Transfer Log-based Anomaly Detection with Pseudo Labels

Shaohan Huang*, Yi Liu*, Carol Fung†, Rong He‡, Yining Zhao‡, Hailong Yang*, Zhongzhi Luan*
*Sino-German Joint Software Institute, Beihang University, Beijing, China
†Computer Science Department, Virginia Commonwealth University, Richmond, Virginia, USA
‡Chinese Academy of Scientific Computer Network Information Center, Beijing, China
{huangshaohan, yi.liu, luan.zhongzhi}@buaa.edu.cn

***Abstract*—Log-based anomaly detection is an important task for service management and system maintenance. Although anomaly labels are valuable to learn anomaly detection model, they are difficult to collect due to their rarity. To tackle this problem, existing methods employ domain adaptation algorithms to transfer anomaly detectors from labeled source domain to unlabeled target domain. However, most of those methods focus on key performance indicator anomaly detection. The semantic information in logs plays an important role in log-based anomaly detection. Therefore, adaptation methods need to consider how to transfer the semantic information in logs. In this paper, we propose a simple and effective adaptation method to transfer log-based anomaly detection model with pseudo labels. In our work, we first train a detection model with labeled samples as a pseudo-label annotator. Then we use it to assign pseudo-labels to unlabeled samples and train anomaly detectors as if they are true labels. Both models share the same feature extraction part, which can help model to transfer the semantic information in logs. We evaluated our proposed method on three log datasets. Our experimental results demonstrate that our method has outperformed other baseline methods.**

***Index Terms*—anomaly detection, domain adaptation, transferring**

I. Introduction

Logs play an important role in many complex systems, including internet of things, software systems, computer systems. Through logs, we can observe the status of running systems and know what happened to the system. Current systems produce a huge volume of logs every day but they are difficult to investigate manually. Log-based anomaly detection is an important task for service management and system maintenance.

Many supervised log-based anomaly detection methods have been proposed, such as SVM [1], LR [2], LogRobust [3]. They require labeled anomaly data to train their anomaly detectors. Although anomaly labels are valuable to learn anomaly detection model, they are difficult to collect due to their rarity. To alleviate this problem, existing methods employ domain adaptation algorithms in anomaly detection tasks. In unsupervised domain adaptation, one needs to train an anomaly detector that works well on a target domain when provided with labeled source samples and unlabeled target samples.

978-3-903176-31-7 ©2020 IFIP

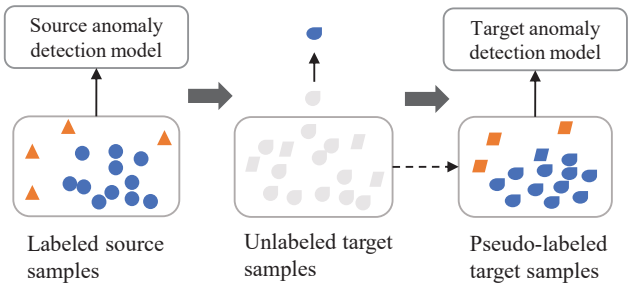

Fig. 1. Overview of transfer anomaly detection with pseudo labels

Existing anomaly detection adaptation approaches [4]–[7], using *key performance indicator* or *indexes* rather than *semantics* of log templates, tend to transfer their models. Atsutoshi [6] designed latent domain vectors to represent different domains, which are used for inferring the anomaly detectors. In [6], autoencoder is used to reduce non-linear dimensionality. It assumed that when the autoencoder is trained with normal instances, the reconstruction errors of normal instances become low. Indexes vectors or key performance indicator vectors are easy to reconstruct. However, it is a challenging problem to reconstruct log data, because log data is plain text and unstructured. Valuable information could be lost if only log indexes are used, because they cannot reveal the semantic relations of logs. For example, some log are similar in semantics but different in indexed. For log-based anomaly detection, the model needs to consider how to transfer the semantic information from logs.

