Towards an NLP-based log template generation algorithm for system log analysis

Satoru Kobayashi
University of Tokyo
sat@hongo.wide.ad.jp

Kensuke Fukuda
NII
kensuke@nii.ac.jp

Hiroshi Esaki
University of Tokyo
hiroshi@wide.ad.jp

ABSTRACT

System log from network equipment is one of the most important information for network management. Sophisticated log message mining could help in investigating a huge number of log messages for trouble shooting, especially in recent complicated network structure (e.g., virtualized networks). However, generating log templates (i.e., meta format) from real log messages (instances) is still difficult problem in terms of accuracy. In this paper we propose a Natural Language Processing (NLP) approach to generate log templates from log messages produced by network equipment in order to overcome this problem. The key idea of the work is to leverage the use of Conditional Random Fields (CRF), a well-studied supervised natural language processing technique. As preliminarily evaluation, with one month network equipment logs in a Japanese academic network, we show that our CRF based algorithm improves the accuracy of generated log templates in reasonable processing time, compared with a traditional method.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*logging*; C.2.3 [**Computer-communication networks**]: Network operations—*network management*; I.2.7 [**Artificial intelligence**]: Natural language processing—*language parsing and understanding*

General Terms

Reliability, Management, Languages

1. INTRODUCTION

Analyzing system logs from network routers and switches helps in detecting troubles and root causes of them in network management [5]. Especially in network management under recent and future complicated network structure with virtualized techniques, the number of logs will increase and dependency of each log will be complicated because of complexity of physical and virtual nodes. It is a time-consuming task for network engineers to check a huge number of system log messages in real-time. Clearly, automated log mining techniques could improve the efficiency and reliability of the network system [3]. However, many research problems are remained for such intelligent support [4]. One of them is that the most important information in system log must be extracted through contextual information of log messages and information on environment (i.e., configuration of the system), rather than single log message.

In order to extract contextual log messages, at first, we require to obtain log templates (i.e., meta format) that exclude variable information like IP address and device name from system logs (i.e., log instances). This enables us to analyze normalized log templates rather than instances of them. For example, log templates can be compared by analyzing time series data of normalized log templates. We can identify the semantic relation of log templates from this information. However, generating log templates from raw log messages is not a simple problem, because the output templates of log messages are not available in many cases, particularly on commercial systems. In addition, existing methods show poor performance of obtaining log templates from log instances. Less reliable log templates frequently lead to miss of finding appropriate log messages related to trouble itself and its root cause. This is a serious problem for further analysis of log messages though this has been heuristically solved in the past literature [7, 10].

We propose a Natural Language Processing (NLP) based approach to tackle this problem. Generating log output templates can be considered as a problem of labeling sequential data in NLP. Specifically, we leverage Conditional Random Fields (CRF) [2, 6] to learn the structure of log messages for obtaining log templates. Our preliminary evaluation with 1 month network equipment logs demonstrates that our CRF-based algorithm achieved more than 99 percent accuracy in word-level comparison in reasonable processing time, while an existing method shows 70 percent of accuracy.

2. RELATED WORK

Xu et al. [9] investigate root causes of failures in operating system and distributed system by examining system logs. Their analysis relies on log templates obtained from source code of system software. However, this approach can be applied only for open source software, and thus its applicability is limited because most source codes of commercial

CFI '14 June 18 - 20 2014, Tokyo, Japan

ACM 978-1-4503-2942-2/14/06 ...$15.00.
http://dx.doi.org/10.1145/2619287.2619290 .

products are not available. Moreover, large scale network is usually built with switches or routers provided by multiple commercial vendors.

Yamanishi et al. [10] define log templates produced by ATM switches by removing numerical descriptions such as IP addresses. This simple algorithm ignores lexical variables without numeric ones, though the main focus or their work is not on defining log templates correctly. The system they analyzed is relatively simple, and the number of potential log templates are small. In this case this simple heuristic algorithm works well, but cannot be used commonly.

Vaarandi [7, 8] proposes an efficient algorithm to extract log templates from system log. First, the algorithm count the number of appearances of each words in a set of logs. Then, it defines words that appear more frequently than a threshold as "descriptions", and defines the others as "Variables", while the threshold should be determined empirically. Finally, it generates a log template by replacing variables to wildcard symbols. Vaarandi's algorithm is based on the assumption that description words appear more frequently than variable words in system log. However, this assumption is not always correct. For example, words like IP address appear more frequently than description words in large-scale system logs. This is because IP addresses appear in wide variety of log messages, in contrast to description words. Similar situation occurs widely in words like numerical values, state strings, device names, and user names. In this case, these variables will be judged as "descriptions". Moreover, this algorithm is affected by popularity of log templates. For example, a number of description words in infrequent log templates is less than that of variable words in frequent log templates. Thus, the formers can be considered as "variables". From this reason, the accuracy of Vaarandi's algorithm is still limited.

