Pre-LogMGAE: Identification of Log Anomalies Using a Pre-trained Masked Graph Autoencoder

ERC Summer Workshop 2024

Aming Wu

Advisor: Prof. Young-Woo Kwon

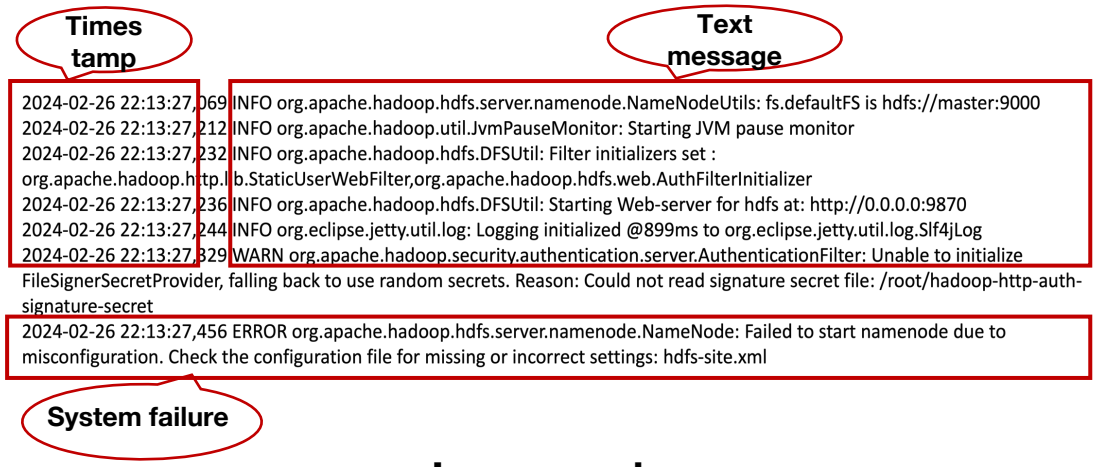
School of Computer Science and Engineering

This work from The 43rd International Symposium Reliable Distributed System (SRDS, 2024)

Background

Significance of analysis system log

- Record system running state;
- Detect potential bugs;
- safety monitoring
- Optimization system performance.



Log sample

Motivation

- **Large scale:** Processing massive log data often takes up a lot of time and space, making anomaly detection inefficient;
- **Lack label:** Most log data is unlabeled, which cannot provide effective data support for supervised learning method;
- **Semi-structured natural property:** Make graph-based method more efficient in capturing complex structural and semantics information.

Challenges and Objectives

Challenges

Limited label data in real-world scenarios

× : Existing methods often rely on large-scale and high-quality label data from massive logs to retrain models;

Information barriers in multi-source log data

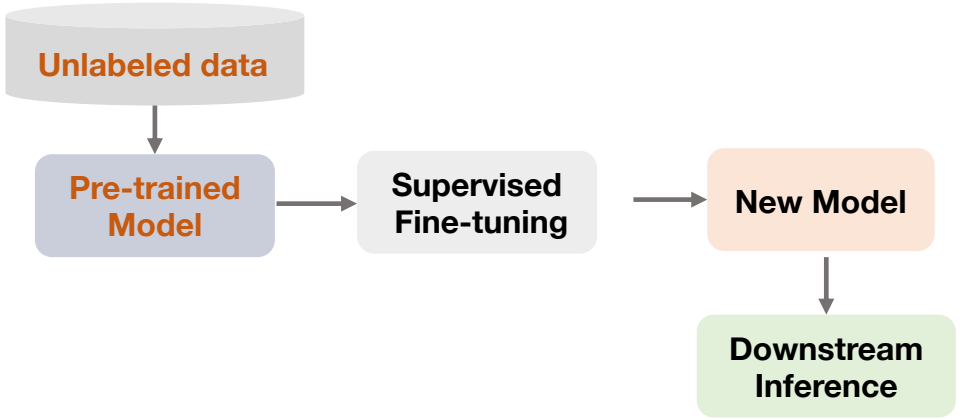
× : Asynchrony of time across different logs can result in challenges in capturing correlations between events or states in multi-source logs (timestamp, pattern, frequency);

Weak generalization ability

× : Previous methods need to retrain from scratch for different domain log data.

Research Objectives

- Develop efficient model training and inference methods;
- Reduce reliance on large-scale labeled data;
- Adapt to diverse log data;
- Enhance the robustness and stability.