Inspired by [8], we propose a simple and effective adaptation method to transfer log-based anomaly detection model with pseudo labels. As described in Figure 1, given labeled source samples and unlabeled target samples, we first train a detection model with labeled samples as a pseudo-label annotator. Then we use it to assign pseudo-labels to unlabeled samples and train anomaly detectors as if they are true labels. In our work, we design two kinds of features named count vectors and template vectors. Count vectors can be used to capture the quantitative relationships of logs. Template vectors represent the semantic and syntax information from log templates. Both source anomaly detection model and target model

TABLE I
EXAMPLE OF LOG PARSING.

Log message:
Received block blk_-2856928563366064757 of size 67108864 from /10.251.42.9
Log Template:
Received block ⟨*⟩ of size ⟨*⟩ from /⟨*⟩
Parameters:
'blk_-2856928563366064757', '67108864', '10.251.42.9'

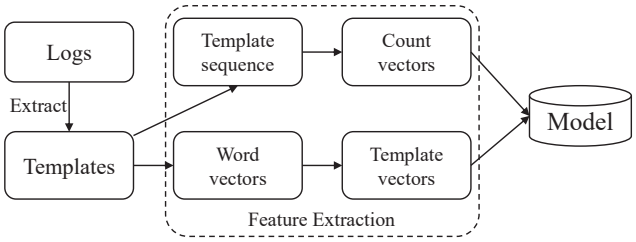

Fig. 2. The framework of log-based anomaly detection model

trained on pseudo-labels share the same feature extraction module, which can help model to transfer the quantitative relationships and semantic information of logs. As shown in Figure 1, source anomaly detection model and target anomaly detection model use the same model architecture, which means that our method is easy to follow and does not need any special implementations.

We evaluated our proposed method on three log datasets including HDFS [9], the BGL dataset [10], and the OpenStack dataset [11], where taking HDFS dataset and BGL dataset as labeled source domain and the OpenStack dataset as unlabeled target domain. Our experimental results demonstrate that our method has outperformed other transferring methods. To the best of our knowledge, this paper is the first work to explore how to transfer a log-based anomaly detector from a labeled source domain to an unlabeled target domain.

II. METHODOLOGY

In this section, we provide details of the proposed model for log-based anomaly detection with domain adaptation. We aim to construct a target specific model by utilizing pseudo-labeled target samples. we first introduce the structure of log-based anomaly detection model. We then show the pseudo labelling method.

A. Log-based anomaly detection model

As illustrated in Figure 2, our log-based anomaly detection model involves three steps: template extraction, feature extraction and classification model.

1) Template extraction: The first step is log parsing. The goal of log parsing is to transfer raw messages content into structured log templates associated with key parameters. As shown in Table I, one log message from HDFS system is parsed into log templates and parameters. Here, $*$ is a wildcard to match parameters. The log template is comprised of fixed text strings, and parameters record some system attributes and variables.

There have been many studies on log parsing. Considering accuracy and speed, we adopt Drain [12] method to parse all log data. After log parsing, we take a sequence log template as input to extract features.

2) Feature extraction: Recently some existing log-based anomaly detection approaches [3], [13] model a log stream as a natural language sequence. For example, LogAnomaly [13] use synonyms and antonyms hidden in log templates to extract the semantic information.

Inspired by those approaches, we design our features from two aspects: count vectors and template vectors. Count vectors can capture quantitative patterns from history logs and template vectors enhance the model to capture the context of semantic information. Consequently, we learn the quantitative pattern of logs as follows. For example, one dataset contains 4 templates, denoted as $\{S_1, S_2, S_3, S_4\}$. For a log sequence $\mathbf{S} = \{S_2, S_1, S_2, S_3, S_1\}$, we calculate the count vector of the log sequence $\mathbf{S}$ as $\mathbf{C} = [c(S_1), c(S_2), c(S_3), c(S_4)]$, which is $[2, 2, 1, 0]$. $c(S_i)$ is the number of S_i in the template vector sequence.

For template vectors, we leverage a transformer model [14] to convert a sequence log templates into a fixed vector. As shown in Figure 2, we first maps each word in log template to a word vector x. The word vector list x flows through the transformer architecture [14]. The transformer is a deep machine learning model introduced in 2017, which can efficiently handle large inputs and outputs. This architecture generates the respective log template and an embedding vector for each log message. After the transformer block, the outputs are fed into an average pooling layer, which reduces the number of weights and transforms a template into a fixed-dimension vector $\mathbf{T}$.

$$X = TransformerBlock(x) \tag{1}$$

$$\mathbf{T} = Average(X) \tag{2}$$

Count vector $\mathbf{C}$ and template vector $\mathbf{T}$ as our features are passed forward through the classification model.

