2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) | 978-1-6654-7679-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISSREW55968.2022.00053

LogVM: Variable Semantics Miner for Log Messages

Yintong Huo
Dept. of Computer Science & Engineering
the Chinese University of Hong Kong
Hong Kong, China
ythuo@cse.cuhk.edu.hk

Yuxin Su
School of Software Engineering
Sun Yat-sen University
Zhuhai, China
suyx35@mail.sysu.edu.cn

Michael Lyu
Dept. of Computer Science & Engineering
the Chinese University of Hong Kong
Hong Kong, China
lyu@cse.cuhk.edu.hk

***Abstract*—Modern automated log analytics rely on log events without paying attention to variables. However, variables, such as the return code (e.g., "404") in logs, are noteworthy for their specific semantics of system running status. To unlock the critical bottleneck of mining such semantics from log messages, this study proposes LogVM with three components: (1) an encoder to capture the context information; (2) a pair matcher to resolve variable semantics; and (3) a word scorer to disambiguate different semantic roles. The experiments over seven widely-used software systems demonstrate that LogVM can derive rich semantics from log messages. We believe such uncovered variable semantics can facilitate downstream applications for system maintainers.**

***Index Terms*—log analysis, variable semantics, text mining**

I. Introduction

A log message is a type of semi-structured language comprising natural language written by software developers and some auto-generated variables during software execution. While logs carry detailed software run-rime information for operators to monitor the software status, the overwhelming logs impede developers from reading every line of log files as modern software systems get more complicated. Intelligent software engineering necessitates automated log analysis.

Unfortunately, modern log analytics only uses the natural language part (i.e., event) and neglects the informative variables (i.e., parameters) inside logs. However, variables inside logs with latent semantics should be noticed. While humans seldomly use digits or character strings (e.g., 949e1227) in communication, parameters in the log message are important with specific meaning. Intuitively, a parameter in a log is used to specify another technical concept in the log. Taking the log in Table I as an example, a knowledgeable engineer understands that the token "949e1227" refers to another token "cell", so "949e1227" is a cell ID. In this way, exploiting such latent semantics benefits the understanding of parameters.

TABLE I: An Example of the log message and its corresponding event, variable and variable semantics.

Log messages	Listing instances in cell 949e1227
Event	Listing instances in cell $< * >$
Variable	949e1227
Variable Semantics	(cell, 949e1227)

Existing log parsing approaches [1], [2] used regular expressions to recognize block ID or IP address from log messages, but none of them captured the semantics of variables. Moreover, designing human handcrafted rules requires tedious effort and can only cover a fairly limited part of the logs.

In this study, we present a new approach, LogVM (i.e., Log Variables Miner), to automate latent mining semantics for variables from log messages via the multi-task training strategy. To our best knowledge, LogVM is the first model that aims to extract the structured semantics of log variables.

II. Terminology

We introduce the terminology by clarifying *concepts* and *instances* firstly, then define the *variable semantics* in logs. In particular, we use the term *concepts* to describe a set of technical terms appearing in a log, such as "cell". We use the term *instances* to denote variables in log messages. In this case, one concept can be instantiated by multiple instances. Therefore, the *variable semantics* are represented by a set of *Concept-Instance pairs (CI pairs)*, which describe the concept that the instance refers to, such as (cell, 949e1227).

III. Overview of LogVM

LogVM is an end-to-end model equipped with three modules to acquire the latent semantics of variables shown in Figure 1. First, a log message is fed into a *Contextual Encoder* for acquiring context-based word representation. Then, the contextualized word representations are separately used for two sub-tasks: (1) a *Pair Matcher* for extracting CI pairs; and (2) a *Word Scorer* for determining the semantic role of each word, respectively. In the training stage, LogVM applies the multi-task learning strategy to simultaneously optimize two sub-tasks. The extracted CI pairs in the testing phase will be considered as variable semantics.

978-1-6654-7679-9/22/$31.00 ©2022 IEEE
DOI 10.1109/ISSREW55968.2022.00053

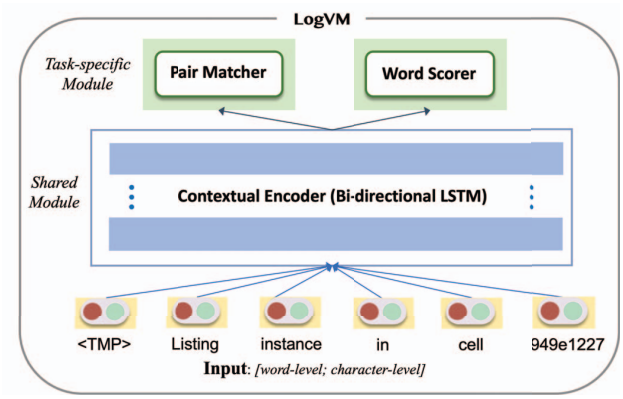

Fig. 1: LogVM Architecture.

