93.90/93.90	92.07/92.07	83.99/84.00	95.01/95.01	93.17/93.17	86.25/86.15	93.32/93.32	92.38/92.38	90.87/90.85	0.894/0.934	0.899/0.939	0.875/0.923
BERT	93.06/93.06	90.01/90.00	79.74/79.74	93.18/93.18	90.06/90.05	79.75/79.74	87.44/87.41	88.25/88.25	83.93/83.81	0.884/0.928	0.876/0.923	0.826/0.891
KnowLog (BERT)	**98.02/98.02**	**97.56/97.56**	**93.51/93.51**	**97.20/97.20**	**96.74/96.74**	**93.50/93.49**	**95.97/95.97**	**96.30/96.30**	**93.45/93.44**	**0.952/0.972**	**0.946/0.968**	**0.841/0.897**
RoBERTa	93.03/93.03	89.26/89.24	78.11/78.10	92.82/92.83	90.31/90.31	80.50/80.50	87.44/87.42	89.31/89.31	83.13/83.10	0.895/0.938	0.862/0.906	0.841/0.894
KnowLog (RoBERTa)	**96.56/96.56**	**96.32/96.32**	**93.25/93.25**	**97.20/97.20**	**96.23/96.23**	**93.25/93.24**	**95.05/95.04**	**96.08/96.08**	**94.84/94.84**	**0.935/0.962**	**0.935/0.963**	**0.861/0.910**

Table 5: Results on IVMM. (Huawei-Cisco indicates two logs from Huawei and Cisco, respectively.)

Methods	Huawei-Cisco			Huawei-H3C		
	Switches	Routers	WLAN	Switches	Routers	WLAN
CNN	69.58/69.53	67.56/67.40	81.25/81.19	80.88/80.87	72.43/72.05	66.21/66.24
BiLSTM (Attention)	71.56/71.56	68.41/68.42	80.63/80.36	81.39/81.32	77.23/77.21	68.28/68.33
UniLog	77.59/77.55	74.93/74.77	83.75/83.76	90.48/90.48	90.74/90.74	75.86/74.97
BERT	74.26/74.25	70.54/70.54	82.50/82.51	86.83/86.83	85.38/85.38	70.34/70.44
KnowLog (BERT)	**79.30/79.28**	**78.61/78.61**	**87.50/87.51**	**92.86/92.86**	**91.74/91.74**	**77.24/77.22**
RoBERTa	74.71/74.70	70.40/70.35	83.13/83.13	87.17/87.17	87.28/87.26	70.34/70.08
KnowLog (RoBERTa)	**80.20/80.19**	**79.18/79.17**	**86.25/86.25**	**92.35/92.35**	**92.41/92.41**	**77.93/77.89**

LDSM task. (3) Further, although UniLog also outperforms BERT and RoBERTa, KnowLog exhibits a notable superiority over UniLog across all datasets. It can be inferred that due to the difference between natural language and logs, pre-training on logs can minimize this gap and make the pre-trained model more specialized. Through in-depth analysis with UniLog, we find that since logs include domain terminology and are highly concise, without domain knowledge enhancement, the simple pre-training task such as reconstructing masked tokens cannot address our proposed challenges and justify the necessity of introducing knowledge. (4)Through a comparative analysis of existing pre-trained models, it becomes evident that KnowLog delivers a noteworthy performance improvement, indicating the effectiveness of KnowLog in acquiring underlying knowledge. In contrast to UniLog's straightforward self-supervised task, our pre-training tasks enhanced with knowledge prove to be more effective.

In addition, on the IVMM task, KnowLog still has significant advantages, which indicates that KnowLog can capture commonality in logs and obtain more universal representations. On the real-world scenario task, from Figure 3, we can see that KnowLog still has an advantage over other pre-trained models under fine-tuning with log instances. This suggests that KnowLog effectively resolves real-world practical issues and provides evidence that pre-training on log templates still has good generalization on log instances.