3) Classification model: We use two multilayer perceptron modules to compute the final score. The final score consist of two parts. The score of count vector s_c and score of template vector s_t are computed as follows.

$$s_c = \texttt{sigmoid}(\mathbf{C}\mathbf{W_c}) \tag{3}$$

$$s_t = \texttt{sigmoid}(\mathbf{T}\mathbf{W_t}) \tag{4}$$

where $\mathbf{W_c}$ and $\mathbf{W_t}$ are both parameters of the classification model. The final score is introduced to represent probability distribution of anomaly:

$$y = \frac{s_c + s_t}{2} \tag{5}$$

The model is trained using the Stochastic Gradient Descent (SGD) [15] algorithm and we use the cross-entropy as the loss

Algorithm 1: Domain Adaptation with Pseudo Labeling

Input: Trained on source data model F, empty dataset D, unlabeled target samples S, target samples number N, threshold t.
while $j < N$ **do**
  Get $s \in S$
  Compute s_c, s_t and y with model F
  if $\max(s_c, s_t) > 0.5$ and $y > t$ **then**
    Get pseudo label l
    Append s and l to D
  end if
  j = j + 1
end while
Train target anomaly detection model F_T with D
Output: F_T

function. Source anomaly detection model and target anomaly detection model use the same model architecture and training method, which means that our method is easy to implement and does not need any special implementations.

B. Pseudo labeling method

Pseudo-labeled target samples will be used to train the target anomaly detection model. However, since they certainly contain false labels, we have to pick up reliable pseudo-labels. Our labeling is aimed at realizing this.

First, we train our source anomaly detection model F with a labeled source training set. After training, we use prediction of F to provide pseudo-labels. For a sequence log s from target domain, we obtain its counting score s_c, template score s_t, and its final score y. In order to pick up reliable pseudo-labels, we propose two conditions. If the following two conditions are satisfied, we will assign a pseudo-label to x. First, we require $\max(s_c, s_t) > 0.5$, which means one of two kinds of features has a high prediction score. The second requirement is that the final score y exceeds the threshold, which we set as 0.7. We suppose that if only one of two classifiers is confident of the prediction, the prediction is not reliable. In our experiments, we will discuss the impact of different thresholds. The entire procedure of training the network is shown in Algorithm 1.

III. Experiment

In this section, we first describe the experimental dataset and evaluation metrics. We then show the performance of our method in different domain adaptation settings.

A. Experiment Setting

1) Datasets: We use three datasets[1] to conduct anomaly detection transferring experiments. We list the summary statistics for the log datasets in Table II. The HDFS dataset [9] and the OpenStack dataset [11] are both generated from distributed systems. The BGL dataset [10] is collected from a BlueGene/L supercomputer system. The detailed information on the three datasets are described as follows:

[1]https://github.com/logpai/loghub

(1) HDFS: The HDFS dataset is generated through running Hadoop-based jobs on more than 200 Amazon's EC2 nodes. The HDFS dataset consists of 11,175,629 logs and each log contains a unique block ID. There are 575,061 blocks, among which 16,838 blocks were labeled as anomalous. All anomaly blocks are labeled by Hadoop domain experts.

(2) OpenStack: OpenStack is a cloud operating system that controls large pools of computer, storage, and networking resources. This data set contains 70,746 logs. OpenStack data is grouped into different sessions by instance ID. There are 503 instances labeled as anomalous.

(3) BGL: BGL is an open dataset of logs collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs. The BGL dataset contains 4,747,963 logs. Each BGL log was manually labeled as either anomalous or normal, and 348,460 logs were labeled as anomalous. We use fixed windows to slice logs as log sequences. We label a log sequence as anomalous if the sequence contains anomaly logs.

TABLE II
Statistics of log data.