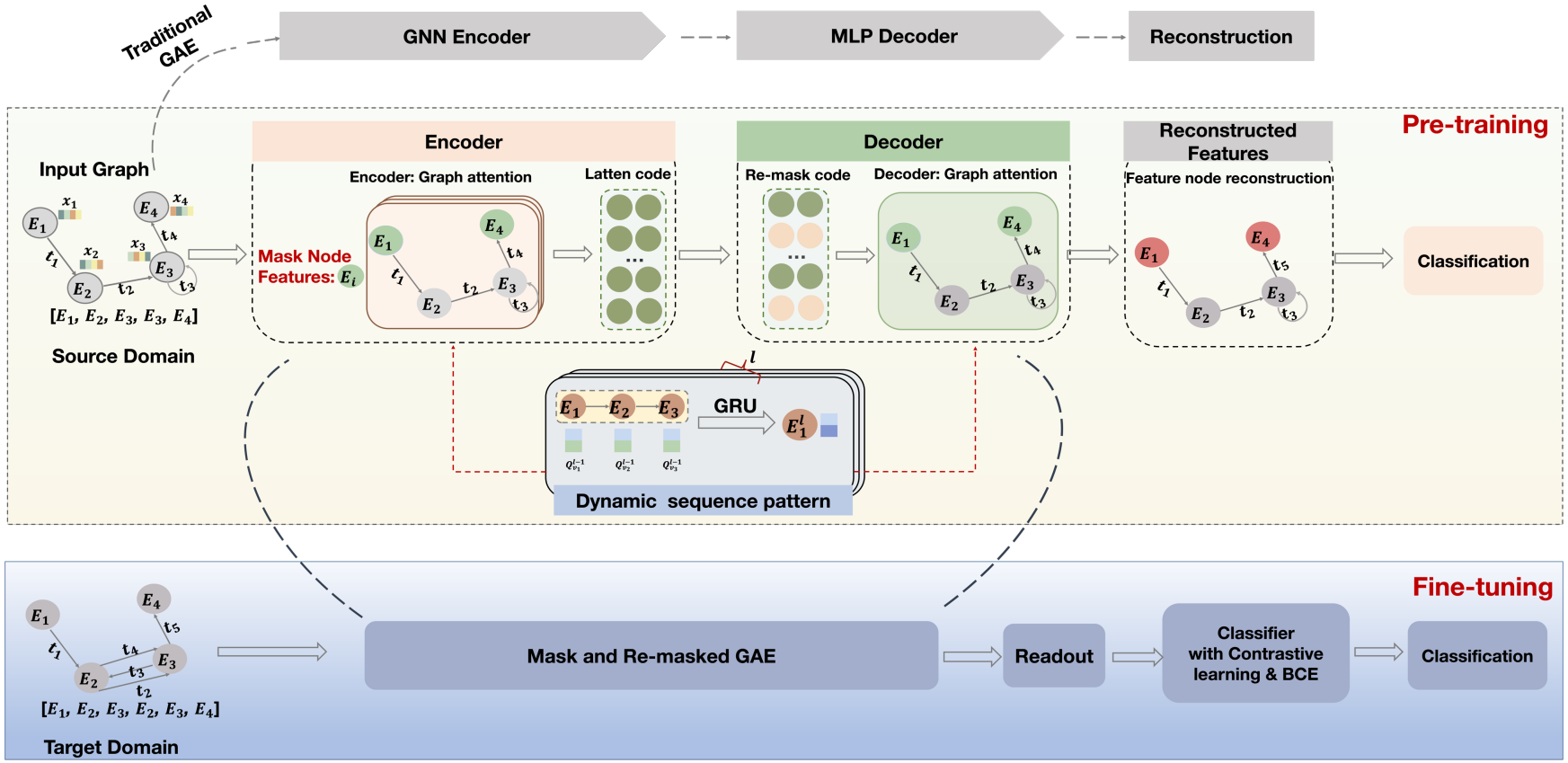


❖ Overflow of supervised fine-tuning

Proposed Pre-LogMGAE Method

Input information

- **Semantic-aware node embedding:**
  - Pre-trained sentence-BERT.
- **Structure encoding:**
  - Degree matrix,
  - Distance matrix,
  - Edge weight matrix.