In conclusion, KnowLog is effective in log analysis tasks and outperforms existing traditional DL-based methods and pre-trained models. Although pre-training on logs is effective, lack of domain knowledge and simple self-supervised tasks for pre-training are not sufficient for log understanding. In addition, these findings suggest that enhancing KnowLog's performance involves more than just pre-training on the log corpus, knowledge also plays a critical role. As demonstrated by the overall experimental results, the knowledge-enhanced pre-training tasks outperform the simple reconstructing masked tokens task.

RQ2: Is KnowLog still effective for downstream tasks in low-resource scenarios? In the scenario of automated log analysis, it is usually difficult to obtain a large amount of labeled data, such as identifying the fault phenomena, and another characteristic of such tasks is that the scale of data is usually not large due to the low frequency of occurrence. Therefore, the available supervised training data are scarce, and we define such tasks as low-resource scenario tasks. To evaluate the performance of KnowLog in low-resource scenarios, we sample different proportions of data from the training set of the FPI task for experiments, sampling from 30% to 70% of the original scale in the step of 10%.

The experimental results are shown in Figure 3, we find that KnowLog still outperforms UniLog and the corresponding BERT and RoBERTa models when training on the full samples. As the training data scale decreases, the performance of BERT, RoBERTa and UniLog decreases sharply, while KnowLog is in a slow downward trend. We surprisingly find that KnowLog outperforms BERT and RoBERTa on the full sample when we have 60% of the training samples. With only 30% training samples, compared to the full sample, the accuracy of UniLog decreases by 31.25%, BERT and RoBERTa decrease by 31.45% and 30.83%, respectively, while KnowLog decreases by only 18.91% and 9.79%, respectively. It shows that KnowLog has learned an amount of underlying knowledge by knowledge enhancement and with a small sample fine-tuning this knowledge can be generalized. We can infer that the model is embedded with a large amount of implicit knowledge with knowledge enhancement that can be effectively activated with a small amount of labeled data. In addition, in practical applications, good results can still be achieved by fine-tuning on a small number of samples, which can reduce the annotations required for the task.

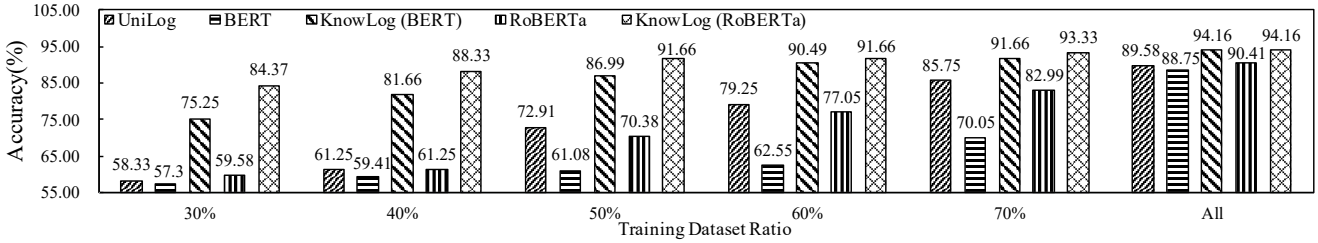

Figure 3: Results of KnowLog under different ratios of training dataset on the FPI task.

Table 6: Results of transfer learning experiments on the task of LDSM. Left side of → indicates source dataset for training and right side indicates target dataset for testing.