3. APPROACH

The goal of this work is to develop a more reliable method for reproducing accurate log templates. However, it is difficult to process logs accurately with methods that only consider logs as a set of single words. Therefore, we rely on an approach to apply NLP techniques, which have been well investigated, to log instances. In general, description of system logs is far from natural language. However, there are some structural patterns in log messages common to natural language. For example, suppose that word "from" is followed by variable descriptions, which are not part of log template in system logs. Indeed, this rule pattern is frequently appeared in real logs. This kind of information is useful in predicting the structure of log messages. Thus, it is reasonable that some NLP techniques can reveal structure of system logs in the same way as for natural language. In order to label which word is a part of log templates or not, we rely on Conditional Random Fields [2, 6] that is a well studied technique in NLP field.

3.1 Conditional Random Fields (CRF)

Conditional Random Fields (CRF) [2, 6] is a supervised learning technique for segmenting or labeling sequential word data. It is, for example, used for chunking of speech texts. Train data consists of log messages and the labeled data of their words. CRF calculates conditional probability of each log message with feature functions, the probability of positional relations of words, and their features defined with

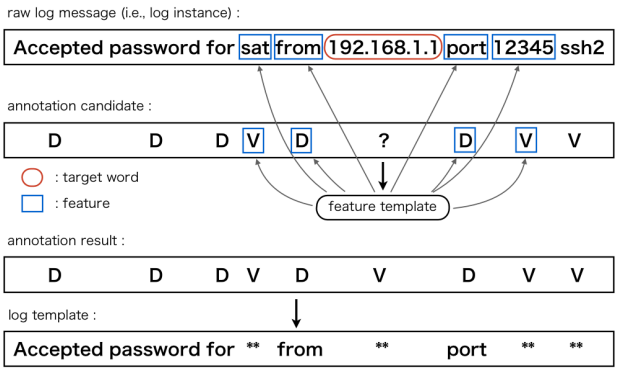

Figure 1: Overview of CRF with feature templates.

feature templates. Feature template consist of positional relations of words and labels to be used in making feature functions. This enables CRF to avoid useless calculation and over-fitting. For example, feature functions are defined as in Figure 1; Feature templates assign two neighboring words and labels before and after the target word. Solving a maximization problem of the conditional probability is used for learning relationship between other words and their features. Finally, CRF classifies each words in unlabeled test data into "descriptions" and "variables". This procedure shows that CRF evaluates the structure of whole words in a single log message. This is an advantage of CRF comparing with other labeling algorithms like Hidden Markov Model (HMM) [1].

Here, we briefly explain a procedure of CRF[1]. For training, CRF requires feature templates and training data. The first step of labeling words of system log is to prepare a train data set with correct labels: "description" or "variable", meaning that the word is a part of log templates or not. For example, words of log instances are annotated, where D and V correspond to "Description" and "Variable", respectively. Next, CRF reads the learning data and constructs a CRF model consisting of conditional expressions made by learning data and feature templates. Finally, the CRF model classifies every line of target logs. The output is its log template, the description of log message with its variable string replaced with wildcard.

3.2 Data set

We investigate a set of log data that is collected at backbone routers and L2 switches at a research and education network in Japan. Three different types of logs are included in the dataset corresponding to three vendors of equipment. We preliminarily analyze one month log data in following experiments. This data consists of 2,572,621 log messages and can be heuristically classified into 201 log templates. We construct labeled data by using a self-made simple regular expression-based analyzer for train data and ground truth data.

Figure 2 shows the distribution of the number of appearances of log instances per log template in the data set. It is clear that the plots have a long tail, meaning that a small number of log templates appears more frequently. Indeed, only top 21 patterns appear over 1000 times while 35 log

[1]We used CRF++ as an implementation of CRF. http://code.google.com/p/crfpp/

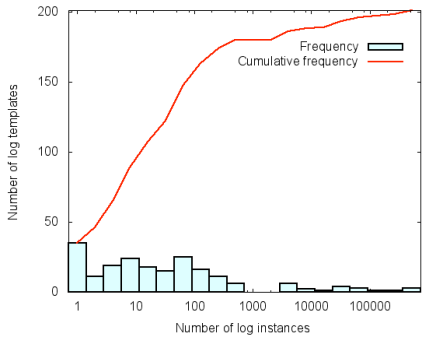

Figure 2: Frequency distribution of log templates.