■ The purpose of pre-training is to learn the basic features through a large amount of unlabeled data

Overview of graph construction

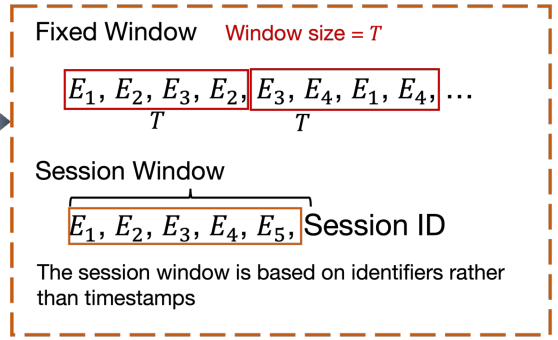
1.- 1117838978 2005.06.03 R02-MI1-NO-C:J12-U11 2005-06-03-15.49.38.026704 R02-MI-NO-C:J12-U11 RAS KERNEL INFO instruction cache parity error corrected
2.- 1117842440 2005.06.03 R23-MO-NE-C:J05-U01 2005-06-03-16.47.20.730545 R23-MO-NE-C:J05-U01 RAS KERNEL INFO
63543 double-hummer alignment exceptions
3.- 1117848119 2005.06.03 R16-M1-N2-C:J17-U01 2005-06-03-18.21.59.871925 R16-MIN2-C:J17-U01 RAS KERNEL INFO
CE sym 2, at 0x0b85cce0, mask 0x05
4.- 1117942120 2005.06.04 R30-MO-N7-C:J08-U01 2005-06-04-20.28.40.767551 R30-MO-N7-C:J08-U01 RAS KERNEL INFO
CE sym 20, at 0x143819el, mask 0x40
5.- 1117955341 2005.06.05 R25-MO-N7-C:J02-U01 2005-06-05-00.09.01.903373 R25-MO-N7-C:J02-U01 RAS KERNEL INFO generating core.2275

Raw Log

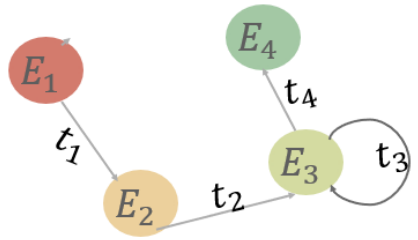
Log Templete:
E1: instruction cache parity error corrctted
E2: <*> double…
E3: CE sym<*>
.
.
.
E5 generating core<*>

Each log message is mapped into a log event:
Log1 →E1 Log2 →E2
Log3 →E3 Log4 →E3
Log5 →E4 Log6 →E5

Log Parsing



Log Group



Graph Construction

Experiment Setting

#Statistics of the datasets

Dataset	HDFS	BGL	Thunderbird
Description	Hadoop distributed file system log	Blue Gene/L supercomputer log	Thunderbird supercomputer log
Data size	1.47G	708.76MB	29.60G
# Messages	11175629	4747963	20000000
# Anomaly	284818	348460	758562
# Anomaly rate	2.93%	7.30%	3.70%
#Error types	53	143	95

- BGL is used for pre-training in this study.
- OwnLog dataset (only with configuration error) is used to explore the case study.