Methods	Huawei → Cisco			Cisco → Huawei		
	Switches	Routers	WLAN	Switches	Routers	WLAN
CNN	62.25/61.59	62.05/61.55	52.35/52.07	69.61/69.40	62.13/62.13	60.25/60.09
BiLSTM (Attenion)	64.00/63.96	60.94/60.85	55.58/55.40	72.03/71.85	66.60/66.52	59.25/59.25
UniLog	77.57/77.40	77.33/77.27	66.69/66.70	90.49/90.49	87.74/87.74	83.75/83.73
BERT	71.46/71.13	72.97/72.88	61.38/61.36	83.73/83.72	84.53/84.52	72.75/72.25
KnowLog (BERT)	**86.63/86.59**	**87.22/87.19**	**80.39/80.04**	**95.56/95.56**	**93.17/93.16**	**96.25/96.25**
RoBERTa	71.95/71.74	73.57/73.41	61.53/61.54	84.03/84.02	84.63/84.62	73.25/73.25
KnowLog (RoBERTa)	**84.20/84.04**	**86.52/86.43**	**80.35/80.00**	**96.34/96.34**	**93.62/93.62**	**96.75/96.74**

In conclusion, KnowLog has a remarkable superiority in low-resource scenarios, which inspires us to seek domain knowledge to improve generalizability in low-resource scenarios. With the increase of fine-tuning data, KnowLog can obtain better performance.

RQ3: How effective is KnowLog in transfer learning across vendors? Since different vendor logs are usually different in syntax [7], different models need to be deployed in a specific task In addition, training different models requires different vendors' annotated data. Obviously, this is quite troublesome, and we expect that one model trained with only annotated data (e.g., annotated data from one vendor) can be transferred to different vendors, which requires the model with transfer learning ability. To evaluate the semantic transfer ability of different vendor logs, we conduct experiments on the LDSM task. Specifically, we train the model with a training set from a source vendor and then make predictions on a testing set from another target vendor. For example, we train the model on the training set of Huawei's WLAN and then evaluate it on the testing set of Cisco's WLAN.

The experiment results are shown in Table 6, results indicate that KnowLog also outperforms other baselines. Statistically, it is found that the BERT-based KnowLog has an average Accuracy improvement of 9.28% and 13.73%, respectively, over BERT and UniLog on all datasets. Compared with traditional DL methods, traditional methods perform the worst on transfer learning, which suggests that the limited semantic comprehension of traditional methods hinders their ability to generalize to logs with different syntax. Compared with BERT and RoBERTa, owing to pre-training on the larger corpus, pre-trained models exhibit better transfer performance compared to conventional approaches. Further, the outperformance of UniLog over BERT and RoBERTa demonstrates that pre-training on logs can be adapted to different syntaxes. On top of that, KnowLog still has a clear advantage over UniLog. After being enhanced with domain knowledge, the model's ability to generalize common knowledge within the logs is improved, enabling the model to acquire more universal semantic representations. We argue that this capability derives from the introduction of domain knowledge and data augmentation for logs. These operations allow logs with different syntax to map in the same semantic space by aligning descriptions of logs and positive samples, thus reducing the impact of different syntax on log understanding.

Consequently, it can be concluded that KnowLog demonstrates superior transfer learning capabilities when processing logs from different vendors. In addition, the introduction of domain knowledge also inspires us to introduce a new perspective to the study of log transfer learning.

Table 7: Results of whether abbreviations join the vocabulary.

Tasks		Abbr Not in Vocab	Abbr In Vocab
LDSM (Huawei)	Switches	96.61/96.61	**97.20/97.20**
	Routers	95.28/95.28	**96.74/96.74**
	WLAN	90.50/90.50	**93.50/93.49**
IVMM (Huawei-Cisco)	Switches	78.94/78.92	**79.30/79.28**
	Routers	75.50/75.51	**78.61/78.61**
	WLAN	86.88/86.88	**87.50/87.51**

RQ4: How does adding abbreviations from logs to KnowLog's vocabulary affect the performance of the model? For enhancing the pre-trained language model to understand the abbreviations in logs, we propose an abbreviation prediction task and add the abbreviations to the tokenizer's vocabulary. In order to explore the impact of adding abbreviations into the vocabulary of the tokenizer, we also train a model keeping the vocabulary unchanged based on the same pre-training objectives as a comparison and evaluate it on downstream tasks.