Datasets	System Type	# of logs	# of anomalies
HDFS	Distributed systems	11,175,629	16,838 (blocks)
OpenStack	Distributed systems	70,746	503 (instances)
BGL	Supercomputers	4,747,963	348,469 (logs)

In the following experiments, we randomly sample 80% data as the training data, and the remaining 20% as the testing data. For the HDFS dataset, anomalies were manually confirmed rather than manually labeled. For the BGL dataset, the system administrators determined the subset of log entries that they would tag as being alerts. For the OpenStack log data set, the author deployed OpenStack with ten nodes. INFO level logs were collected. Three types of anomalies were injected at different execution points. We regard HDFS dataset and BGL dataset as labeled source domain and OpenStack dataset as unlabeled target domain.

2) Evaluation Metrics: Anomaly detection is a binary classification task. We use Precision, Recall and F1-Score as our evaluation metrics, which are commonly used in anomaly detection tasks [16].

3) Baselines: We compare our method with some unsupervised domain adaptation baseline methods. These method are briefly described as follows:

- Train on source: Source anomaly detection model trained on labeled source samples is applies to infer target samples directly.
- One-class support vector machine (OSVM) [17]: OSVM is a semi-supervised anomaly detection method, which uses only normal instances in the target domain for training.
- k-Nearest Neighbor based adaptation (kNN-Ad) [18]: Similar to our method, kNN-Ad method uses k-Nearest Neighbor to assign pseudo-labels. In our experiments, we set k as 5.

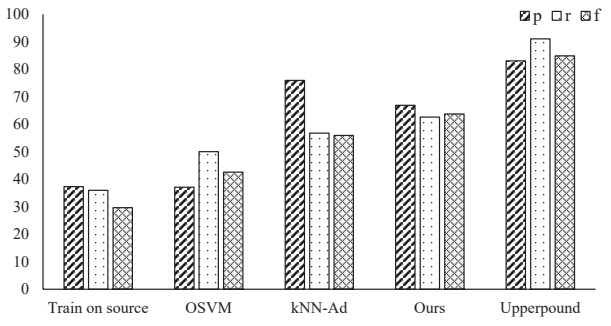

Fig. 3. Evaluation on HDFS → OpenStack

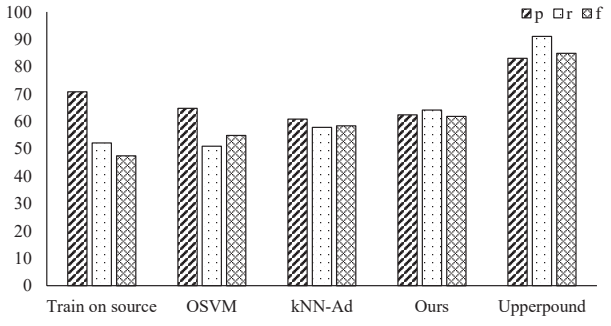

Fig. 4. Evaluation on BGL → OpenStack

- Upper pound: We use labeled target samples to train log-based anomaly detection model, which is regarded as upper pound.

B. Experimental results

Figure 3 shows the performance of our method compared to other baseline methods when the HDFS dataset is the labeled source dataset. Our method achieves the highest accuracy among the those methods, having an F1 score of 63.73. We can observe that only training on source dataset results the lowest performance, which means that there exists a huge gap between HDFS data set and OpenStack dataset. Compared to only training on source data, OSVM slightly increases in precision score, indicating that one-class method is hard to detect anomaly in very different domain. kNN-Ad method shows high precision on this data set but the recall rate is low (less than 60%). Our method is slightly better than kNN-Ad method. This is because both methods use pseudo-labelling to train the target domain detection model. However, our method is easier to implement and costs less computation than kNN method.

We also observe that there is still room for improvement compared to the upper bound, which demonstrates transferring anomaly detection between different domains will improve in the long term.

Figure 4 presents the results when transferring the trained BGL model into OpenStack. The results demonstrate that our proposed model outperforms other baseline models on the source BGL data set, with an F1-score of 61.87%. Our method achieves the highest recall score in this setting, which demonstrates that our pseudo labeling method can collect various anomaly data.