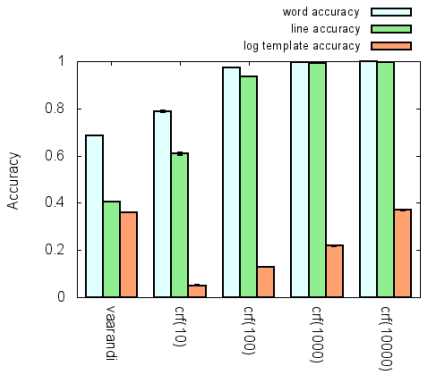

Figure 3: Accuracy of the proposed algorithm and Varrandi's algorithm.

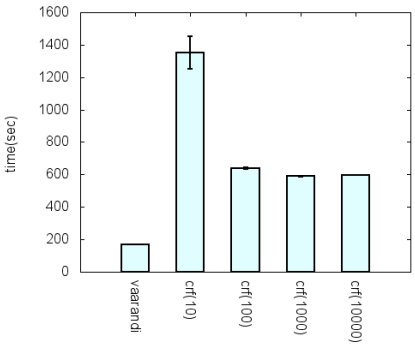

Figure 4: Processing time of two algorithms.

templates appear only once; however 1,478,386 log messages belong to them. In other words, they account for 57 percent of all log messages. This data shows that the actual system log data is characterized by a skewed distribution.

4. EVALUATION

4.1 Performance Metrics

In order to evaluate performance improvements of the CRF-based method, we conduct multiple experiments with two log template generation algorithms (i.e., Vaarandi's and the proposed one) with different parameter sets. For each experiment we perform 20 trials with train data selected randomly and independently, and we show their average and standard error, considering dependencies on the selection of the train data set. The performance metric we used is as follows:

4.1.1 Accuracy

Accuracy is defined as the rate how correctly the algorithm can generate appropriate log templates. We consider three levels of accuracy in this paper:

- Word accuracy : Assigned label is validated in word level. This checks whether the label of a description word (or variable word) is "description" (or "variable"), or not.
- Line accuracy : Assigned labels are validated in a log message. It is failed if more than one label in a log message does not match.
- Log template accuracy : This is the averaged line accuracy weighted by the number of appearance of log messages.

4.1.2 Reduction rate

The log template generation algorithms make 1 log message into 1 log template, and same templates from multiple log messages are merged. However, if labeling variable words fails, log templates which have different variable words cannot be aggregated. Thus, the number of the final log templates become large. In other words, incorrect labeling of words generates unnecessary and verbose log templates. As a consequence, the number of the final log templates is an alternative metric to measure the appropriateness of the algorithm. such patterns will not decrease the total number of log templates. A reduction rate of log templates is defined as the ratio of the number of all log messages to the number of generated log templates, i.e., a larger reduction rate is better.

Table 1: Comparison of the number of output log templates and the ratio to original log messages (2572621)

Algorithm	Templates	Reduction rate
Regular expression	201	12799
Vaarandi's algorithm	1302	1975
CRF (with 10 train data)	85750	41
CRF (with 100 train data)	13649	884
CRF (with 1000 train data)	1156	2237
CRF (with 10000 train data)	960	2717

4.1.3 Processing Time

The processing time includes not only classification parts, but also making a word dictionary in Vaarandi's algorithm and making CRF models by learning train data.

4.2 Results

We compare the proposed algorithm with Vaarandi's one. Figure 3 indicates the comparison of three types of accuracy with error bars: word, line, and log template accuracies. A CRF with sufficient train data achieves 30 percent improved in word accuracy and 60 percent improved in line accuracy than Vaarandi's algorithm. As for these two metrics, our method indicates totally higher accuracy.

In addition, we can see that the more learning data given, the more accurately CRF produces log templates. Clearly a sufficient number of train data enables us to analyze less frequently appearing log templates. Therefore, it is reasonable that a log template can be analyzed correctly if train data include some of similar templates. However, in terms of log template accuracy, CRF requires 10000 train data to achieve same accuracy as Vaarandi's algorithm.

As described in Sec. 2, Vaarandi's algorithm cannot produce log templates correctly from log instances including variable words (e.g., IP addresses) that appear on multiple log templates. Such logs, which are majority in the dataset, yield low word and line accuracies of Vaarandi's algorithms. On the other hand, system log messages without major vari-

ables can be parsed correctly. Therefore, we conclude that Vaarandi's algorithm successfully parse a part of minor log templates.