The experiment results are shown in Table 7, we found that adding abbreviations to the vocabulary can improve the performance of the model. From this, we argue that preserving the integrity of the domain terminology is more effective in understanding the semantics of the terminology.

After careful analysis, it can be inferred that the split domain terminology can destroy the semantic information of tokens thus causing the degradation of the model performance. This inspires us to avoid destroying the integrity of domain terminology can improve the performance of the model and thus obtain more effective log representations.

RQ5: How effective is KnowLog pre-training on log templates compared with log instances? In this paper, we pre-trained KnowLog on log templates and the reasons were discussed in Section 4.1. In order to experimentally compare the effectiveness of pre-training on log instances and log templates, we collect 133,235 log

Table 8: Result on the log instances pre-training. Evaluation metrics are Accuracy/Weighted F1.

Methods	MC		LDSM	
	Huawei	H3C	Huawei	H3C
KnowLog (Log instances)	82.55/81.54	80.15/77.57	91.51/91.50	86.06/86.04
KnowLog (Log templates)	**85.51/85.44**	**82.32/80.70**	**93.53/93.53**	**88.82/88.81**

instances without data cleaning from *Huawei 12800 Series Switches* for pre-training where there is a large amount of duplication in the instance logs. As a comparison, we also pre-train with only 6,418 Huawei Switches log templates. The experiment environment and setup are the same as KnowLog (BERT). Finally, we evaluate them on the Module Classification (MC) task and Log and Description Semantic Matching (LDSM) task.

The experimental results are shown in Table 8, we can find that pre-training on log templates outperforms log instances. This indicates that duplicate log instances reduce the effectiveness of pre-training. In addition, we also evaluate downstream tasks beyond the pre-training data, experimental results on H3C indicate that duplicate pre-training data also can reduce the generalization ability of the model. This suggests that the single log template collected from the documentation has a better performance due to the balanced distribution of logs. Since we are unable to collect the full log instances, we cannot further prove the effectiveness of pre-training on log instances. Moreover, considering that the parameters of logs are rich in semantic information as well, we do not exclude the use of log instances for pre-training, as long as data quality is guaranteed.

In conclusion, the pre-training corpus has a significant impact on log pre-training, and the log instances without data cleaning that contain duplicate logs can degrade the performance of the model.

4.6 Ablation Studies

To investigate the effectiveness of each pre-training objective, we conduct ablation experiments on MC and LDSM, which are two typical tasks of multi-class classification task and semantic matching task. We analyze the results by observing 10 epochs on the validation set. The results of the experiments are shown in Figure 4, where we notice that: (1) KnowLog with pre-training on all objectives exhibits the most superior performance. Moreover, it can be found that a reduction in model performance upon the removal of any objective, thereby highlighting the positive impact of our proposed objective on enhancing the model's pre-training; (2) with missing abbreviation prediction (AP) objective, the model performance decreases on both tasks, more especially on the MC task, confirming that the understanding of abbreviations is an essential part of understanding the semantic of logs and AP can help learn local knowledge from abbreviations; (3) the performance on the LDSM task decreases significantly without description discrimination (DD) objective, indicating that natural language descriptions can enrich the semantic of logs and DD allows the model to gain global knowledge from the descriptions; (4) without the log contrastive learning (LCL) objective, a different degree of decrease is observed on both tasks, indicating that LCL can obtain more universal semantic representations and the universal representations can enhance performance in downstream tasks.

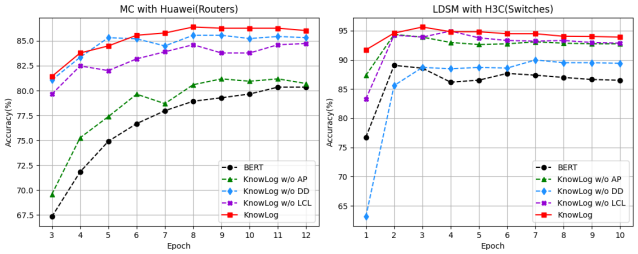

Figure 4: Ablation studies for KnowLog.