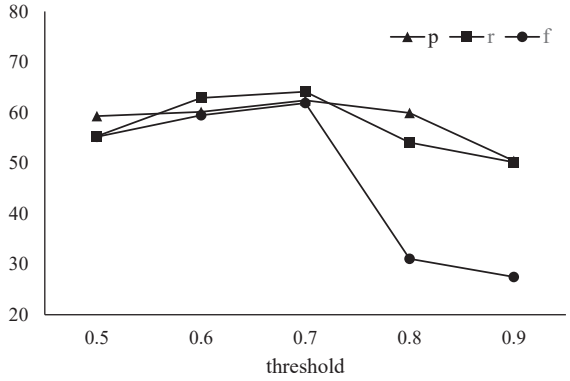

Fig. 5. The performance of different thresholds

Data from this table can be compared with the data in Table II which shows that domain adaptation method achieves better performance where the source domain is BGL. This may be because BGL log data is more similar to OpenStask dataset than HDFS dataset. Figure 4 also present the upper bound results. Compared to upper bound, domain adaptation methods still remain to be enhanced.

C. Impact of different thresholds

We have conducted experiments to study how our method performs under different thresholds. We show results of taking BGL as source domain dataset. Figure 5 shows that the threshold t has impact on the accuracy of anomaly prediction. The F1 score drops dramatically when the threshold is more than 0.7. Though the quality of pseudo label increases when the threshold is higher, our method cannot collect enough data with a high threshold. With few pseudo labeled data, the target domain anomaly detection model is under-fitting. Under a low threshold, our method can annotate more pseudo labels. However, it brings more noise and makes detection model worse.

IV. Conclusion

This paper proposed a simple and effective adaptation method to transfer log-based anomaly detection model with pseudo labels. We divided our features into two categories and proposed two conditions to help pick up reliable pseudo-labels. Our experimental results demonstrated that our method has outperformed other baseline methods. Through our experiments, we can observe that our method has improved the performance of domain adaptation, there is still room for improvement compared to the upper bound. A further study with more focus on log-based anomaly detection transferring is therefore suggested.

ACKNOWLEDGMENT

This research is supported by National Key R&D Program, under the grant No. 2018YFB0204002; National Science Foundation of China, under the grant No. 61732002.

REFERENCES

[1] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "Towards automated log parsing for large-scale log data analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 931–944, 2017.

[2] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in ibm bluegene/l event logs," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 583–588.

[3] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. 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*, 2019, pp. 807–817.

[4] J. Andrews, T. Tanay, E. J. Morton, and L. D. Griffin, "Transfer representation-learning for anomaly detection." JMLR, 2016.

[5] R. Chalapathy, A. K. Menon, and S. Chawla, "Anomaly detection using one-class neural networks," *arXiv preprint arXiv:1802.06360*, 2018.

[6] A. Kumagai, T. Iwata, and Y. Fujiwara, "Transfer anomaly detection by inferring latent domain representations," in *Advances in Neural Information Processing Systems*, 2019, pp. 2467–2477.

[7] V. Vercruyssen, W. Meert, and J. Davis, "Transfer learning for anomaly detection through localized and unsupervised instance selection." in *AAAI*, 2020, pp. 6054–6061.

[8] K. Saito, Y. Ushiku, and T. Harada, "Asymmetric tri-training for unsupervised domain adaptation," *arXiv preprint arXiv:1702.08400*, 2017.

[9] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 117–132.

[10] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. IEEE, 2007, pp. 575–584.

[11] M. Du, F. Li, G. Zheng, and V. 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*, 2017, pp. 1285–1298.

[12] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 33–40.

[13] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization*, vol. 7, 2019, pp. 4739–4745.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[15] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[16] S. Huang, C. Fung, K. Wang, P. Pei, Z. Luan, and D. Qian, "Using recurrent neural networks toward black-box system anomaly prediction," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 2016, pp. 1–10.

[17] G. Yin, Y. Zhang, and Z. Zhao, "A novel computer network intrusion detection algorithm based on osvm and context validation," in *2016 International Conference on Progress in Informatics and Computing (PIC)*. IEEE, 2016, pp. 591–595.

[18] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European conference on computer vision*. Springer, 2010, pp. 213–226.