In our CRF algorithm, major variable words like IP addresses are correctly annotated with high probabilities. Clearly a sufficient number of train data enables us to analyze less frequently appearing log templates. Therefore, it is reasonable that a log template can be analyzed correctly if train data include some of similar templates, because the same variable word is likely to appear in train data and has more chance to be learned. However, our method failed to annotate words in infrequent log templates correctly, in contrast with Vaarandi's algorithm which failed in log templates containing frequent variable words. Figure 2 shows that the number of log instances in 159 log templates have less than 100. In other words, a probability of appearing such a log templates is less than 4 percent in 1000 train data. This result demonstrates that the performance of the proposed algorithm largely depends on the selection of train data set, and CRF is potentially poor at extracting minor log templates with random selection of train data.

Table 1 lists the number of generated log templates of each algorithms and their corresponding rate. The raw of "Regular expression" indicates the ideal case of log template generation. The result of reduction rate is similar to that of the log template accuracy. However, our algorithm achieved similar reduction rate to that of Vaarandi's algorithm with smaller train data, comparing with the log template accuracy. This is because reduction rate decreases largely with major log templates parsed correctly.

The result of processing time (Fig. 4) shows that Vaarandi's algorithm has advantage over our method, but our method works in reasonable time. If a sufficient amount of learning data are provided, our method takes time from 3 to 5 times of Vaarandi's algorithm. Even if the size of test data increases, this relations remains as it is, because our method processes each log messages independently.

Additionally, we can see that CRF processes test data in less time with a sufficient number of learning dataset. It takes more time to construct a CRF model with more learning data. However, its impact is small as far as learning data is considerably smaller than test data; learning 10000 data takes only 5 seconds, though annotation takes 592 seconds. The time to process log messages with CRF model depends heavily on the number of learning data. This is likely due to pruning process in implementation of CRF. If there is dominant information in learning data, unnecessary calculation can be omitted by pruning process. An enough number of learning data that contain essential information reduces the processing time of a CRF model.

5. CONCLUDING REMARKS

In order to extract contextual information from system log through classifying log messages by their log templates and comparing time series data of the appearance of each log template, it is necessary to generate log templates from raw system log. We proposed a method to do this by learning the structure of log messages. In many cases, system logs are characterized by a skewed distribution in the appearance of each log templates, and this feature prevents us from analyzing actual system log with simple approaches. The key idea of the proposed algorithm relies on the use of CRF, a recent natural language processing technique. The accuracy of our proposed method basically outperformed that of a traditional algorithm.

However, our CRF-based algorithm still has a remaining issue that it fails minor log messages with a limited number of train data. There are two potential solutions to this. One is to combine CRF with Vaarandi's algorithm. Section 4.2 indicates that these two algorithms have different types of mislabeling in generating log templates; Vaarandi's algorithm cannot annotate variable words like IP addresses that appear in multiple log templates, and the CRF algorithm is poor at labeling words of minor log templates. However, these methods achieved similar log template accuracy. Therefore, it is plausible to generate log templates more accurately with a hybrid method. Second one is an approach to select train data set based on the structure of log messages. If train data consists of log messages that are similar to a particular one, CRF cannot annotate log messages with different structure from that. In order to generate log templates correctly with a limited number of train data, it is expected that train data includes as various log templates as possible. This could be achieved by feedback (or boosting) to select better train data set, e.g., online learning. We will continue to improve the accuracy of parsing log messages with CRF from two above-mentioned view points.

6. REFERENCES

[1] D. Freitag and A. McCallum. Information extraction with HMM structures learned by stochastic optimization. In *AAAI*, pages 584–589, 2000.

[2] J. Lafferty, A. Mccallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[3] C. Lim, N. Singh, S. Yajnik, A. Labs, and B. Ridge. A log mining approach to failure analysis of enterprise telephony systems. In *IEEE DSN*, pages 398–403, 2008.

[4] A. Medem, M. Akodjenou, and R. Teixeira. Troubleminer: Mining network trouble tickets. In *IFIP/IEEE IM*, pages 113–119, 2009.

[5] T. Qiu, G. Tech, Z. Ge, F. Park, D. Pei, and J. Xu. What happened in my network? Mining network events from router syslogs categories and subject descriptors. In *ACM IMC*, pages 472–484, 2010.

[6] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *NAACL*, pages 134–141, 2003.

[7] R. Vaarandi. A data clustering algorithm for mining patterns from event logs. In *IEEE IPOM*, pages 119–126, 2003.

[8] R. Vaarandi. Real-time classification of IDS alerts with data mining techniques. In *IEEE MILCOM*, pages 1–7, 2009.

[9] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan. Detecting large-scale system problems by mining console logs. In *ACM SOSP*, pages 117–131, 2009.

[10] K. Yamanishi and Y. Maruyama. Dynamic syslog mining for network failure monitoring. In *ACM KDD*, pages