Table 9: Qualitative examples of KnowLog and baselines. The input is a Log-NL or Log-Log pair and the Score indicates the cosine similarity.

Label	Examples	Models	Score
Match	**Log**: BGP/4/ASPATH_EXCEED_MAXNUM: The number of AS-PATHs exceeded the limit([limit-value]). (Operation=[STRING])	BERT	0.7250
		UniLog	0.7061
	NL: The number of AS-Paths exceeded the maximum value.	KnowLog	0.8006
UnMatch	**Log**: BGP/4/ASPATH_EXCEED_MAXNUM: The number of AS-PATHs exceeded the limit([limit-value]). (Operation=[STRING])	BERT	0.5715
		UniLog	0.3008
	NL: The OSPF process successfully exited from GR.	KnowLog	0.0056
Match	**Log1**: DEVM/3/hwRemoteFaultAlarm_active(l): The remote fault alarm has occurred. (IfIndex=27, IfName=10GE1/0/17)	BERT	0.9550
		UniLog	0.8031
	Log2: DEVM/3/hwRemoteFaultAlarm_active: The remote fault alarm has occurred.	KnowLog	0.9750
UnMatch	**Log1**: BGP/4/ASPATH_EXCEED_MAXNUM: The number of AS-PATHs exceeded the limit([limit-value]). (Operation=[STRING])	BERT	0.8338
		UniLog	0.2997
	Log2: DEVM/3/hwRemoteFaultAlarm_active: The remote fault alarm has occurred.	KnowLog	0.1514

4.7 Qualitative Analysis

In order to verify the effectiveness of KnowLog intuitively, we select four representative cases for qualitative analysis. Specifically, we employ a pre-trained model without fine-tuning to obtain embeddings of inputs and then calculate the cosine similarity between embeddings as the metric. Two types of inputs are selected for analysis: Log-NL and Log-Log. In the Log-NL type, an input consists of a log and a natural language description. In the Log-Log type, an input consists of two logs. The positive sample includes a log instance and its corresponding log template, while the negative sample includes two different log templates.

The experiment results are shown in Table 9. From these cases, BERT demonstrates a proficient semantic understanding of positive examples, while it exhibits limited capability in discerning negative examples. This indicates BERT's powerful linguistic capability but also its bias in understanding log semantics. Conversely, UniLog excels in accurately distinguishing negative samples, but it does not enhance the semantic comprehension of positive samples. This shows that pre-training on logs can improve the differentiation among logs, but the lack of domain knowledge cannot further improve the semantic understanding of logs. We observe that KnowLog achieves the highest similarity scores for all positive samples and the lowest similarity scores for all negative samples. This indicates that KnowLog can better capture log semantics, thereby confirming its effectiveness in log understanding. Furthermore, through the analysis of Log-NL cases, we find that KnowLog demonstrates improved accuracy in semantic understanding, benefiting from domain knowledge enhancement. Similarly, in the analysis of Log-Log cases, KnowLog proves capable of obtaining more universal representations, benefiting from contrastive learning between logs. The above results demonstrate KnowLog's excellent performance in log understanding.

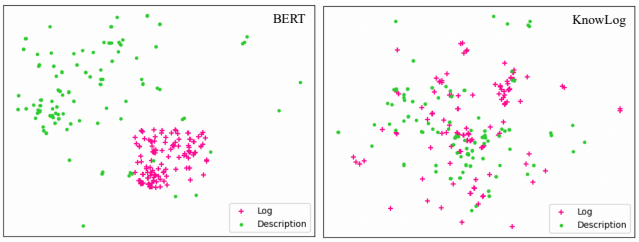

Figure 5: The representation visualization of logs and the corresponding descriptions.