A. Contextual Encoder

Motivated by the success of long short-term memory networks (LSTM) across natural language processing tasks, we devise a bi-directional LSTM-based network as the contextual encoder to capture interactions and dependencies between words in log messages. Considering the severe out-of-vocabulary (OOV) problem [3] caused by the large portion of customized words in log messages (e.g., function names, cell IDs), LogVM also takes character-level representation learnt from a convolutional neural network. Consequently, for each ward, its word-level representation and the character-level representation are concatenated for contextual word encoding.

B. Pair Matcher

To resolve the variable semantics, this module is designed for discerning the (*concept*, *instance*) pairs between words in a log message. We abstract this problem as a multi-classifier problem: for each word w_i in a sentence $S = w_1, w_2, ..., w_n$, the matcher determines what previous word $w_j (0 \leq j < i)$ does the word w_i refer to. [1] For each word, the pair matcher built upon a feed-forward neural network ranks the probability score for each previous token and regards the highest one as the semantic pair.

C. Word scorer

We build a word scorer to determine the semantic role for each token, i.e., whether it is a *concept*, an *instance* or *neither of both* via another feed-forward neural network. The semantic role is crucial because it can be used to disambiguate which word in a CI pair is an instance (or a concept). Moreover, word scorer benefits the pair matcher. Compared with two identified concepts, one concept and one instance are more likely to be a CI pair.

D. Multi-task training

Multi-task learning (MTL) is a training paradigm that trains a collection of neural network models for multiple tasks simultaneously [4] by leveraging the shared data representation for learning common knowledge. By minimizing the loss of pair matcher and word scorer, the model naturally learns the CI pairs and the semantic roles for each token with shared representations generated from the contextual encoder. In the inference, for each word, we regard the highest probability of its semantic pairs and its semantic role as the final results.

[1] A dummy token <TMP> (w_0) is added to indicate the word does not refer to any of the previous words in the message (e.g., in).

TABLE II: Statistics and experimental results.

System	# CI Pairs	Precision	Recall	F1
Android	6,478	0.979	0.858	0.915
Linux	2,905	0.992	0.947	0.969
Hadoop	2,592	0.993	0.880	0.933
HDFS	3,105	1.000	0.999	0.999
OpenStack	4,367	0.994	0.989	0.992
Spark	4,887	1.000	0.937	0.967
Zookeeper	1,189	0.997	0.940	0.968
Average	-	0.994	0.936	0.963

IV. Preliminary Result

We evaluate LogVM on seven widely-applied system log files with 14,000 log messages (2,000 for each) and 25,523 CI pairs. Two volunteer Ph.D. students majoring in Computer Science annotated the datasets separately and deliberate over their answers to reach consensus.

In the experiment, we train LogVM on 150 OpenStack logs, then finetune it on 50 logs for other systems and display the result in Table II. We achieve an average F1 score of 0.963 for seven systems logs with even a small number of the fine-tined samples. The results demonstrate that our model could extract high-quality and comprehensive variable semantics from log messages.

V. Conclusion and Future Work

In this study, we first discuss the limitations of variable semantics usage in modern log analytics and then suggest LogVM, the first automated variable semantics mining approach for log messages. Afterward, we demonstrate the efficacy of LogVM over seven representative systems.

Apart from the intra-message semantics, it is also observed that semantics can be resolved from the inter-message level. In the future, we will (1) extend LogVM to cover semantics scattering in several log messages; (2) evaluate how the extracted variable semantics can help the downstream applications.

References

[1] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *Proceedings of the 2017 IEEE ICWS, Honolulu, HI, USA, June 25-30*, pp. 33–40.

[2] S. Messaoudi, A. Panichella, D. Bianculli, L. Briand, and R. Sasnauskas, "A search-based approach for accurate identification of log message formats," in *Proceedings of the 2018 ICPC, Gothenburg, Sweden, May 27-28*, pp. 167–177.

[3] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *Proceedings of the 2021 IEEE/ACM ASE*, pp. 492–504.

[4] M. Shetty, C. Bansal, S. Kumar, N. Rao, N. Nagappan, and T. Zimmermann, "Neural knowledge extraction from cloud service incidents," in *Proceedings of the 2021 IEEE/ACM ICSE-SEIP*, pp. 218–227.