Comparison with Baselines

Type	Experiment Setup	Model	Metrics	Precision/Recall/F1						
				HDFS	BGL			Thunderbird		
				Session	20logs Fiexed	80logs Fiexed	100logs Fiexed	20logs Fiexed	80logs Fiexed	100logs Fiexed
Traditional Statistical Learning	Semantic embedding=BERT Training data =0.8 Validation data = 0.2 Epochs=10~40 Batch size=128 Average over 3 runs	PCA	Precision	0.9583	0.2291	0.2443	0.2647	0.4167	0.4687	0.5924
			Recall	0.4447	0.5922	0.6372	0.6337	0.5337	0.5121	0.6217
			F1	0.6074	0.3301	0.3492	0.3439	0.4973	0.4893	0.5549
		IM	Precision	0.9749	0.4155	0.3123	0.2876	0.3719	0.3934	0.3367
			Recall	0.6325	0.6991	0.4908	0.4734	0.3929	0.4521	0.4132
			F1	0.7678	0.5334	0.3817	0.4219	0.3821	0.4207	0.5431
Sequential-Based		LogRobust	Precision	0.9335	0.9104	0.9301	0.9115	0.8213	0.8117	0.8318
			Recall	0.9578	0.9147	0.9012	0.8917	0.8813	0.8726	0.9012
			F1	0.9453	0.8913	0.9054	0.9019	0.7822	0.7967	0.8649
		DeepLog	Precision	0.9124	0.7513	0.7827	0.7612	0.4813	0.4554	0.4755
			Recall	0.7841	0.8781	0.7622	0.9119	0.5317	0.6168	0.7186
			F1	0.8433	0.7645	0.7723	0.7452	0.6771	0.5246	0.5724
		LogBERT	Precision	0.8679	0.8021	0.8741	0.8611	0.9105	0.8976	0.9321
			Recall	0.7845	0.7911	0.8102	0.9024	0.8733	0.9291	0.9047
			F1	0.8241	0.7965	0.8409	0.8812	0.8915	0.913	0.9181
Graph-Based		DeepTraLog	Precision	0.8577	0.7617	0.7531	0.7742	0.5974	0.6172	0.6367
			Recall	0.9126	0.8198	0.8022	0.8337	0.6801	0.7233	0.7331
			F1	0.8842	0.7344	0.6953	0.7687	0.7855	0.6661	0.6815
		Pre-LogMGAE	Precision	**0.9898 ±0.0045**	**0.9257 ±0.0020**	**0.9723 ±0.0026**	**0.9651 ±0.0020**	**0.8722 ±0.0032**	**0.9346 ±0.0031**	**0.9477 ±0.0076**
			Recall	**0.9917 ±0.0071**	**0.8688 ±0.0026**	**0.9216 ±0.0028**	**0.9316 ±0.0033**	**0.9472 ±0.0021**	**0.9748 ±0.0043**	**0.9734 ±0.0062**
			F1	**0.9903 ±0.0012**	**0.9064 ±0.0046**	**0.9462 ±0.0081**	**0.9481 ±0.0064**	**0.9033 ±0.0016**	**0.9539 ±0.0036**	**0.9603 ±0.0053**

- A changing log sequence (different fix windows) shows proposed method outperforms previous baselines.
- Pre-LogMGAE is feasible to adapt the real-world scenario (different logs).

Ablation Study

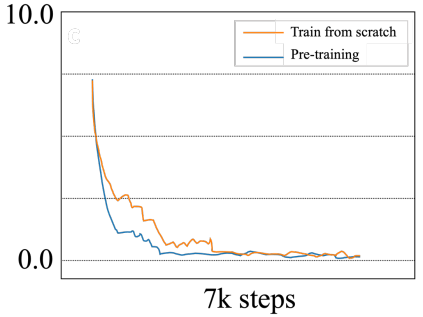
Different decoder types, mask and re-mask strategies

Two-stage masking strategy: Masking selected input nodes and re-masking encoded information before decoding.

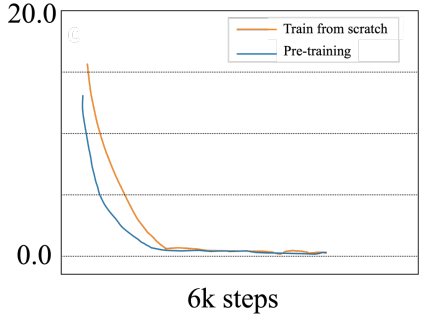
Datasets		F1_score			
		HDFS	BGL	Thunderbird	OwnLog
Component	Pre-LogMGAE	0.9865	0.9466	0.9361	0.8321
	w/o mask	0.9534	0.8647	0.8213	0.7927
	w/o re-mask	0.9620	0.8822	0.8611	0.8065
	w/o contrastive learning	0.9628	0.9021	0.8867	0.8202
Decoder	MLP	0.9142	0.7983	0.7182	0.6275
	GCN2	0.9385	0.7704	0.7601	0.7446
	GATv2	0.9711	0.8697	0.8226	0.8101
	GINE	0.9507	0.8289	0.8323	0.7511

- Decoder type and mask strategy significantly impacts the results;
- Pre-LogMGAE with sequence modeling outperforms other components.