5 DISCUSSIONS

In this section, we discuss open questions regarding the performance and threats to the validity of KnowLog.

5.1 Why does KnowLog Work?

There are two reasons to make KnowLog work better. First, KnowLog is pre-trained on logs and fine-tuned on different tasks. The general pre-trained models focus on understanding human language rather than the specific logs, and KnowLog focuses on understanding logs. By pre-training on log corpus, the model acquires the capability to understand logs and effectively capture their contextual information. Subsequently, when fine-tuning the model, the knowledge embedded within the pre-trained model can be proficiently applied to various downstream tasks.

Second, compared to existing pre-trained models, KnowLog leverages local and global knowledge to enhance pre-training respectively. In addition, KnowLog also makes the semantic representations of logs more universal through contrastive learning. With domain knowledge enhancement, KnowLog can capture the log semantics comprehensively and has better performance in automated log analysis tasks. To visually verify the effectiveness of KnowLog, we give the visualization of the model representations in Figure 5. From the visualization results, we can find that the representation of logs and the corresponding natural language descriptions based on BERT are clearly separated in the vector space, and both of them are almost unaligned. Ideally, logs should be able to be aligned with their corresponding descriptions in semantic space. This indicates that the current pre-trained model has an obvious semantic gap between the logs and their corresponding descriptions. In contrast, KnowLog is able to pull logs and the corresponding descriptions closer, allowing the model to understand the logs comprehensively.

5.2 Threats to Validity

Construct Validity. This study relies on natural language descriptions obtained from public documentation as a source of global knowledge to enhance the model and we assume that the source of knowledge is reliable. However, the quality of the model's performance can be influenced by potential noise present in the descriptions. To mitigate this issue, we have performed data cleaning on the collected descriptions, including supplementing missing descriptions manually and removing identical descriptions to ensure knowledge quality.

It should be noted that our approach primarily considers abbreviations and descriptions as knowledge sources, and does not incorporate other forms of knowledge such as *Code*, *Configuration*, and *Flowcharts*. While our approach can be theoretically generalized to other types of logs with the provision of domain knowledge, it is limited in processing other knowledge. For enhancing the model with other knowledge, our approach may not be the optimal solution and we will address these challenges in future work.

Internal Validity. It is a widely accepted fact that hyperparameters have a significant effect on the DL model. Due to the limited computational resources, some hyperparameters of KnowLog are not tuned experimentally but rather determined empirically. Therefore there exist other hyperparameter settings might give better results. On top of this, the encoder of KnowLog used only explored BERT and RoBERTa architecture, based on the scaling law [22], larger models and more data will yield better results.

External Validity. Another threat is the imbalance of the data. As shown in Table 1, the distribution of data is uneven among vendors and devices, leading to imbalanced datasets. Therefore, our method does not guarantee the same results on different devices and vendors. To address this concern, we have employed the Weighted F1-score as a more appropriate evaluation metric, which accounts for the imbalanced nature of the data.

6 CONCLUSION

In this paper, we propose KnowLog, a knowledge-enhanced pre-trained language model for log understanding, which improves state-of-the-art performance on log understanding tasks. We propose three novel pre-training tasks effectively leveraging domain knowledge to enhance the model and make the representations of logs more universal. After pre-training with knowledge enhancement, KnowLog was fine-tuned on six different downstream tasks. Compared to other pre-trained models without knowledge enhancement, KnowLog achieves state-of-the-art performance, which proves the effectiveness of knowledge for improving log understanding. Ablation analysis demonstrates the effectiveness of these pre-training tasks for leveraging knowledge to understand logs.

In the future, we will explore the integration of more types of knowledge to further improve log understanding such as configuration, code, flowcharts, and other multi-modal information.

7 ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their feedback which helped improve this paper. This work is supported by the National Key Research and Development Project (No.2020AAA0109302), the National Natural Science Foundation of China under No.U2033209.

REFERENCES

[1] Miltiadis Allamanis. The adverse effects of code duplication in machine learning models of code. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pages 143–153, 2019.

[2] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, 2019.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *USENIX Security Symposium*, volume 6, pages 2633–2650, 2021.

[5] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. Legal-bert: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, 2020.

[6] Qibin Chen, Jeremy Lacomis, Edward J Schwartz, Graham Neubig, Bogdan Vasilescu, and Claire Le Goues. Varclr: Variable semantic representation pre-training via contrastive learning. In *Proceedings of the 44th International Conference on Software Engineering*, pages 2327–2339, 2022.

[7] Rui Chen, Shenglin Zhang, Dongwen Li, Yuzhe Zhang, Fangrui Guo, Weibin Meng, Dan Pei, Yuzhi Zhang, Xu Chen, and Yuqing Liu. Logtransfer: Cross-system log anomaly detection for software systems with transfer learning. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 37–47. IEEE, 2020.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607, 2020.

[9] Oihana Coustié, Josiane Mothe, Olivier Teste, and Xavier Baril. Meting: A robust log parser based on frequent n-gram mining. In *2020 IEEE International Conference on Web Services (ICWS)*, pages 84–88. IEEE, 2020.

[10] Li Du, Xiao Ding, Kai Xiong, Ting Liu, and Bing Qin. Enhancing pretrained language models with structured commonsense knowledge for textual inference. *Knowledge-Based Systems*, 254:109488, 2022.

[11] Min Du, 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*, pages 1285–1298, 2017.

[12] Chris Egersdoerfer, Dong Dai, and Di Zhang. Clusterlog: Clustering logs for effective log-based anomaly detection. *arXiv preprint arXiv:2301.07846*, 2023.

[13] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, 2021.

[14] Sina Gholamian and Paul AS Ward. On the naturalness and localness of software logs. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 155–166. IEEE, 2021.

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

[16] Hongcheng Guo, Xingyu Lin, Jian Yang, Yi Zhuang, Jiaqi Bai, Bo Zhang, Tieqiao Zheng, and Zhoujun Li. Translog: A unified transformer-based framework for log anomaly detection. *arXiv preprint arXiv:2201.00016*, 2021.

[17] Xiao Han and Shuhan Yuan. Unsupervised cross-system log anomaly detection via domain adaptation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3068–3072, 2021.

[18] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R Lyu. A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)*, 54(6):1–37, 2021.

[19] Shilin He, Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, Michael R Lyu, and Dongmei Zhang. Identifying impactful service system problems via log analysis. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 60–70, 2018.

[20] Shaohan Huang, 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(4):2064–2076, 2020.

[21] Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 10697–10707. PMLR, 2022.

[22] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[23] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[24] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, 2018.

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

[26] Van-Hoang Le and Hongyu Zhang. Log-based anomaly detection with deep learning: How far are we? In *Proceedings of the 44th International Conference on Software Engineering*, pages 1356–1367, 2022.

[27] Van-Hoang Le and Hongyu Zhang. Log parsing with prompt-based few-shot learning. *arXiv preprint arXiv:2302.07435*, 2023.

[28] Jinhyuk Lee, 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(4):1234–1240, 2020.

[29] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, 2022.

[30] Eric Lehman, Sarthak Jain, Karl Pichotta, Yoav Goldberg, and Byron C Wallace. Does bert pretrained on clinical notes reveal sensitive data? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 946–959, 2021.

[31] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[32] Xiaoyun Li, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba Yu. Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 92–103. IEEE, 2020.

[33] Yinhan Liu, 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.

[34] Yudong Liu, Hailan Yang, Pu Zhao, Minghua Ma, Chengwu Wen, Hongyu Zhang, Chuan Luo, Qingwei Lin, Chang Yi, Jiaojian Wang, et al. Multi-task hierarchical classification for disk failure prediction in online service systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3438–3446, 2022.