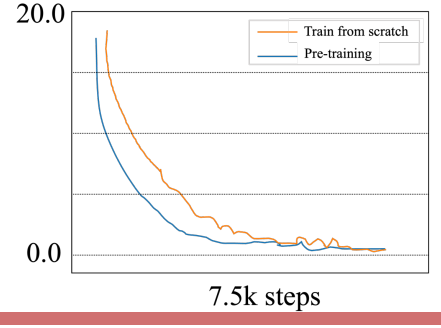
Effect of the pre-training on loss



■ HDFS



■ Thunderbird

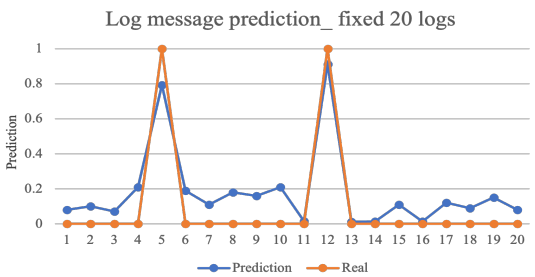


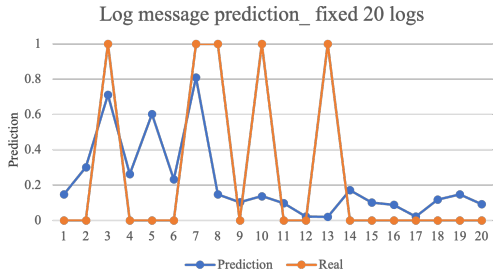
■ OwnLog

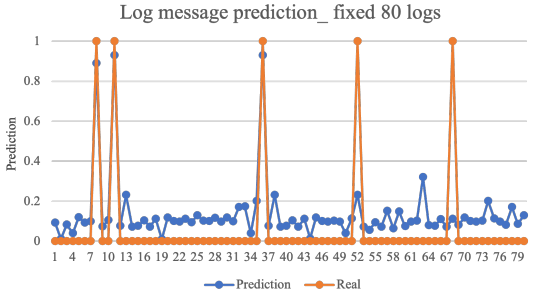
Case study

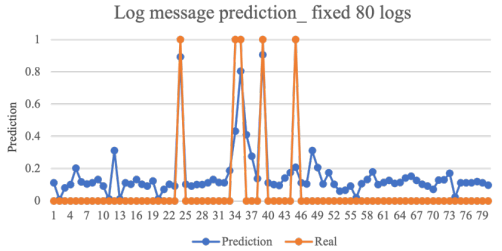
Practical Application

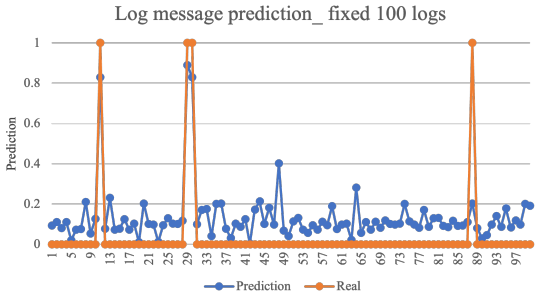
- **Practical Evaluation Data:**
- **GAIA, with the full name Generic AIOps Atlas, is an overall dataset for analyzing operation problems.**
- **OwnLog, only includes system log with configuration error.**
- **Pre-LogMGAE achieve good results on different fixed windows.**

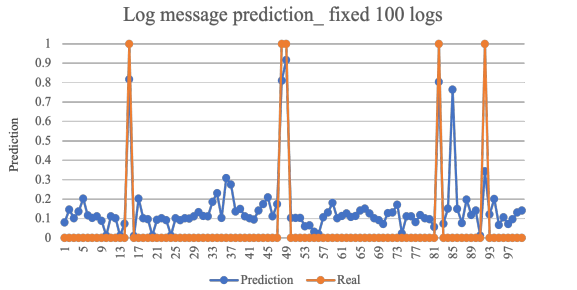












GAIA Data **OwnLog**

Conclusion

Summary

Pre-LogMGAE framework introduces pre-training and fine-tuning graph learning for detecting anomaly logs.

✓ **Extensive experiments confirm that Pre-LogMGAE enhanced adaptability to diverse logs.**

Pre-training strategy	Masking and re-masking reconstruction strategy	Fine-tuning with contrastive learning
■ Reduced dependence on large labeled data and training cost	■ Overcome the issue of overemphasizing unrelated information.	■ Improve and diverse the input expressiveness ability connecting a contrastive learning objective.

✓ **Early Detection and Prevention:** Identify anomaly behaviors, providing early warnings to prevent system failures.

✓ **System Stability and Reliability:** System administrators can quickly find issues, enhancing the system's reliability.

THANK YOU