[35] Yudong Liu, Xu Zhang, Shilin He, Hongyu Zhang, Liqun Li, Yu Kang, Yong Xu, Minghua Ma, Qingwei Lin, Yingnong Dang, et al. Uniparser: A unified log parser for heterogeneous log data. In *Proceedings of the ACM Web Conference 2022*, pages 1893–1901, 2022.

[36] Siyang Lu, 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)*, pages 151–158. IEEE, 2018.

[37] Chuan Luo, Pu Zhao, Bo Qiao, Youjiang Wu, Hongyu Zhang, Wei Wu, Weihai Lu, Yingnong Dang, Saravanakumar Rajmohan, Qingwei Lin, et al. Ntam: Neighborhood-temporal attention model for disk failure prediction in cloud platforms. In *Proceedings of the Web Conference 2021*, pages 1181–1191, 2021.

[38] Minghua Ma, Yudong Liu, Yuang Tong, Haozhe Li, Pu Zhao, Yong Xu, Hongyu Zhang, Shilin He, Lu Wang, Yingnong Dang, et al. An empirical investigation of missing data handling in cloud node failure prediction. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1453–1464, 2022.

[39] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[40] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, volume 19, pages 4739–4745, 2019.

[41] Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. Self-supervised log parsing. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part IV*, pages 122–138. Springer, 2021.

[42] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. Bertweet: A pre-trained language model for english tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, 2020.

[43] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

[44] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[45] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019.

[46] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. Bpe-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, 2020.

[47] Jiaxing Qi, Zhongzhi Luan, Shaohan Huang, Carol Fung, Hailong Yang, Hanlu Li, Danfeng Zhu, and Depei Qian. Logencoder: Log-based contrastive representation learning for anomaly detection. *IEEE Transactions on Network and Service Management*, 2023.

[48] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[49] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[50] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

[51] Mohammad S Sorower. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18(1):25, 2010.

[52] Shimin Tao, Weibin Meng, Yimeng Cheng, Yichen Zhu, Ying Liu, Chunning Du, Tao Han, Yongpeng Zhao, Xiangguang Wang, and Hao Yang. Logstamp: Automatic online log parsing based on sequence labelling. *ACM SIGMETRICS Performance Evaluation Review*, 49(4):93–98, 2022.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[54] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuan-Jing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, 2021.

[55] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

[56] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[57] Zhirong Wu, Yuanjun Xiong, X Yu Stella, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3733–3742. IEEE Computer Society, 2018.

[58] Tong Xiao, Zhe Quan, Zhi-Jie Wang, Kaiqi Zhao, and Xiangke Liao. Lpv: A log parser based on vectorization for offline and online log parsing. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1346–1351. IEEE, 2020.

[59] Rakesh Bahadur Yadav, 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)*, pages 1215–1220. IEEE, 2020.

[60] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenqing Fu, Xiya Wu, Qingwei Lin, and Dongmei Zhang. Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning. In *Proceedings of the 44th International Conference on Software Engineering*, pages 623–634, 2022.

[61] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817, 2019.

[62] Xu Zhang, Yong Xu, Si Qin, Shilin He, Bo Qiao, Ze Li, Hongyu Zhang, Xukun Li, Yingnong Dang, Qingwei Lin, et al. Onion: identifying incident-indicating logs for cloud systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1253–1263, 2021.

[63] Nengwen Zhao, Honglin Wang, Zeyan Li, Xiao Peng, Gang Wang, Zhu Pan, Yong Wu, Zhen Feng, Xidao Wen, Wenchi Zhang, et al. An empirical investigation of practical log anomaly detection for online service systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1404–1415, 2021.

[64] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 121–130. IEEE, 2019.

[65] Yichen Zhu, Weibin Meng, Ying Liu, Shenglin Zhang, Tao Han, Shimin Tao, and Dan Pei. Unilog: Deploy one model and specialize it for all log analysis tasks. *arXiv preprint arXiv:2112.03159*, 2021.